2025 AES International Conference on Artificial Intelligence and Machine Learning for Audio

Queen Mary University of London

8th to 10th, September 2025

Proceedings

Audio Engineering Society

Queen Mary University of London

centre for digital music

These proceedings are edited by:

Chin-Yun Yu (Late Breaking Demo Papers Chair/ Proceedings compiler, QMUL)

George Fazekas (Papers Co-Chair, QMUL)

Brecht de Man (Papers Co-Chair, AES)

Josh Reiss (General Chair, QMUL)

Organising Committee

Josh Reiss (General Chair, QMUL)

Soumya Vanka (Special Sessions Co-Chair, QMUL)

Franco Caspe (Special Sessions Co-Chair, QMUL)

George Fazekas (Papers Co-Chair, QMUL)

Brecht de Man (Papers Co-Chair, AES)

Hayley Cork (Facilities Chair, QMUL)

Farida Yusuf (Sponsorship Chair, QMUL)

Emmanouil Benetos (Publicity Chair, QMUL)

Nelly Garcia (Social Events Coordinator, QMUL)

Ilias Ibnyahya (Treasurer, QMUL)

Chin-Yun Yu (Late Breaking Demo Papers Chair, QMUL)

Aikaterini Maria Primenta (Invited Speakers Coordinator, QMUL)

Jayant Manchanda (Web Content Coordinator, QMUL)

Pedro Sarmento (MusicAI)

Sohyun Im

Shuoyang Zheng (QMUL)

Christopher Mitcheltree (QMUL)

Gary Bromham

Hidetaka Imamura (YAMAHA)

Bradford Swanson (ProSound Effects)

David Marttila (QMUL)

Yisu Zong (QMUL)

Yorgos Velissaridis (QMUL)

Minhui Lu (QMUL)

Florian Schmid

Paul Primus

Jonathan Greif

Reviewers

Adrian Artacho

Aidan Hogg

Akito van Troyer

Alec Wright

Alexander Refsum Jensenius

Alexander Williams

Alexey Lukin

Alistair Carson

Andrew Horner

Angeliki Mourgela

Anna Xambó

Benoit Alary

Bo Holm-Rasmussen

Bradford Swanson

Carla Scaletti

Champ Darabundit

Chin-Yun Yu

Christian Dittmar

Christian Uhle

Christos Chousidis

Christos Plachouras

Cumhur Erkut

Diamantino Freitas

Eloi Moliner

Elvira Burdiel

Emilia Stefanowska

Emmanouil Benetos

Farida Yusuf

Filipe Cunha Monteiro Lopes

Francis Stevens

Gavin Barrett-Hayes

George Tzanetakis

Gloria Dal Santo

Gordon Wichern

Gualtiero Volpe

Harshal Shah

Herve Bouley

Ilias Ibnyahya

Isabel Barbancho

Jamie Angus

Jan Skoglund

Jean-Marc Jot

Jiawen Huang

Jingjing Tang

Jorge Garcia

Jose Luis Blanco

Joseph Colonel

Karolina Prawda

Keith McElveen

Ken Felton

Lars Engeln

Laurent Pottier

Lin Wang

Luca Turchet

Mary Pilataki

Maryam Safi

Mathieu Lagrange

Matija Marolt

Matthew Pitkin

Michele Geronazzo

Mike Newton

Nicki Holighaus

Nikhil Singh

Nils Peters

Orchisama Das

Pavel Rajmic

Pedro Pestana

Pedro Sarmento

Philippe Esling

Riccardo Giampiccolo

Riccardo Simionato

Robert Rowe

Romain Michon

Russell Mason

Sebastian Schlecht

Sergio Canazza

Shuoyang Zheng

Stefan Bayer

Stefania Cecchi

Stefano D'Angelo

Stefano Damiano

Stephane Ragot

Thaddeus Paez

Thomas Mckenzie

Tony Tew

Vesa Välimäki

Vincent Lostanlen

Vlad Baran

Yannis Vasilakis

Yinghao Ma

Yisu Zong

Yorgos Velissaridis

Sponsors

Silver

Challenge Sponsor

Awards

Best Paper

Extraction and neural synthesis of low-order spectral components for Head-Related Transfer Functions

Mary Pilataki, Chris Buchanan and Cal Armstrong

Best Student Paper

Automatic Audio Equalization with Semantic Embeddings

Eloi Moliner, Vesa Välimäki, Konstantinos Drosos and Matti Hämäläinen

Keynote Speakers

Keynote 1: Commercial AI in 2025 – an Investment Perspective

Speaker: Hazel Savage

Abstract:

How the latest AI technology is disrupting the creative industries and creativity itself

The world of music has always been defined by disruption. From vinyl to CDs, from downloads to streaming, every leap in technology has rewritten the rules of how music is created, distributed, and consumed. Now, the industry faces its most profound shift yet: artificial intelligence.

For some, AI is a saviour – offering new ways to manage the tidal wave of content and unlock creativity. For others, it is a destroyer – threatening artistry, originality, and livelihoods. As with past transformations, the truth likely lies somewhere in between. But what is certain is that AI is already reshaping the music business at every level: discovery, production, distribution, and monetisation.

Keynote 2: Towards a fairer approach to generative AI training

Speaker: Ed Newton-Rex

Abstract:

Ed will discuss the issues that arise when generative AI companies scrape training data without consent, and the alternative – licensing training data – that is being embraced by many AI music companies.

Keynote 3: Present-Day Trends in AI for Music, Music Production and Audio Technology

Speaker: Jonathan Wyner

Abstract:

We'll hear perspectives resulting from research examining policies and approaches to integrating AI in curriculum development and industry partnerships within music, music production, and audio technology education programs, based on interviews with program directors and institutional leaders, and offering a broad perspective on current trends.

As AI moves into educational programs they face a crossroads: embrace its potential or guard against its pitfalls. This talk maps the current terrain of threats and opportunities in music and technology training.

Keynote 4: Ethical MIR Research for Trustworthy AI: Principles, Challenges, and Practices

Speaker: Xavier Serra

Abstract:

As AI systems increasingly shape how music is created, discovered, and experienced, the need for ethical, transparent, and culturally sensitive research in Music Information Retrieval (MIR) has never been more urgent. In this talk, I will argue for the importance of rethinking MIR research in the context of trustworthy AI, grounding the discussion in a set of ethical principles—respect for creative integrity, cultural sensitivity, user agency, and fairness.

Building on these foundations, I will explore key research directions that emerge from such a commitment, including the development of culturally diverse datasets, tools for attribution and content provenance, mechanisms for bias detection and mitigation, and the design of explainable, open, and user-controllable AI systems. I will emphasize the need for AI to support—not replace—human creativity, advocating for collaborative platforms that foster meaningful interaction between humans and machines.

Throughout the talk, I will illustrate these ideas with examples from ongoing projects at the Music Technology Group (MTG) of Universitat Pompeu Fabra, showcasing how our work strives to balance technological innovation with ethical responsibility, cultural awareness, and openness. The aim is to outline a roadmap for a more inclusive, responsible, and impactful future for AI and MIR research.

Papers

Audio Engineering Society

Conference Paper 1

Presented at the AES International Conference on Artificial Intelligence and Machine Learning for Audio
2025 September 8-10, London, UK

This paper was peer-reviewed as a complete manuscript for presentation at this conference. This paper is available in the AES E-Library (http://www.aes.org/e-lib) all rights reserved. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Journal of the Audio Engineering Society.

Motor2Synth: Leveraging Differentiable Digital Signal Processing for Generating Combustion Engine Sounds Compatible with Active Sound Design Frameworks

Thiago Henrique Gomes Lobato[1], Stefan Hank[1], Zhang Hanyi[2], Luo Haofu[2]

[1]HEAD acoustics GmbH
[2]Xiaomi

Correspondence should be addressed to Thiago Lobato (Thiago.lobato@head-acoustics.com)

ABSTRACT

In Active Sound Design (ASD) applications for electric vehicles, many manufacturers want to have at least some sounds that emulate aspects of combustion engine noise, while being different enough to show their own identity. Consequently, combustion engine sounds frequently serve as the foundation for creative sound design. Traditional artificial sound generation methods, however, struggle to replicate the distinct acoustic properties and dynamics of combustion engines, leading to unrealistic results. Although real vehicle recordings are available, their chaotic nature and significant RPM fluctuations make them challenging to process for sample-based synthesis. This paper introduces a novel approach to model arbitrary combustion engine sounds using artificial intelligence, specifically through Differentiable Digital Signal Processing (DDSP). DDSP models a sound or process with differentiable implementations of typical signal processing modules such as harmonic generator, noise filter and reverberation. The model can then generate sounds based on an estimation of a fundamental frequency and signal loudness, effectively as a synthesizer, being the reason, we call our tool Motor2Synth. For motor sounds, however, the fundamental frequency should be the rotational speed (RPM), and pre-trained models for music such as CREPE fail to provide good results. To address this, we developed a self-supervised pitch estimator based on the PESTO model to estimate rotational speed, integrating it into a DDSP framework to generate sounds. We show that our model can correctly estimate the rpm of combustion engine noise while also producing realistic outputs that can be used as base for ASD applications. Finally, we demonstrate a designed sound using a DDSP version of a high-performance engine as a component to create a sound variant for the car Xiaomi SU7 and Xiaomi SU7 Ultra.

1 Introduction

In recent years, the percentage of electric vehicles being produced and purchased has sharply increased [1]. These vehicles have a different, less prominent sound signature than combustion engines, which allows engineers to shape the vehicles' identity with their own sound compositions through Active Sound Design (ASD). The types of sounds can vary from very futuristic to classical sporty tones, as long as they fulfil the user's expectations. These expectations often rely on familiar sounds; thus, classical combustion engine sounds serve as a basis for many designs. To generate ASD sounds, a combination of synthesizers and digital signal processing (DSP) effects can be used. This approach works especially well when developing futuristic sounds but is limited when emulating combustion engine sounds due to their unique characteristics and variations as a

function of the car's rotational speed. These characteristics are particularly difficult to emulate and demand a high amount of manual fine-tuning for each individual case, which is very expensive or impractical when modelling many different motors. An alternative would be to use recorded motor samples, but commercially available options mostly contain recordings with constant acceleration, which are not suitable for the often-used sample-based synthesis which is applied in electric cars to process and playback the sound interactively.

Recently, the use of Differentiable Digital Signal Processing (DDSP) has proven effective in tasks such as instrument modification [2], piano synthesis [3], automatic effect transfer [4], wavetable synthesis [5], sound matching of a synthesizer [6], modelling of animal sounds [7] and many others [8]. DDSP works by modelling classical DSP approaches as differentiable elements in an automatic differentiation framework such as PyTorch [9]. For instrument modelling, a combination of harmonic generation and filtered noise is commonly used [2], possibly with additional support effects like reverb. This allows one to learn any instrument as a function of its loudness and pitch using only a small number of recordings. Loudness is usually calculated as the A-weighted sound pressure level, and pitch is estimated with neural models such as CREPE [10].

Unlike conventional engine-sound pipelines, DDSP can learn the motor's acoustic behaviour end-to-end. This is achieved since its harmonic oscillators and noise generators are implemented as trainable DSP block so that a DDSP model 1) automatically captures narrow-band resonances of the block, exhaust and chassis without manual tuning, 2) separates and parametrises broadband noise so that intake hiss or valve clatter can be shaped independently, and 3) extrapolates to unseen RPM trajectories simply by driving the latent pitch and loudness controls beyond the training set [2].

By contrast, procedural / physical-model synths must represent each acoustic cavity with explicit filter coefficients, and thus matching the coupled resonances of real hardware quickly becomes labour-intensive. Additionally, the hand-tuned model can break when the target engine geometry or load profile changes [11]. Sample-based and granular replay systems avoid that design effort but tie fidelity to the coverage of the original recordings: wide-range pitch-shifting smears formants, and the fixed loops make it hard to decouple tonal orders from stochastic components, limiting both realism and design space [12].

This work explores the use of a DDSP framework to model motor sounds as "instruments", so they can be used as synthesizers in classical digital audio workstations (DAWs). We demonstrate that with a DDSP approach, it is possible to fully transform a motor into a synthesizer with satisfactory accuracy for creative applications. We start by presenting the dataset and methodology used, then we show our results and the application of our approach to create a component for a real car ASD, the Xiaomi SU7/SU7 Ultra. A full representation of our method can be seen in Figure 1.

2 Data

To train and evaluate the model, we used sample packs of combustion engine noise with the total of 9 vehicles, namely:

- 1966 Chevelle SS Big Block
- 1966 Chevelle SS Small Block
- 1966 Mustang
- 1968 Galaxie
- 1969 Satellite
- 1976 Regal
- 1997 Express
- 2007 Durango
- 2007 GT500

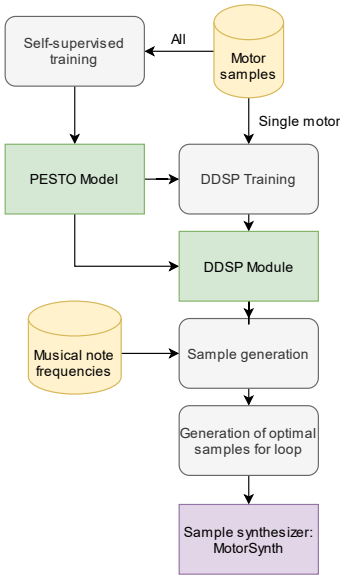


Figure 1. Schematic of all steps necessary to get from motor samples to a musical synthesizer in ASD applications using our Motor2Synth approach.

This list encompasses motors considered "vintage" as well as relative modern combustion engine ones. The

samples were bought from the boom Library [13] and contained diverse measurements for different conditions. We pre-processed the samples removing the silence, so that only the parts in which the motor was running were considered for the training. The total length of "useful" audio for each motor was around 15 minutes, which is a relatively small amount and was enough to generate good results. The spectrogram of some of the generated signals can be seen in Figure 2.

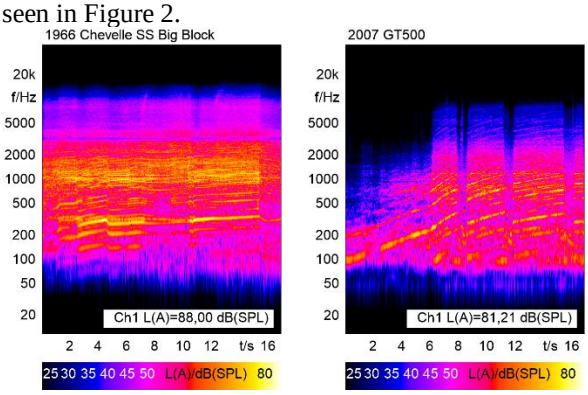


Figure 2. Motor training samples given to the DDSP model.

We see that each motor has a very different harmonic distribution and different resonances as function of the RPM, which explains the challenge of properly modelling them.

3 DDSP model

For the DDSP model we use the Pytorch DDSP framework [14] to train a harmonic plus noise model with a differentiable reverberation module. The standard noise implementation of the library was a concatenation of filtered noise, which we modified to perform an overlap-and-add synthesis. The actual sound model of a single block s_b is simply defined:

$$s_b = A \cdot \sum_{i=1}^{N_h} \mathrm{a_i} \sin(2\pi f_0 \cdot i \cdot t) + h(t) * n(t) \quad (1)$$

For the harmonic part, N_h is the number of harmonics (in our case 100, which was necessary to model modern motors), A the global harmonic amplitude, $\mathrm{a_i}$ the strength of the partials between 0 and 1 and f_0 is the fundamental frequency (in our case the rotational speed). For the noise part, $h(t)$ is an impulse response defined with a FIR filter with 75 samples and $n(t)$ is sampled from a Gaussian noise with mean zero and variance 1. Blocks are then combined in an overlap-and-add fashion with 75% overlap and using a Hanning window. Afterwards, the resulting signal is processed with a learnable reverberation module defined as a noise with exponential decay. The data sampling rate was set to 16000 Hz (since motor harmonics rarely go above 8000 Hz) with 160 samples block size.

The standard pitch estimation of the used Pytorch DDSP framework [14] is CREPE [10], which was trained mainly for musical instruments and voice. In the results section we will show that the version of CREPE trained with musical instruments is not suitable to identify RPM of motor signals, and thus it is not possible to create a reasonable motor model with it. To train a new CREPE, labelled RPM data would be needed, which is expensive to get, especially for sound-sample data. As an alternative, we trained an RPM estimator using the PESTO [15] model. The main idea of the model is that it is easier to learn relative pitch differences than absolute ones. Thus, if we can apply a known pitch shift to a signal and have a model that has equivariant outputs to this pitch difference, we just need a single calibration afterwards to have a good estimator. Additionally, we can add noise to our data for which the shift is invariant (this can be either white noise or any disturbance that should not affect the shift) and train the model to identify those. This can then be learned in a fully unsupervised fashion; thus, we do not need labels. The RPM estimator was trained using the data of all vehicles. To consider that each vehicle may have resonances at different harmonics/orders, thus possibly confusing the RPM estimators, a post-processing of the estimated rotational speed with a scalar factor to maximize the number of orders with high energy can be done. This factor can also be used to consider the high Subharmonic-to-Harmonic Ratio (SHR) of motor sounds by selecting an "artificial" RPM equal to half the actual RPM. In this way, the DDSP model is also able to handle half-orders, which are very prominent in motor sounds.

At last, to ensure the pitch is mostly continuous between time steps, we modelled the time-pitch transition as a hidden-Markov model that favours continuous estimations, so that sharp transitions on the RPM were penalized (this sampler was also used for CREPE estimator but without success). Additionally, we used a moving average smoothing to compensate the discrete resolution of PESTO.

To evaluate the estimation, we estimated the ground-truth rpms of 100 samples from different cars with the RPM generator tool from the software ArtemiS. As a fully signal-processing baseline, we also added the YIN [16] pitch estimator to the evaluation. To compensate possible boundary effects

due to the PESTO pitch smoothing, we removed the boundaries not fully in the average window from all comparisons. Additionally, since we used "artificial" rpms to handle sub-harmonics, we evaluate the estimation up to a factor that is calculated for each case individually.

4 Training scheme

To learn the actual DDSP model, we trained each vehicle individually. The training was done with the AdamW optimizer with 1e-3 learning rate with cosine schedule to 1e-4. 80 % of the data was used for the training and 20% for the validation. The loss was a multi-scale STFT loss defined as follows:

$$L = \frac{1}{M} \cdot \sum_{i=1}^{N} \left\| S_{i,x} - S_{i,y} \right\|^1 + \left\| \log\left(S_{i,x}\right) - \log\left(S_{i,y}\right) \right\|^1 \quad (2)$$

in which $S_{i,x}$ is a spectrogram with a block size defined by i and M is the number of samples. We used scales with length {128,256,512,1024,2048,4096} containing 75% overlap in time. This loss allows one to evaluate the representation in multiple time-frequency scales.

We noticed that for all vehicles the loss could reach validation values of about 2.X. However, by training the model that much we ended up with DDSP models that 1) extrapolate badly above the frequency/loudness range of the sample data (which was not so large) and 2) used too much tonal components to compensate the noise, generating very "metallic" sounds. By searching other stop criteria, we found out that interrupting training at the epoch in which the validation loss crossed a threshold below 5.0 was enough to produce realistic generations while retaining reasonable extrapolations above and below the training range. This value coincides with the recommendation found in the Magenta DDSP training notebook example found in their code repository [2].

5 Results: RPM estimation

In Figure 3 we show results of the absolute frequency deviation of the different frequency estimators for 100 manually labelled samples. We see that CREPE has an average error of 17.3 % and some very strong outliers. This result is very similar to the YIN method, which shows that CREPE has an accuracy equivalent to a typical signal-processing approach under the domain-shift from musical instruments to motor sounds. The PESTO method, on the other hand, shows an error of only 0.8%, which is an order of magnitude lower and, in musical terms, is less than half a semi-tone (or around 18 cents). This indicates that for RPM estimations, both YIN and CREPE trained mainly for musical instruments are not as appropriated as a tailored PESTO model, which is not particularly surprising, since they both represent very different type of sounds and the motor sounds have a considerably higher noise level.

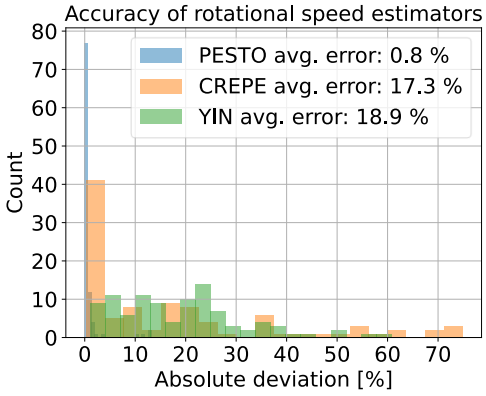


Figure 3. Accuracy of frequency estimation in the 100 manually labelled examples for different estimators.

To grasp the meaning of this difference in accuracy for the pitch estimation, we show in Figure 4 and 5 below some estimations in real signals done both with PESTO and CREPE, in which it becomes clear that the pre-trained CREPE model is not able to consistently estimate a valid rotational speed for motor sounds, as it shows sharp discontinuities in signals with continuous changes. It is however very likely that by training CREPE with motor data, the result will improve. However, getting the training data is a very expensive endeavour that we found to be unnecessary due to the high accuracy of PESTO.

6 Results: sound reproduction

Spectrograms of some results can be seen in the examples that follow in the Figures 6 and 7 for different vehicles, all with SFTF-multi scale errors of around 5.0.

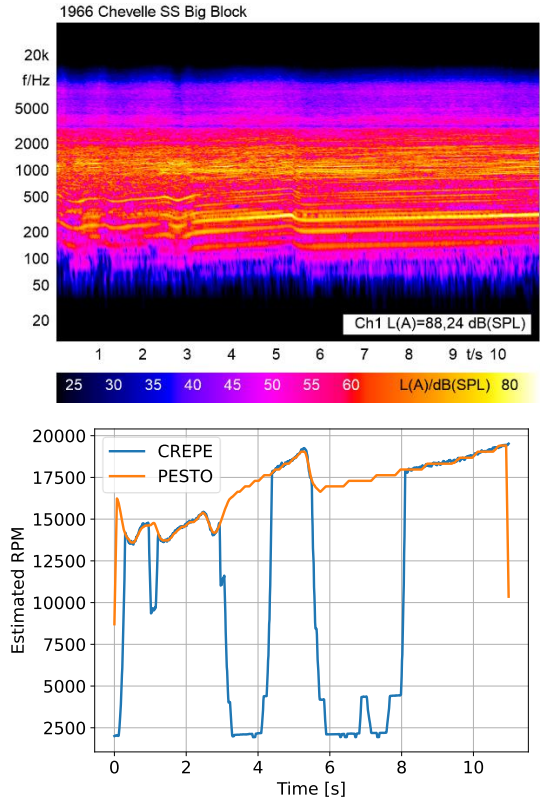


Figure 4. Estimated rotational speed with CREPE and PESTO of a 1966 Chevelle sample.

All sounds are still easily distinguishable in an informal A/B comparison. However, to our main sound designer, the fidelity was judged enough for creative applications, especially since the idea is to use those sounds as basis or layers in a sound composition. Thus, we have defined a process in which we can approximately represent a set of combustion engine motor recordings in a full parametric fashion, with respect to a loudness and pitch distribution, that is good enough for a sound designer to use as basis for new sounds.

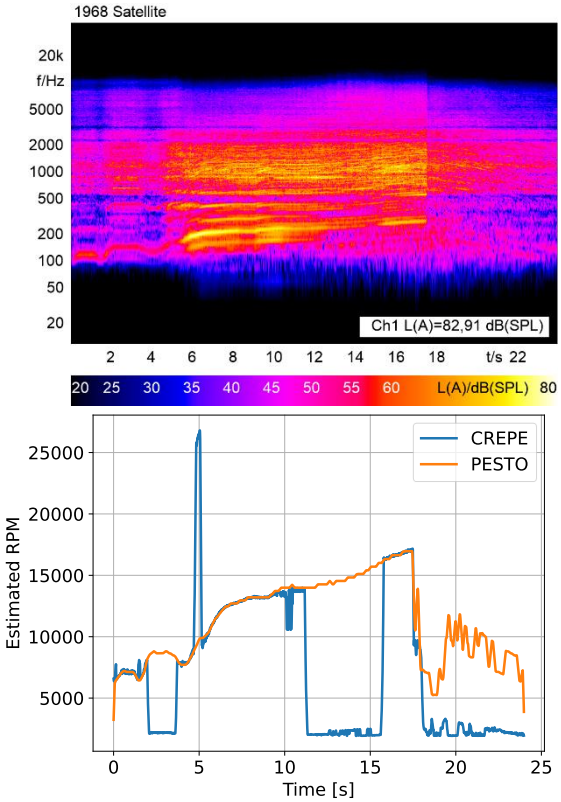


Figure 5. Estimated rotational speed with CREPE and trained PESTO of a 1968 Satellite sample.

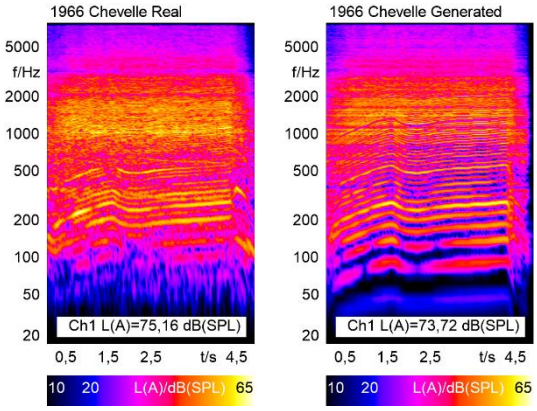


Figure 6. Example of the DDSP reconstruction of a 1966 Chevelle Big Block.

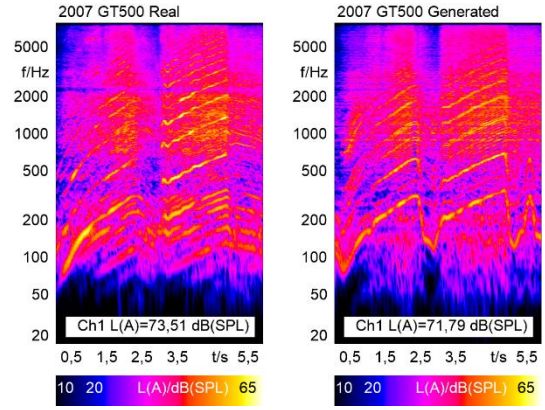


Figure 7. Example of the DDSP reconstruction of 2007 GT500.

A good example of this parametric capability can be seen in Figure 8 below in which we perform a "motor transfer", namely transforming an input measurement of a vintage 1966 Chevelle to a more modern 2007 GT500. This is done by estimating the loudness and pitch from the Chevelle and use those as inputs to the GT500 model.

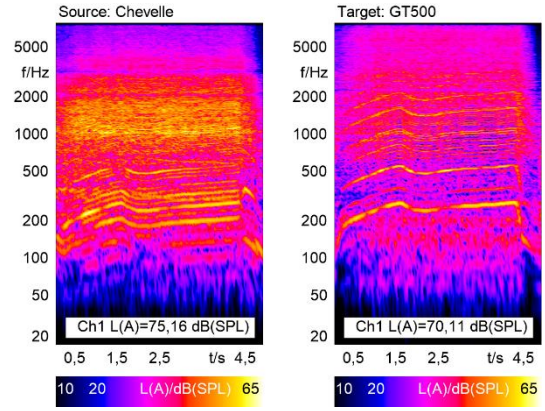


Figure 8. Example of a "motor transfer", in which left we have the input 1966 Chevelle motor and right the target 2007 GT500.

Besides the motor transfer approach, one possible creative application is motor interpolation. This could be done either by 1) assigning a latent variable to each motor and then interpolating between both or by 2) interpolating the filter and harmonic parameters given by two different models. Although we don't investigate such an approach in this paper, we see it as a very interesting creative application for future research.

7 Data pre-processing for ASD

In ASD applications for electric vehicles, sounds need to follow a given interface and be connected to the main inputs of the car such as the user pedal input. Algorithms also need to be efficient and low energy. We achieve this by using the software QNX® Sound from Blackberry, which allow us to obtain drivers' inputs and use those to generate our sounds, which we do based on samples. Those samples are generated by sampling the different motors at given musical pitches and different levels of loudness. This can be seen, in fact, as a downgrade from the actual DDSP model, however, it is necessary step to a practical implementation. Some advantages of the DDSP model, thought, such as range extrapolation and separation of tonal and noise components can still be leveraged.

For the samples to be usable, we need to place them on a loop, so that they comply with the application of possibly indefinitely generation during the drive. Thus, it is important that the transition between end and start of the loop cannot be perceived by the driver, which could disturb their immersion and be perceived as annoying.

For motor samples, a simple cross-fade is not enough to produce good loops, since they possess a strong modulation that is audible if the sound loops at the wrong position. Additionally, due to the noise, the perfect cut point does not exactly match the signal period. To solve this problem, we searched the position that best matched the initial window of the signal. This approach makes sure that there's no strong discontinuity in the signal, both in respect to individual samples as well as to longer modulations.

An example of this process can be seen in Figure 9, in which a window is defined, matched and the signal is looped. One can see that both the scale of the window (long modulation) as well as in the scale of individual samples the resulting signal looks healthy and, indeed, does not produce any hearable artefact.

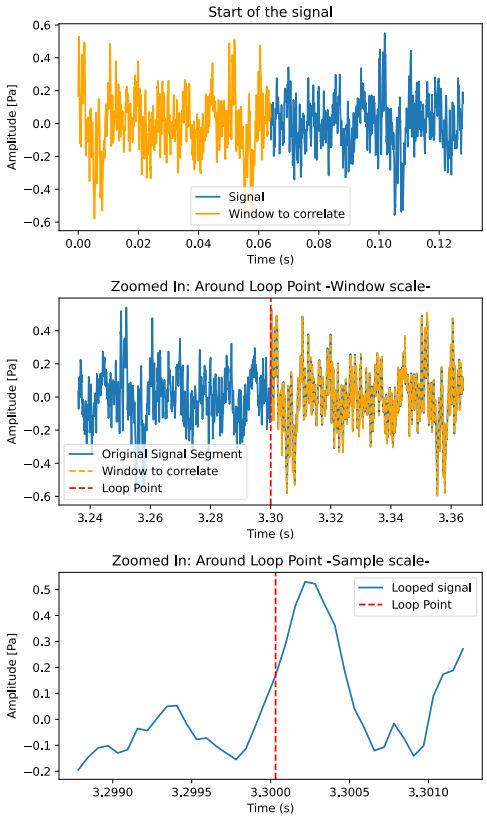


Figure 9. Process to estimate ideal cut position for the signal. An initial window is defined and then the best correlation in the signal is sought. A cut point is then selected as the one that produces the smoothest loop transition.

8 Creation of a real ASD sound

The Xiaomi SU7 is an electric full-size sedan with a unique and sporty visual design. The challenge of the active sound design process was to create a sound that matches all different power versions of the Xiaomi SU7 and creates a unique and distinctive sound, that fits to the visual design and supports the driving experience.

In total, the vehicle has three different active sound designs, which are all created with very different techniques. The purpose of the sound design utilizing the Motor2Synth Engine was to reinterpret the sound and emotion of a V8 engine.

For the sound design, a car with a V8 motor engine was selected as basis and a DDSP model was created in the exact same matter as the other cars so far. An example of a real and reconstructed sample is shown in Figure 10 (with a multi-scale STFT loss of ca. 4.9).

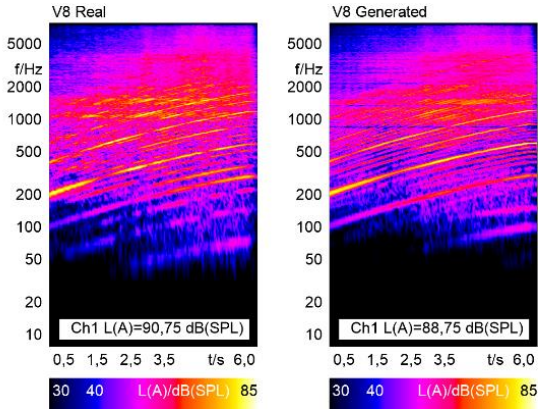


Figure 10. Example of the real and generate V8 motor used as basis for the Xiaomi SU7.

For the dynamic playback of the active sound design, a sample-based and an additive synthesis approach were used. This allows a good tradeoff between creative flexibility and efficient processing of the sound during driving. Since the car does not shift gears and the sound also does not have a virtual gear shifting, one challenge was to avoid a too high pitch shift of the used sound samples. By using a Shepard tone-like structure, the samples could cover the entire speed range by only shifting two octaves in pitch. The samples for the ASD sound were derived from the Motor2Synth engine. By utilizing the ability of the Motor2Synth engine to render sounds with different pitches, single layers for the low, mid and high frequency range could be rendered and filtered to reconstruct a unique V8 like sound.

With additional processing, using frequency filter, saturation, compression and modulation-based effects a futuristic character could be added to the V8 sound. After applying the effects, the sound samples had to be rendered and recut to result in an audio sample that can be looped perfectly. However, since the Motor2Synth engine already provides quasi periodic samples, this process was easier. In Figure 11 a spectrogram of the final sound can be seen.

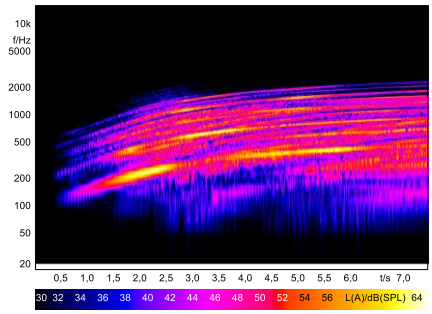


Figure 11. The created V8 sound design without wind-, road and powertrain noise.

9 Conclusion

We've shown that it is possible to use a DDSP approach with a differentiable harmonic generator plus noise module to learn "motor synthesizers" based only on a couple of minutes of training data. Those models can then be used to generate samples for a sample-based synthesizer and, even though they are still perceptually different from the original one, are deemed realistic enough for creative applications by a sound designer.

In this work, we also demonstrated that the main fundamental frequency estimators used for instrument modelling are not appropriate for motor sounds, likely due to the sound characteristics of combustion engines and musical instruments being considerable different. This was mitigated by training a model specific for motor sounds using the self-supervised model PESTO. This model was validated both by the quality of the generated sounds as well as by utilizing artificial generation, in which the ground truth of the rotational speed was known. This test showed that the CREPE model and YIN algorithm had a similar average estimation error of, respectively, ~17% and ~18%, while PESTO had an error of only ~1%, one order of magnitude lower, showing that classical pitch estimators used in music are not optimal for estimating the RPM. Figures 4 and 5 of this paper show what this magnitude difference means in the results.

At last, we show that our approach is ready and relevant for real applications, since it is already in use on a real electric vehicle, the Xiaomi SU7. A future work could be done to model different motors with the same network backbone but using different embeddings for each vehicle, so that designers could sample the latent space and easily interpolate between very different motors. An interesting creative application of this could be to allow drivers themselves to manipulate such a latent space based on pre-made designs to obtain the "perfect" individual sound.

References

[1] International Energy Agency, Global EV Outlook 2023. Paris, France: IEA, 2023. [Online]. Available: https://www.iea.org/reports/global-ev-outlook-2023 (accessed 30 Jun 2025)

[2] J. Engel, L. Hantrakul, C. Gu, and A. Roberts, "DDSP: Differentiable digital signal processing," in Proc. Int. Conf. Learning Representations (ICLR), Virtual/ Addis Ababa, Ethiopia, 2020.

[3] L. Renault, R. Mignot, and A. Roebel, "Differentiable piano model for MIDI-to-audio performance synthesis," in Proc. 25th Int. Conf. Digital Audio Effects (DAFx20in22), Vienna, Austria, 6–10 Sep 2022.

[4] C. J. Steinmetz, N. J. Bryan, and J. D. Reiss, "Style transfer of audio effects with differentiable signal processing," arXiv preprint arXiv:2207.08759, 2022.

[5] S. Shan, L. Hantrakul, J. Chen, M. Avent, and D. Trevelyan, "Differentiable wavetable synthesis," in Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP), Singapore, 22–27 May 2022, pp. 4598–4602.

[6] N. Masuda and D. Saito, "Synthesizer sound matching with differentiable DSP," in Proc. 22nd Int. Soc. Music Information Retrieval Conf. (ISMIR), Online, 7–12 Nov 2021, pp. 428–434.

[7] M. Hagiwara, M. Cusimano, and J.-Y. Liu, "Modeling animal vocalizations through synthesizers," arXiv preprint arXiv:2210.10857, 2022.

[8] B. Hayes, J. Shier, G. Fazekas, A. McPherson, and C. Saitis, "A review of differentiable digital signal processing for music and speech synthesis," Frontiers in Signal Processing, vol. 3, 2024, Art. 1284100, doi: 10.3389/frsip.2023.1284100.

[9] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in Advances in Neural Information Processing Systems 32 (NeurIPS 2019), Vancouver, Canada, 2019, pp. 8024–8035.

[10] J. W. Kim, J. Salamon, P. Li, and J. P. Bello, "CREPE: A convolutional representation for pitch estimation," in Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP), Calgary, Canada, 15–20 Apr 2018, pp. 161–165.

[11] S. Baldan, H. Lachambre, S. Delle Monache, and P. Boussard, "Physically informed car engine sound synthesis for virtual and augmented environments," in Proc. IEEE 2nd VR Workshop on Sonic Interactions for

Virtual Environments (SIVE), Arles, France, 24 Mar 2015, pp. 1–6, doi: 10.1109/SIVE.2015.7361287.

[12] J. Jagla, J. Maillard, and N. Martin, "Sample-based engine noise synthesis using an enhanced pitch-synchronous overlap-and-add method," J. Acoust. Soc. Am., vol. 132, no. 5, pp. 3098–3108, Nov. 2012, doi: 10.1121/1.4754663

[13] Boom Library, Cars – V8 (sound-effects collection). [Online]. Available: https://www.boomlibrary.com/sound-effects/.

[14] L. Renault, ddsp_pytorch (Version 1.0) [Source code]. GitHub, 2022. Available: https://github.com/acids-ircam/ddsp_pytorch (accessed 30 Jun 2025)

[15] A. Riou, S. Lattner, G. Hadjeres, and G. Peeters, "PESTO: Pitch estimation with self-supervised transposition-equivariant objective," in Proc. 24th Int. Soc. Music Information Retrieval Conf. (ISMIR), Milan, Italy, 5–9 Nov 2023, pp. 535–544.

[16] A. de Cheveigné and H. Kawahara, "YIN, a fundamental frequency estimator for speech and music," J. Acoust. Soc. Am., vol. 111, no. 4, pp. 1917–1930, Apr. 2002, doi: 10.1121/1.1458024.

Audio Engineering Society

Conference Paper 2

Presented at the AES International Conference on Artificial Intelligence and Machine Learning for Audio 2025
September 8–10, London, UK

This paper was peer-reviewed as a complete manuscript for presentation at this conference. This paper is available in the AES E-Library (http://www.aes.org/e-lib), all rights reserved. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Journal of the Audio Engineering Society.

Sound Matching an Analogue Levelling Amplifier Using the Newton-Raphson Method

Chin-Yun Yu[1] and György Fazekas[1]

[1] *Centre for Digital Music, Queen Mary University of London, London, UK*

Correspondence should be addressed to Chin-Yun Yu (chin-yun.yu@qmul.ac.uk)

ABSTRACT

Automatic differentiation through digital signal processing algorithms for virtual analogue modelling has recently gained popularity. These algorithms are typically more computationally efficient than black-box neural networks that rely on dense matrix multiplications. Due to their differentiable nature, they can be integrated with neural networks and jointly trained using gradient descent algorithms, resulting in more efficient systems. Furthermore, signal processing algorithms have significantly fewer parameters than neural networks, allowing the application of the Newton-Raphson method. This method offers faster and more robust convergence than gradient descent at the cost of quadratic storage. This paper presents a method to emulate analogue levelling amplifiers using a feed-forward digital compressor with parameters optimised via the Newton-Raphson method. We demonstrate that a digital compressor can successfully approximate the behaviour of our target unit, the Teletronix LA-2A. Different strategies for computing the Hessian matrix are benchmarked. We leverage parallel algorithms for recursive filters to achieve efficient training on modern GPUs. The resulting model is made into a VST plugin and is open-sourced.

1 Introduction

Controlling the dynamic range of an audio signal is a fundamental task in audio engineering. Audio effects such as compressors, limiters, and expanders are designed to achieve this goal and are often used in music production, broadcasting, and live sound reinforcement. Analogue compressors developed in the early days have iconic sounds, such as the popular Teletronix LA-2A, which is known for its smooth and musical compression. Considerable interest has been devoted to replicating their characteristics in the digital world, known as virtual analogue (VA) modelling [1].

1.1 Problem Definition

Let us denote an audio stimulus as $x(t)$ and its discrete version as $\mathbf{x} = [x(0), x(T), \ldots, x((N-1)T)]^\top \in \mathbb{R}^N$ with $T \in \mathbb{R}_+$ being the sampling period. We want to model a target analogue compressor $\delta : \mathbb{R}^\infty \to \mathbb{R}$, specifically, its behaviour to the stimulus $x(t)$ as $y(t) = \delta(x(s) \mid s \leq t)$. We denote $\mathbf{y} = [y(0), y(T), \ldots, y((N-1)T)]^\top \in \mathbb{R}^N$ as the sampled compressed signal. Given a controllable base model $f : \mathbb{R}^N \times \mathbb{R}^M \to \mathbb{R}^N$ parametrised by M parameters, we aim to find the optimal parameters $\boldsymbol{\theta}^* \in \mathbb{R}^M$ that minimises the distance between the output $\hat{\mathbf{y}} = f(\mathbf{x}, \boldsymbol{\theta})$ and $\mathbf{y}$ according to a distance function $\mathscr{D} : \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}_+$.

1.2 Related Work

In recent years, neural networks (NNs) with various architectures have become popular base models for f. Hawley et al. [2] are pioneers in this approach, using an U-Net-like autoencoder with skip connections to model the LA-2A end-to-end. Steinmetz and Reiss [3] proposed temporal convolutional networks (TCNs) to enhance efficiency while maintaining large receptive fields of the networks. Different conditioning strategies have been explored to improve the modelling ability [4, 5]. More recent works explored the use of state-space models for VA modelling [6, 7]. The choice of architecture is crucial since compressors are used in real-time applications, and the computational cost is a concern. Despite impressive results, the black-box nature of NNs makes the learnt parameters difficult to interpret and limits creative control.

Among previous works, Wright and Välimäki [8] proposed a grey-box approach that combines NNs and signal processing components. A feed-forward compressor is used as f with a lightweight Gated Recurrent Unit (GRU) at the end to model time-varying non-linear make-up gain. This approach achieves competitive results while requiring only 5% of arithmetic operations compared to a single GRU. The learnt control parameters also give us a better understanding of the target unit's behaviour. Yu et al. [9] further improve their training speed by implementing a specialised kernel for backpropagating the gradients through the ballistics module of the compressor.

1.3 Motivation

Both [8] and [9] use gradient descent optimisation to find the optimal feed-forward compressor parameters for VA modelling. Gradient descent is widely used in training neural networks instead of more robust methods like Newton-Raphson (NR) due to its simplicity and lower storage cost, favouring methods with many parameters ($M \gg 10^6$). However, since the feed-forward compressor has only a handful of parameters ($M < 10$), the quadratic storage cost of NR is feasible on modern computers. NR converges faster and more robustly than gradient descent when the objective function is convex around the solution and twice differentiable. This motivates us to explore the NR method for VA modelling of analogue compressors. Although NR has been used to solve differential equations in white-box models, such as wave digital filters [10] and the physical modelling of musical instruments [11], to the authors' knowledge, it has not been used for grey-box modelling of analogue compressors.

1.4 Contributions

In this article, we formulate VA modelling as a sound matching problem, aiming to find the optimal parameters of a digital compressor f with the closest match to the target sound. We verify the feasibility of using the NR method for modelling the LA-2A compressor. Moreover, we further accelerate the backpropagation method from [9] on modern GPUs using parallel algorithms for recursive filters. The learnt mapping from the circuit's peak reduction to the parameters is presented, providing an interpretable and intuitive way to control the compressor. We made the resulting model into a VST plugin, open-sourced under the MPL-2.0 license[1], for validation and as a viable music production tool.

2 Methodology

Continuing the work from [8, 9], we use a feed-forward compressor as the base model f for its efficiency and high modelling capability. We will discuss the choice of distance function in Section 2.2. Since the input signal $\mathbf{x}$ is fixed during optimisation, we use $f_{\mathbf{x}} : \mathbb{R}^M \to \mathbb{R}^N$ to denote the curried f given $\mathbf{x}$, to simplify the later derivative notations.

2.1 Feed-Forward Compressor (f)

We adopt `torchcomp` [9], a differentiable implementation of the feed-forward compressor from [12]. A digital compressor is defined as:

$$\hat{y}[n] = x[n]g[n] = x[n]g\left(s[m], \boldsymbol{\theta} \mid m \leq n\right), \qquad (1)$$

where $g : \mathbb{R}^{n+1} \times \mathbb{R} \to \mathbb{R}_+$ is the gain reduction function with parameters $\boldsymbol{\theta}$ and $s[m]$ is the side-chain signal. The design is feed-forward when $s[m] = x[m]$ [13]. We remove the root mean square (RMS) level detector from the design for two reasons. Firstly, Table 7 in [9] shows the converged RMS smoothing coefficient is close to one, implying the smoothing effect is negligible. Secondly, we found that when running the NR method, the converged $\mathscr{D}(\hat{\mathbf{y}}, \mathbf{y})$ is slightly lower without it. A possible explanation is that the extra degree of freedom

[1] github.com/aim-qmul/4a2a

introduced by the RMS filter creates a local minimum that traps the NR optimisation, a phenomenon we will discuss in Section 2.2.

The resulting compressor consists of five parameters ($M = 5$): threshold (CT), ratio (R), attack time (t_{at}), release time (t_{rt}), and make-up gain (γ). Threshold and make-up gain are expressed in decibels, and the attack/release times are in milliseconds. Threshold controls the level at which the compressor starts to work, and the ratio determines the amount of gain reduction. The make-up gain adjusts the output level to compensate for the gain reduction. The attack and release time control the ballistics of the gain reduction, which is defined as:

$$g[n] = \begin{cases} \alpha_{at}\hat{g}[n] + (1-\alpha_{at})g[n-1] & \text{if } \hat{g}[n] < g[n-1], \\ \alpha_{rt}\hat{g}[n] + (1-\alpha_{rt})g[n-1] & \text{otherwise,} \end{cases} \tag{2}$$

$$\alpha_{at/rt} = 1 - \exp\left(-\frac{2200T}{t_{at/rt}}\right). \tag{3}$$

$\hat{g}[n]$ is the gain reduction before applying the ballistics. Equation (2) is a recursive smoothing filter, and the attack and release times determine how fast the filter responds to the loudness change. The whole signal flow is shown in Fig. 1.

Both threshold and make-up gain can be any real numbers, but not for the other parameters. We also want to bound the parameters to some reasonable range that is common in practice. To enforce these constraints, we define the parameter vector as $\theta = \{CT, \gamma, \hat{R}, \hat{\alpha}_{at}, \hat{\alpha}_{rt}\} \in \mathbb{R}^5$ and get

$$\{R, \alpha_{at}, \alpha_{rt}\} = \Big\{ \lambda(u) \mid u \in \{\hat{R}, \hat{\alpha}_{at}, \hat{\alpha}_{rt}\}, \\ \lambda(u) = \theta_{min}(u) + \big(\theta_{max}(u) - \theta_{min}(u)\big)\sigma(u) \Big\}, \tag{4}$$

where $\sigma : \mathbb{R} \to [0,1]$ is the monotonic sigmoid function, and $\theta_{min} : \mathbb{R} \to \mathbb{R}$ and $\theta_{max} : \mathbb{R} \to \mathbb{R}$ return the lower and upper bounds of the parameters, respectively.

2.2 Newton-Raphson Optimisation

Newton-Raphson is a fundamental optimisation method that finds the minimum of a smooth and convex function $\mathscr{L} : \mathbb{R}^M \to \mathbb{R}$. It updates the parameter θ by the following equation:

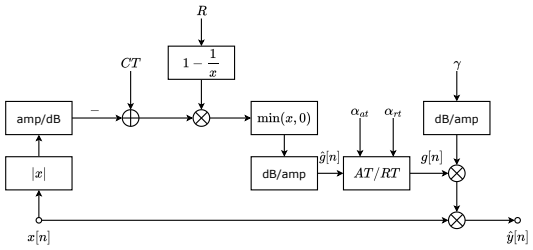


Fig. 1: Block diagram of the feed-forward compressor. "AT/RT" refer to the ballistics control in Eq. (2).

$$\theta \mapsto \theta - [\nabla^2 \mathscr{L}(\theta)]^{-1} \nabla \mathscr{L}(\theta). \tag{5}$$

Here $\mathscr{L}(\theta) = \mathscr{D}(f_{\mathbf{x}}(\theta), \mathbf{y})$, and $\nabla\mathscr{L}(\theta) \in \mathbb{R}^M$ and $\nabla^2\mathscr{L}(\theta) \in \mathbb{R}^{M\times M}$ are the gradient and Hessian matrix of $\mathscr{L}$ at θ, respectively. NR leverages second-order information from the Hessian; thus, it guarantees convergence to a local minimum and converges faster than gradient descent, especially when the initial point is close to the minimum.

NR has two requirements that need to be satisfied:

1. $\mathscr{L}$ must be convex around the optimal θ^*.
2. $\mathscr{L}$ must be twice differentiable.

We validate 1) empirically by observing the convergence behaviour, which we will discuss more in Section 4.4. For 2), we set the distance function to be the square distance $\mathscr{D}(f_{\mathbf{x}}(\theta), \mathbf{y}) = (f_{\mathbf{x}}(\theta) - \mathbf{y})^\top (f_{\mathbf{x}}(\theta) - \mathbf{y})$, which is twice differentiable. Regarding the differentiability of the feed-forward compressor f, although `torchcomp` computes the sub-gradients for operations that are not differentiable everywhere (the absolute and minimum functions in Fig. 1 and the if-else statement in Eq. (2)), it is shown that they are accurate enough for first-order optimisation. Since its gradient backpropagation function is also differentiable, we can quickly evaluate its second-order derivatives within automatic differentiation frameworks like PyTorch [14].

3 Hessian Computation

`torchcomp` implements the Vector-Jacobian product (VJP) [15, 16] $\mathbf{v} \mapsto \mathbf{v}^\top \nabla f_{\mathbf{x}}(\theta)$ of the compressor, which lets us compute the VJP of the loss function $\mathbf{v} \mapsto \mathbf{v}^\top \nabla \mathscr{L}(\theta)$ using backpropagation where

$\nabla\mathscr{L}(\theta) = \nabla_{f_\mathbf{x}(\theta)}\mathscr{D}(f_\mathbf{x}(\theta),\mathbf{y})\nabla_\theta f_\mathbf{x}(\theta)$. We get the gradient by setting $\mathbf{v}=1$.

To get the Hessian matrix $\nabla^2\mathscr{L}(\theta)$, note that the Hessian matrix is the Jacobian of $\nabla\mathscr{L}(\theta)$. Given an identity matrix $\mathbf{I}\in\mathbb{R}^{M\times M}$, if we pass the i^{th} column of $\mathbf{I}$ into the VJP of $\nabla\mathscr{L}:\mathbb{R}^M\mapsto\mathbb{R}^M$, which is $\mathbf{v}\mapsto\mathbf{v}^\top\nabla^2\mathscr{L}(\theta)$, the result will be the i^{th} row of the Hessian $\nabla^2\mathscr{L}(\theta)$; computing the entire Hessian equals passing every column of $\mathbf{I}$ to the VJP function in parallel ($\mathbf{I}^\top\nabla^2\mathscr{L}(\theta)$).

Although we can get the Hessian using VJP twice (which is known as *reverse-over-reverse* mode) as we show above, the second (sometimes also called outer) Jacobian can also be computed with Jacobian-Vector product (JVP) $\mathbf{v}\mapsto\nabla h(\theta)\mathbf{v}$ [17]. This is called *forward-over-reverse* mode where $h=\nabla\mathscr{L}$. Since the Jacobian of $\nabla h(\theta)$ is a square matrix, computing it by passing the basis vectors, either from the right or the left, has to be repeated the same number of times. The efficiency thus depends on the computational costs of VJP and JVP of $h:\mathbb{R}^M\to\mathbb{R}^M$, although in practice, the *forward-over-reverse* strategy is usually preferred. For a fair comparison, we derive the JVP of the function $\mathbf{v}\mapsto\mathbf{v}^\top\nabla f_\mathbf{x}(\theta)$, specifically, the backpropagation of the ballistics module in Eq. (2), as it involves recursive filters and is non-trivial to implement in PyTorch.

3.1 Forward-Mode Gradients for Time-Varying One-Pole Filters

To get the gradients of Eq. (2), let us rewrite it as a time-varying one-pole filter:

$$\beta[n] = 1-\alpha_{at}^{\zeta[n]}\alpha_{rt}^{1-\zeta[n]}, \tag{6}$$

$$\tilde{g}[n] = (1-\beta[n])\hat{g}[n], \tag{7}$$

$$\underbrace{g[n]}_{\text{output}} = \underbrace{\tilde{g}[n]}_{\text{input}} + \underbrace{\beta[n]}_{\text{multiplier}}\ \underbrace{g[n-1]}_{\text{previous output}}, \tag{8}$$

where $\zeta[n]\in\{0,1\}$ is the attack and release phase indicator function. We colour-coded Eq. (8) to emphasise the recursions as it will show up multiple times in the derivations, and we have special treatment for it in Section 4.2.

According to [9], backpropagating $\nabla_{g[n]}\mathscr{L}(\theta)$ through Eq. (8) can be expressed as passing through a time-varying one-pole filter in reversed time:

$$\nabla_{\tilde{g}[n]}\mathscr{L}(\theta) = \nabla_{g[n]}\mathscr{L}(\theta) + \beta[n+1]\nabla_{\tilde{g}[n+1]}\mathscr{L}(\theta). \tag{9}$$

The gradients $\nabla_{\beta[n]}\mathscr{L}(\theta)$ can be computed using chain rule:

$$\nabla_{\beta[n]}\mathscr{L}(\theta) = \nabla_{\tilde{g}[n]}\mathscr{L}(\theta)(g[n-1]-\hat{g}[n]). \tag{10}$$

The gradients $\nabla_{\alpha_{at}}\mathscr{L}(\theta)$, $\nabla_{\alpha_{rt}}\mathscr{L}(\theta)$, and $\nabla_{\hat{g}[n]}\mathscr{L}(\theta)$ are trivial to compute given Eq. (6) and Eq. (7).

Backpropagation of $\nabla_{\tilde{g}[n]}\nabla\mathscr{L}(\theta)$ (we simplify $\nabla_{\nabla_*\mathscr{L}(\theta)}$ as ∇_* for brevity) through Eq. (9) can be done in the same way as in Eq. (8):

$$\nabla_{g[n]}\nabla\mathscr{L}(\theta) = \nabla_{\tilde{g}[n]}\nabla\mathscr{L}(\theta) + \beta[n]\nabla_{g[n-1]}\nabla\mathscr{L}(\theta), \tag{11}$$

$$\nabla_{\beta[n+1]}\nabla\mathscr{L}(\theta) = \nabla_{g[n]}\nabla\mathscr{L}(\theta)\nabla_{\tilde{g}[n+1]}\mathscr{L}(\theta). \tag{12}$$

For forward propagation (JVP) on Eq. (9), let us use Eq. (8) as an example. Since in the forward mode the Jacobians are multiplied from right to left (opposite to the backpropagation), we need to apply Eq. (9) and Eq. (10) in reverse order and traverse the computational graph from the outputs back to the inputs. This also means applying recursive filters in the opposite direction. Given forwarded gradients $\nabla_\theta\tilde{g}[n]$ and $\nabla_\theta\beta[n]$, the forward version of Eq. (9) is:

$$\nabla_\theta g[n] = [\nabla_\theta\tilde{g}[n] + \nabla_\theta\beta[n]\,(g[n-1]-\hat{g}[n])] + \beta[n]\nabla_\theta g[n-1], \tag{13}$$

which is just changing the input of Eq. (8) to $\nabla_\theta\tilde{g}[n] + \nabla_\theta\beta[n](g[n-1]-\hat{g}[n])$. Applying the same process to Eq. (9), we can get the forward version of Eq. (11) similar to Eq. (13) as:

$$\nabla_\theta\nabla_{\tilde{g}[n]}\mathscr{L}(\theta) = [\nabla_\theta\nabla_{g[n]}\mathscr{L}(\theta) + \nabla_\theta\beta[n]\nabla_{\tilde{g}[n+1]}\mathscr{L}(\theta)] + \beta[n+1]\nabla_\theta\nabla_{\tilde{g}[n+1]}\mathscr{L}(\theta). \tag{14}$$

Comparing the VJP of Eq. (9) (Eq. (11) and Eq. (12) combined) to its JVP (Eq. (14)), we see that the JVP has one more addition than the VJP. Nevertheless, the extra addition does not occur within the recursion, and modern computers are very good at vectorised atomic addition, so the computation time is often negligible.

In summary, in this section, we show that propagating the gradients of a time-varying one-pole filter in forward mode equals filtering the gradients with the same filter[2]. The resulting Eq. (13) is required for the JVP of the inner Jacobian $\mathbf{v}\mapsto\nabla\mathscr{L}(\theta)\mathbf{v}$, while Eq. (14) is required for the JVP of the outer Jacobian $\mathbf{v}\mapsto\nabla h(\theta)\mathbf{v}$.

[2]This result generalises to higher order all-pole filters as well.

4 Experiment

4.1 Dataset

We use the recordings from SignalTrain [18] as it is the largest curated analogue compressor dataset known to the authors and has been utilised in many previous works [3, 8, 9, 5]. It contains paired recordings sampled at 44.1 kHz from LA-2A. Technically, the LA-2A is a levelling amplifier (LA) and does not have the parameters mentioned in Section 2.1. It has only one knob, called *peak reduction*, that controls the compression. The peak reduction was sampled from 0 to 100 with a spacing of 5, and the gain value was fixed for the compressor and limiter modes. We select recordings labelled with `3c` in the file name, which use an identical 20 min audio stimulus consisting of both musical and synthetic sounds for all configurations.

4.2 Acceleration Techniques

We implement everything in PyTorch. To accelerate the training process, we split the audio into 12 s chunks with 1 s overlap, so $f_{\mathbf{x}}$, which is inherently sequential in contrast to $\nabla f_{\mathbf{x}}$, can be parallelised at the batch level. The overlap parts are used as a warm-up, and the loss is computed only on each chunk's last 11 s. Following [8], we apply a pre-emphasis filter $\frac{1-z^{-1}}{1-0.995z^{-1}}$ to $\mathbf{y}$ and $\hat{\mathbf{y}}$ prior to computing the loss to reduce low-frequency noise.

Furthermore, we identify that a lot of the operations are one-pole filters (Eq. (8)), and a parallel algorithm exists for these filters called *parallel associative scan* [19]. The idea is to convert the recursion into a form like $s[0] \oplus s[1] \oplus \ldots \oplus s[n]$ where $\oplus$ is associative, then the scan algorithm can be applied. We use the CUDA implementation from [20] to parallelise the coloured equations (except Eq. (8)) and the pre-emphasis filter on the GPU.

4.3 Analysis on Computational Cost

PyTorch has two implementations for computing the Hessian. One lies under the `autograd` module, which is the default implementation and has been around since PyTorch 1.5. It only has full coverage for *reverse-over-reverse* (hereafter referred to as *rev-rev*) mode. The other implementation is the `func` module, introduced in PyTorch 2.0, which is relatively new but allows for the flexible composition of forward and reverse mode Jacobians. We evaluate all the four combination strategies for the Hessian $\nabla^2 \mathscr{L}(\boldsymbol{\theta})$ using `func` and compare them to the default `autograd` implementation.

Table 1: The memory cost (in MB) and runtime (in milliseconds) of different hessian computation strategies in PyTorch.

	`autograd`	`func`			
	rev-rev	rev-rev	fwd-rev	rev-fwd	fwd-fwd
Memory	**1066**	2358	3534	6306	5364
Time	**26.5**	28.1	28.1	64.7	38.9

We report the computational cost in Table 1, measured on an RTX 3060 with a batch size of 16. From Table 1, we see that the *rev-rev* and *fwd-rev* modes are equally fast, which implies that the extra addition we mentioned in Section 3 does not impact the performance. The *rev-fwd* mode is the slowest and most memory-consuming. We see that *fwd-rev* requires more memory than *rev-rev*, probably due to non-optimal memory utilisation for the forward gradients in our implementation[3]. The table also shows inherent overhead in memory and time for the `func` module, as the numbers are all bigger than those of `autograd`. This is likely because this feature is still in the Beta stage and has not been optimised enough.

Based on the above findings, we choose the `autograd` implementation. All the computations for each configuration can be fitted on a single RTX 3060 with 12GB of VRAM, and no mini-batching is required. The resulting optimisation takes roughly just 3.5 s for one NR update. Moreover, training on the entire dataset takes less than 20 min, instead of hours as reported in [8].

4.4 Optimisation Strategy

We perform the damped NR method with backtracking line search [21]. In each step, the parameters are updated as $\boldsymbol{\theta} \mapsto \boldsymbol{\theta} - \tau \boldsymbol{\nu}$, where $\tau \in [0,1]$ is the step size and $\boldsymbol{\nu}$ is the solution of $\nabla^2 \mathscr{L}(\boldsymbol{\theta})\boldsymbol{\nu} = \nabla \mathscr{L}(\boldsymbol{\theta})$. We use an equation solver to avoid performing matrix inversion for better numerical stability. Starting from $\tau = 1$, we divide τ by 2 until the Armijo–Goldstein

[3] In our filter implementation, only one dimension of the tensor is treated as batch dimension. The straightforward way to call the JVP M times in parallel using the same function is to duplicate the parameters M times along the batch dimension.

condition [22] $\mathscr{L}(\theta - \tau v) \leq \mathscr{L}(\theta) - \alpha\tau\nabla\mathscr{L}(\theta)^\top v$ is satisfied with $\alpha = 0.0001$. If the Hessian matrix is not positive semi-definite (which implies negative curvature), we randomly sample a new direction orthogonal to v and repeat the process.

We find different optimal parameters for each sampled peak reduction. Since the NR method is sensitive to the initialisation, we start from 100 peak reduction which has the heaviest compression so it is easier to fit, and then use the previous result to initialise the subsequent peak reduction, e.g., $100 \rightarrow 95 \rightarrow 90$, etc. This ensures the starting point for each configuration is as close to the solution as possible. The initial values are determined empirically based on early experiments and shared beliefs of the target circuit, which are $CT = -36$ dB, $\gamma = 0$ dB, $R = 4$, $t_{at} = 1$ ms, and $t_{rt} = 200$ ms. We restrict the range of the parameters to $R \in [1, 20]$, $t_{at} \in [0.1, 100]$ ms, and $t_{rt} \in [10, 1000]$ ms.

The optimisation converges in less than 10 iterations in most cases. We stopped the process at 40 peak reduction, as the NR method could not find a consistent solution for lower peak reductions. The optimisation encounters negative curvature structure frequently [4]. We investigated the data and found that the compression is less noticeable in this range and is surpassed by other non-linearities that a feed-forward compressor cannot capture. For other peak reductions, the NR method converges to the same solution in multiple different initialisations, verifying that $\mathscr{L}(\theta)$ is mostly convex around the optimal solution.

4.5 Results

> *...The average compression ratio is always set at roughly 3:1, while the average Attack time is 10 milliseconds, and the Release time is about 60 milliseconds for 50% of the release, and anywhere from 1 to 15 seconds for the rest.* [5]

The learnt configurations are shown in Fig. 2. The mappings to the parameters are nearly identical between the compressor and limiter modes, except for the ratio, which is slightly higher in the limiter mode but exhibits a similar trend. The ratio is typically around 4:1 in compressor mode, slightly deviating from the manufacturer's stated ratio of 3:1. The attack and release times vary exponentially with the peak reduction, rather than fixed values. Since there is only one release time parameter, it does not capture the documented two-stage release behaviour of the LA-2A, but an average of them.

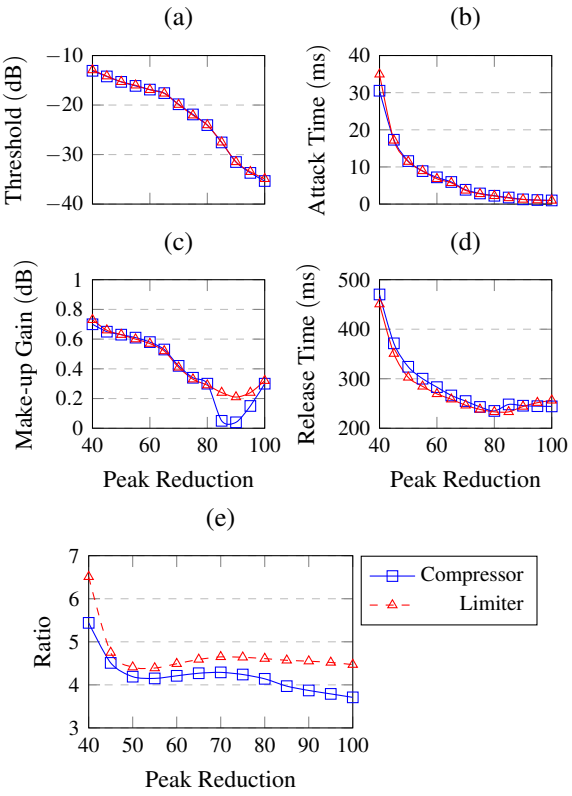


Fig. 2: The mapping from LA-2A peak reduction to the compressor's parameters.

5 Evaluation

We compare our compressor (denoted as 4A-2A and "4" stands for feed-forward) with the following baselines: the official UAD LA-2A plugin [6], the emulation plugin CA-2A by Cakewalk [7], the emulation plugin CLA-2A by Waves [8], and the grey-box compressor from [8]. We evaluate 4A-2A using its training data, as the five-parameter model cannot fully represent the LA-2A and no overfitting is observed on this dataset

[4] For peak reductions over 40, this rarely happens (it happens a few times when random initialisation is used).

[5] www.uaudio.com/blog/la-2a-collection-tips-tricks

[6] www.uaudio.com/uad-plugins/compressors-limiters/teletronix-la-2a-tube-compressor

[7] legacy.cakewalk.com/Products/CA-2A

[8] www.waves.com/plugins/cla-2a-compressor-limiter

with more complex models [8]. We evaluate performance using the Error-to-Signal Ratio (ESR) and the Loudness Dynamic Range difference (ΔLDR). ESR is defined as $\frac{(\mathbf{y}-\hat{\mathbf{y}})^\top(\mathbf{y}-\hat{\mathbf{y}})}{\mathbf{y}^\top\mathbf{y}}$. ΔLDR is the difference between the LDR [23] of the target signal and the predicted signal. The same pre-emphasis filter is applied to all audio before evaluations.

To calculate LDR given $\mathbf{y}$, we first compute the log ratio between the short-term and long-term RMS envelopes as

$$L_{\mathbf{y}}[n] = 10\log_{10}\frac{\mathrm{RMS}_{\mathrm{short}}(y[n])}{\mathrm{RMS}_{\mathrm{long}}(y[n])}. \tag{15}$$

The integration time for the short-term RMS is set to 50 ms and the long-term RMS is set to 3 s where the details can be found in [23]. We define $\mathrm{LDR}(\mathbf{y})$ as

$$\mathrm{LDR}(\mathbf{y}) = \sqrt{\frac{1}{N}\sum_{n=0}^{N-1} L_{\mathbf{y}}[n]^2}, \tag{16}$$

which tells how much the loudness varies locally (microdynamics) in $\mathbf{y}$. ΔLDR is simply $\mathrm{LDR}(\hat{\mathbf{y}}) - \mathrm{LDR}(\mathbf{y})$ so if $\Delta\mathrm{LDR} > 0$, it means $\hat{\mathbf{y}}$ is less compressed than $\mathbf{y}$ and vice versa.

Since we do not know what setup and gain staging were used to model the commercial plugins, the plugins' input and output gains must be tuned. We manually adjust the input gain so that the sum of its ΔLDR across peak reductions is close to zero, ensuring its compression curve matches the SignalTrain dataset. We then adjust the output gain so that its overall ESR is minimised. We render the audio through plugins using `pedalboard` [24].

We choose the Model 5 configuration for the grey-box compressor, their best-performing model. It is architecture-wise very similar to our 4A-2A, but the make-up gain γ is replaced with a GRU. Thus, we follow their setup to train a GRU on the same training set but pre-processed by 4A-2A with $\gamma = 0$ dB. We set the batch size to 26 (13 configurations × two modes) and the sequence length to 12288. The training converges after around eight epochs. We stop the training at the 15$^{\text{th}}$ epoch and pick the checkpoint with the lowest loss. The process takes less than 30 minutes on an RTX 3060 Laptop GPU. We denote this model as 4A-2A-G.

The evaluation results are shown in Fig. 3. 4A-2A-G has the lowest ESR and is close to those reported in [8]. 4A-2A shows comparable performance to the official UAD plugin. At around 75 peak reduction, the LA-2A behaves most like the feed-forward compressor, thus having the lowest ESR. Beyond this point, the 4A-2A's ESR increases, and the errors are more for the limiter mode. For the commercial plugins, their curves are less smooth and have a few apparent spikes. Among them, the CA-2A has the highest ESR in general and even reaches over 40 % at 100 peak reduction.

Regarding their ΔLDR profiles, we see that CA-2A varies the most with a maximum mismatch of 1 dB at 100 peak reduction. UAD and CLA-2A exhibit similar patterns of difference, which can be divided into three stages: from 40 to 60 (peak reduction), their compressions are lighter than the ground truth; then, the compressions become heavier from 60 to 80 and then return to lighter levels beyond 80. 4A-2A and 4A-2A-G have their ΔLDR very close to zero thus out-performing others, where 4A-2A-G being the closest.

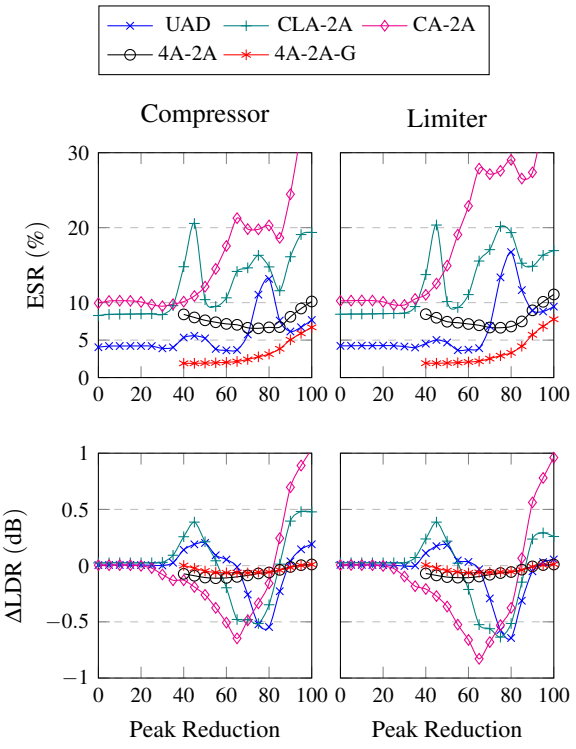


Fig. 3: ESR and ΔLDR of all the evaluated models under different peak reduction and mode.

6 Application: the 4A-2A Plugin

We implement our 4A-2A compressor as a real-time audio plugin using JUCE [25], providing it as a cre-

ative tool for audio engineers. In the plugin interface (Fig. 4), we have five sliders that control the parameters of the underlying compressor. At the bottom, there is one slider for peak reduction and a switch for the mode. The peak reduction slider is linked to the above compressor's parameters, mapping the peak reduction to them using the learnt mappings from Fig. 2. The mode switch toggles between the compressor and limiter profiles. The user can adjust the peak reduction and observe the compressor's behaviour in real-time. If they want to fine-tune the compressor, they can adjust the parameters manually.

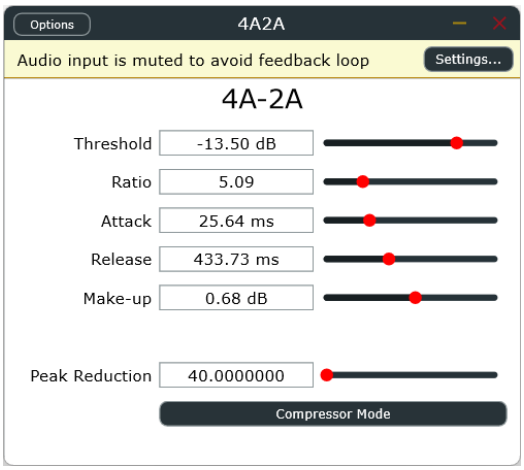


Fig. 4: The GUI of the 4A-2A plugin.

The peak reductions in SignalTrain are sampled on a grid. Interpolation is required for continuous mapping $[40,100] \rightarrow \mathbb{R}^5$. To evaluate the interpolation error, we exclude peak reductions $\{45,55,\ldots,95\}$ and use the remaining 7 points to interpolate them using either linear or cubic spline interpolation. Their ESR is shown in Fig. 5. Linear interpolation and spline interpolation perform the worst at 65 and 95 peak reduction, respectively. On average, the spline interpolation has a slightly higher ESR than the linear interpolation. We suppose this finding generalises to denser grids and choose the linear interpolation for the plugin.

We made the GRU model from 4A-2A-G into a standalone plugin. The GRU model is very lightweight, consisting of only one layer with eight hidden units. We wrapped the GRU model as a PyTorch script module using Neutone [26], which can be loaded into their Neutone FX plugin [9]. Cascading the 4A-2A and the

[9] github.com/Neutone/neutone_sdk

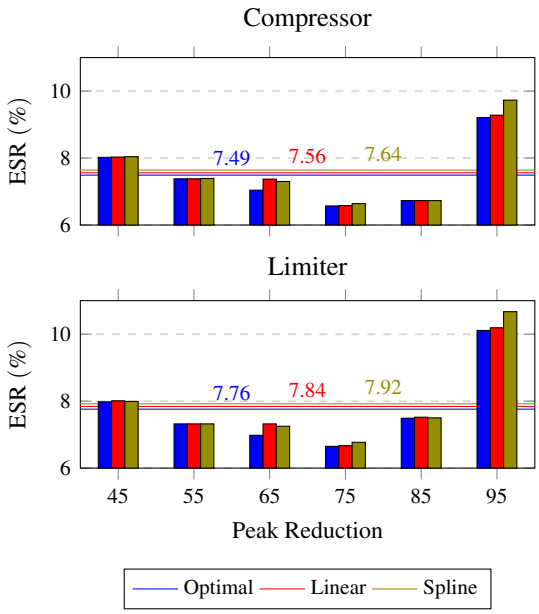


Fig. 5: Comparison of different interpolation methods. The horizontal lines are the averages.

GRU model resembles the 4A-2A-G baseline, a real-time implementation of the Model 5 emulator from [8], with the feed-forward part being trained using the NR method, outperforming all commercial plugins in our evaluation.

Comparing the residuals with or without the GRU make-up gain shown in Fig. 6, the most noticeable difference is in the high-frequency range (> 4 kHz). The GRU successfully captures the non-linearities of these analogue circuits, which cannot be modelled by a simple digital compressor, and spreads out the errors across frequencies.

7 Discussion

The NR method effectively finds a unique solution to match the LA-2A's response to the square distance criterion. There is room for improvement, as LA-2A is technically a feedback compressor; therefore, the signal flows of 4A-2A and LA-2A are not precisely matched. By using a digital feedback compressor ($s[n] = \hat{y}[n-1]$) and modelling the two-stage release behaviour explicitly, we can potentially improve the performance. However, efficiently backpropagating gradients through the feedback path is challenging and requires custom implementations. In contrast, our grey-box sound-matching

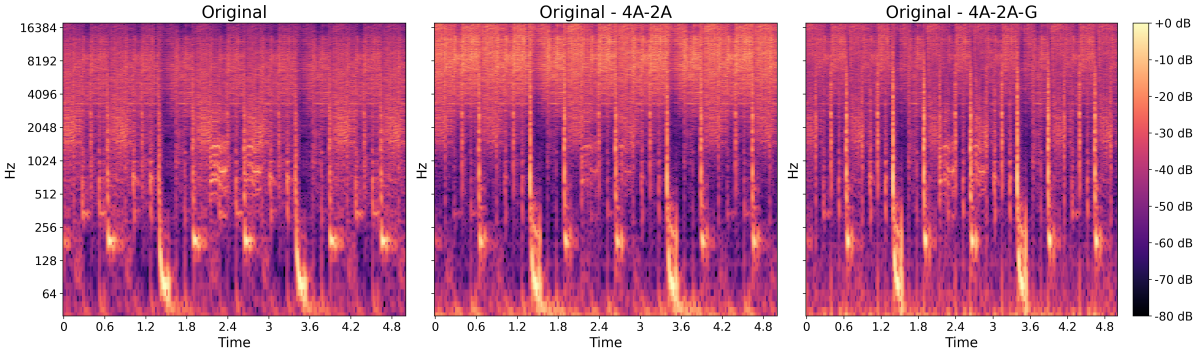


Fig. 6: The spectrograms of a drum loop from the SignalTrain and the residuals ($\mathbf{y} - \hat{\mathbf{y}}$) of the model's predictions with 75 peak reductions operated in compressor mode.

approach can be generalised to any dynamic range controller circuit, and our trial on the LA-2A already yields satisfactory results.

Due to the variations between analogue circuits, the inferior plugin performance we see in Section 5 is expected. Their modelled LA-2A might behave slightly differently from the SignalTrain one. It is interesting to note that the UAD and CLA-2A have similar ESR/∇LDR patterns, which could be due to similar target circuits or modelling approaches.

We evaluate the entire Hessian matrix, as the number of parameters is smaller. However, for more complex effects and longer effect chains, the Hessian can become too large to fit in memory. Methods like conjugate gradients using Hessian-vector product $\mathbf{v} \mapsto \nabla^2 \mathscr{L}(\theta)\mathbf{v}$ can be used to solve the equation $\nabla^2 \mathscr{L}(\theta)\nu = \nabla \mathscr{L}(\theta)$ without explicitly computing the Hessian [17, 27]. Nevertheless, increasing the number of parameters also raises the possibility of encountering more local minima, making it harder to find the optimal solution.

8 Conclusion

This paper presented a method to emulate the LA-2A optical compressor using a feed-forward digital compressor with parameters optimised via the Newton-Raphson method. Our approach successfully turned the LA-2A's response into interpretable parameters that can be used for creative control. The efficient implementation of ballistics and the parallel associative scan algorithm for gradient computation significantly accelerate optimisation. Forward-mode gradient propagation through the compressor is derived to benchmark different strategies for computing the Hessian.

Our findings suggest that the Newton-Raphson method is a promising approach for the grey-box modelling of audio compressors, providing a foundation for further research and development in this area. Further improvements could be made by using a feedback design and explicitly modelling the LA-2A's two-stage release behaviour. Future work could extend Newton-Raphson optimisation to other analogue compressors or non-linear audio effects. Another direction could be reducing the computational costs of the Hessian matrix for more complex signal chains.

Acknowledgements

The first author is a research student supported jointly by UKRI (grant number EP/S022694/1) and Queen Mary University of London.

References

[1] Eichas, F., Gerat, E., and Zölzer, U., "Virtual analog modeling of dynamic range compression systems," in *AES 142nd Convention*, 2017.

[2] Hawley, S. H., Colburn, B., and Mimilakis, S. I., "Profiling Audio Compressors with Deep Neural Networks," in *AES 147th Convention*, 2019.

[3] Steinmetz, C. J. and Reiss, J. D., "Efficient neural networks for real-time modeling of analog dynamic range compression," in *AES 152nd Convention*, 2022.

[4] Comunità, M., Steinmetz, C. J., Phan, H., and Reiss, J. D., "Modelling black-box audio effects with time-varying feature modulation," in *Proc. IEEE ICASSP*, pp. 1–5, 2023.

[5] Yeh, Y.-T., Hsiao, W.-Y., and Yang, Y.-H., "Hyper Recurrent Neural Network: Condition Mechanisms for Black-box Audio Effect Modeling," in *Proc. DAFx*, pp. 97–104, 2024.

[6] Simionato, R. and Fasciani, S., "Comparative Study of Recurrent Neural Networks for Virtual Analog Audio Effects Modeling," *arXiv preprint arXiv:2405.04124*, 2024.

[7] Simionato, R. and Fasciani, S., "Modeling Time-Variant Responses of Optical Compressors with Selective State Space Models," *Journal of the Audio Engineering Society*, 73, pp. 144–165, 2025.

[8] Wright, A. and Välimäki, V., "Grey-box modelling of dynamic range compression," in *Proc. DAFx*, pp. 304–311, 2022.

[9] Yu, C.-Y., Mitcheltree, C., Carson, A., Bilbao, S., Reiss, J. D., and Fazekas, G., "Differentiable All-pole Filters for Time-varying Audio Systems," in *Proc. DAFx*, pp. 345–352, 2024.

[10] Bernardini, A., Bozzo, E., Fontana, F., and Sarti, A., "A Wave Digital Newton-Raphson Method for Virtual Analog Modeling of Audio Circuits with Multiple One-Port Nonlinearities," *IEEE/ACM TASLP*, 29, pp. 2162–2173, 2021.

[11] Bilbao, S., Torin, A., and Chatziioannou, V., "Numerical Modeling of Collisions in Musical Instruments," *Acta Acustica united with Acustica*, 101(1), pp. 155–173, 2015.

[12] Zölzer, U., *DAFX: Digital Audio Effects*, chapter Nonlinear Processing, pp. 110–112, John Wiley & Sons, 2011, ISBN 0470665998.

[13] Giannoulis, D., Massberg, M., and Reiss, J. D., "Digital dynamic range compressor design—A tutorial and analysis," *Journal of the Audio Engineering Society*, 60(6), pp. 399–408, 2012.

[14] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. NeurIPS*, pp. 8026 – 8037, 2019.

[15] Rirsch, M. W. and Smale, S., *Differential equations, dynamical systems, and linear algebra*, ACADEMIC PRESS. INC., 1974.

[16] Robeyns, M., "Of VJPs and JVPs," `https://maximerobeyns.com/of_vjps_and_jvps`, 2023, [Accessed: 2025-02-17].

[17] Dagréou, M., Ablin, P., Vaiter, S., and Moreau, T., "How to compute Hessian-vector products?" in *ICLR Blogposts*, 2024, `https://iclr-blogposts.github.io/2024/blog/bench-hvp/`.

[18] Colburn, B. and Hawley, S., "SignalTrain LA2A Dataset," 2020, doi:10.5281/zenodo.3824876.

[19] Blelloch, G. E., "Prefix Sums and Their Applications," Technical Report CMU-CS-90-190, School of Computer Science, Carnegie Mellon University, 1990.

[20] Martin, E. and Cundy, C., "Parallelizing Linear Recurrent Neural Nets Over Sequence Length," in *ICLR*, 2018.

[21] Moré, J. J. and Sorensen, D. C., "Newton's method," Technical report, Argonne National Lab.(ANL), Argonne, IL (United States), 1982.

[22] Armijo, L., "Minimization of functions having Lipschitz continuous first partial derivatives," *Pacific Journal of Mathematics*, 16(1), pp. 1 – 3, 1966.

[23] Nercessian, S., McClellan, R., and Lukin, A., "A direct microdynamics adjusting processor with matching paradigm and differentiable implementation," in *Proc. DAFx*, 2022.

[24] Sobot, P., "Pedalboard," 2021, doi:10.5281/zenodo.7817838.

[25] Raw Material Software Limited, "JUCE: Jules' Utility Class Extensions," `https://juce.com/`, 2025, [Accessed: 2025-02-16].

[26] Fyfe, A. and Mitcheltree, C., "Neutone-Real-time AI audio plugin for DAWs," in *Audio Developer Conference*, 2022.

[27] Hestenes, M. R. and Stiefel, E., "Methods of conjugate gradients for solving linear systems," *Journal of research of the National Bureau of Standards*, 49, pp. 409–435, 1952.

Audio Engineering Society

Conference Paper 3

Presented at the AES International Conference on Machine Learning and Artificial Intelligence for Audio 2025 September 8–10, London, UK

This paper was peer-reviewed as a complete manuscript for presentation at this conference. This paper is available in the AES E-Library (http://www.aes.org/e-lib), all rights reserved. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Journal of the Audio Engineering Society.

Compression of Scene-Based Higher Order Ambisonics with Multichannel RVQGAN

Toni Hirvonen and Mahmoud Namazi

Samsung Research America

Correspondence should be addressed to Toni Hirvonen (t.hirvonen@samsung.com)

ABSTRACT

A multichannel extension to the RVQGAN neural coding method is proposed, and realized for data-driven compression of third-order Ambisonics audio. The input- and output layers of the generator and discriminator models are modified to accept multichannel input. We also propose a loss function which accounts for spatial perception in immersive reproduction, and transfer learning from single-channel models. Listening test results with 7.1.4 immersive playback show that the proposed extension is suitable for coding ambient scene-based, third-order (16-channel) Ambisonics content with good quality at 16 kbps when trained and tested on the EigenScape database. The model presented is the first neural codec dedicated to immersive audio to the authors' knowledge and has potential applications for learning other types of content and multichannel formats.

1 Introduction

Audio compression is an important aspect of storage and transport, in particular with respect to new immersive formats. The size and dimensionality of this type of content motivates discovering new, efficient coding methods. This is especially true in the case of Higher Order Ambisonics, a prominent surround sound encoding method [1] which has in recent times been a popular solution for immersive capture and reproduction.

At the same time, the rise of data-driven processing methods and available computational power make it feasible to explore methods beyond conventional audio coding. However, neural coding of multichannel- and immersive audio has received relatively little attention. This is partly due to lack of material available to train and test such models.

This paper presents a simple, yet general and efficient, multichannel extension to the popular RVQGAN audio coding methods suitable for the compression of Ambisonic audio. Despite being anchored in audio coding, i.e. faithfully reproducing an original reference, we also draw inspiration from music generation. The training and tests conducted with scene-based material illustrate the feasibility of the proposed method, and are aimed to inspire further applications in the future.

Our contributions include: The model presented in this paper is the first neural codec dedicated to immersive audio to the authors' knowledge. In addition to the modified model architecture, we also propose customized training methods. As a novel testing aspect for neural codecs, multichannel loudspeaker listening is used to evaluate if the method in terms of spatial quality and compression efficiency.

2 Background

2.1 Neural Audio Coding and Music Generation

In this section, we draw commonalities between neural coding and music generation, and survey how both fields relate to the present study.

RVQGAN is a popular model for end-to-end neural audio coding that includes a data-compressing encoder and an expanding decoder [2, 3]. Diffusion models [4, 5] as well as transformer-based alternatives [6] have also been proposed. All neural codecs, despite having very good quality at particular low-bitrate range, have issues that impede real-life use. Computational complexity, especially on the decoder side, is a predicament for real-life deployment on many devices. End-to-end neural methods can also have problems scaling towards complete transparency, and hybrid systems aim to address this by using generative neural techniques to enhance the quality of a traditional codec [7].

The majority of neural audio coding research has been focused on single-channel aka mono audio compression. However, most modern audio content has more than one channel, making relevant the spatial and immersive aspects of audio. In recent decades, formats and reproduction systems utilizing more than two channels have emerged, starting from 5.1 surround and culminating with the immersive audio formats of recent years [8, 9, 10, 11, 12].

Some attempt to apply neural coding methods to multichannel audio coding amount to processing a stereo input as dual-mono, thereby doubling the bitrate [13]. These works do not take advantage of the interchannel correlations that are often present. As an exception, Kleijn et al. [14] generated multichannel (in the experiment only stereo) as a mono channel + spatial metadata, and used generative methods for the spatial information. Two other works [15, 16] propose neural codecs which are capable of binaural speech and general speech coding for one/two speakers, respectively. However, as opposed to being more generative in nature, the present study aims to replicate and compare to the original multichannel spatial impression of non-speech material.

In the field of music generation, stereo audio output seems more common than in coding [17, 18, 19]. One possible reason is that compression and faithful reproduction of the reference signal is viewed as most suitable for content with more learnable structure than in unconstrained music, including speech (which is typically mono). Applying music generation modeling techniques to audio coding and obtaining benefits in terms of multichannel handling is not trivial. Music generation uses more relaxed measures of goodness that are not applicable to the coding paradigm. Also, as music generators typically have no audio input, not many insights can be obtained for encoding. Nevertheless, neural audio decoding and (conditional) audio generation are closely related problems.

While acknowledging the transparency scaling issues and taking inspiration from the music generation studies, our goal is to remain within the end-to-end audio coding paradigm. Nevertheless, generative and compressive neural methods can be relevant for the coding of spatial aspects even when striving toward perceptual transparency, as discussed in Sec. 3.

We limit the practical testing of this study to ambient, scene-based background material that is typically relatively constrained in structure. Often such background element is a less critical part of the content in e.g. mixed presentations of multiple audio elements [11], and could benefit from efficient multichannel compression due to its high channel count.

However, we also postulate that the application of the proposed neural architecture is not limited to ambient background material, in cases where other types of training material is available. Such material may include more discrete moving objects, or other types of audio, such as music. As our general model is a universal approximator [20], this notion is neither unrealistic nor surprising if the model size is not limited. Nonetheless, the key contribution of this paper is to illustrate the spatial coding efficiency potential of the convolutional arithmetic, as proposed by our testing with scene-based material.

2.2 Ambisonics

Ambisonics is a prominent audio representation widely utilized for spatial capture [1]. This section briefly summarizes the technical aspects of the format. Higher Order Ambisonics (HOA) leverages spherical harmonics coefficients for sound field encoding. These coefficients, in conjunction with spherical harmonics as basis functions, are summed to approximate the original recorded sound field. Mathematically, for a given radius r, angle θ, and wave number k, the pressure in

a plane is represented by Equation 1, where J_n is the n-th order Bessel function and $B_{mm}^{\pm 1}$ denotes the HOA coefficients.

$$p(r,\theta) = B_{00}^{+1} J_0(kr) + \sum_{m=1}^{\infty} J_m(kr) B_{mm}^{+1} \sqrt{2} cos(m\theta) + \sum_{m=1}^{\infty} J_m(kr) B_{mm}^{-1} \sqrt{2} sin(m\theta) \quad (1)$$

To achieve faithful reproduction of the original sound field, an infinite number of spherical harmonics would theoretically be needed. However, practical implementations necessitate truncation to a finite order M, where increased order yields enhanced fidelity. The resultant truncated multichannel HOA signal, known as B-format, demands $(M+1)^2$ Ambisonics channels to represent the sound field accurately. For instance, first-order Ambisonics contains 4 channels, second-order Ambisonics contains 9 channels, and third-order Ambisonics contains 16 channels. Decoding these channels to speaker signals involves applying a decoding matrix contingent upon the HOA B- format order and speaker positioning.

Despite its efficacy in capturing immersive audio environments, HOA poses challenges regarding data compression and transmission. The high channel count inherent in HOA entails substantial bitrate requirements for transmission. Thus, efficient compression schemes tailored to HOA data are imperative for practical deployment in various applications. Previous methods have relied on classical signal processing techniques involving linear combinations of lower-order channels or past channels [21] or using techniques such as SVD to reduce the dimensionality of the Ambisonics signal [22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32]. These compression methods are able to reduce bitrates from tens of megabits/second to the order of hundreds of kilobits/second. However, as proposed in this work, it may be possible to further reduce bitrates using a data-driven neural approach.

One mitigating aspect for real-life applications may be that the material typically captured with HOA is more truncated in terms of probability distribution than unconstrained general audio. Modeling a general domain such as “all music" is a hard problem, and many neural coding systems opt to test only a truncated distribution like speech in order to get good quality. We argue that common uses of HOA would also involve a truncated distribution: spatial capture is often applied to background ambience and/or the overall scene of e.g. a live event in transmission systems such as [11].

In the former case, the HOA element can be thought as "bed" [33], where it can be complemented with separately coded discrete elements such as audio objects [34, 8, 11]. In the latter case even without the additional audio elements, the scene-based HOA capture can hypothetically be distinguished from generic music etc. material by having added ambience and reverberation, and less perceptually discrete sounds. Both of these use cases also apply to synthetic HOA in VR/AR, where again, the discrete virtual elements are more naturally handled by audio objects [34].

While the above implies that a major share of HOA content can be less difficult to compress in the traditional audio coding sense, for spatial fidelity it desirable to use at least third-order Ambisonics [35]. This in turn implies the need for efficient channel-wise compression of HOA content with 16 or more channels. Therefore, real-life application of neural methods for scene-, and bed-based audio can benefit from the low rates offered by data-driven compression, and can be viable even without scaling to high-quality transparency. To illustrate, we test our proposed method for scene-based HOA coding with commonly available material.

3 Methods

3.1 Model Architecture

The basic idea of the present multichannel architecture extension can be summarized as simply increasing the channel count of the first and last convolutional layer of RVQGAN to match the number of audio channels. In the case of third-order Ambisonics material, this channel count is 16. If desirable, it is possible to keep the dimensionality, and therefore the compression efficiency of the remaining model layers. The change in the first model layer is illustrated in Fig. 1. Besides this architechture modification, the present proposal implies non-trivial changes to the typical loss functions, which is discussed more in Sec. 3.3.

More formally, the standard convolutional layer used in deep neural networks (DNNs) [36] utilizes multiple channels and kernels, which can be also thought of

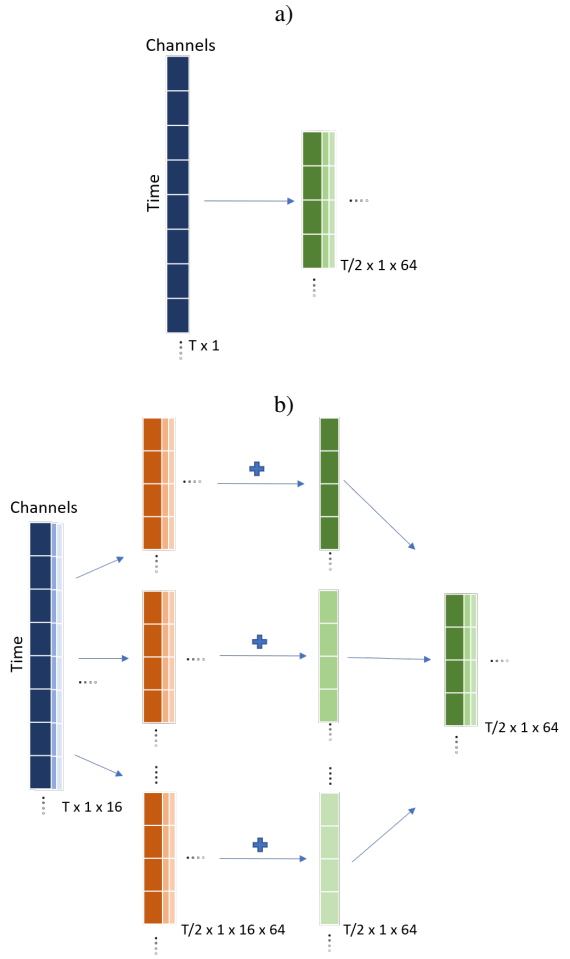


Fig. 1: Visualization of the first layer in a) the single-channel Descript [3] and b) the proposed codec for third-order Ambisonics. The dimensions of the tensors are indicated at each processing stage.

as FIR filters. Time-domain PCM audio signals are typically treated as 1-D time series signals, i.e. the kernels are one-dimensional (as in Conv1D layers). In this case, the layer output for the output channel c_{out} is

$$\begin{aligned}\text{output}(c_{out}) &= \text{bias}(c_{out}) \\ &+ \sum_{k=0}^{n_{in}-1} \text{kernel}(c_{out},k)*\text{input}(k)\end{aligned} \tag{2}$$

where * is the valid cross-correlation operator, and k denotes the input channel index. While the sequence index is omitted in Eq. 2, the layer input and output have channel count and sequence length of $[n_{in}, l_{in}]$ and $[n_{out}, l_{out}]$, respectively. Each input channel is processed with their own dedicated kernel tailored to the specifics of that signal, and the results are summed to the next layer signal. The kernel filters are optimized to both find the inter-channel structure, and to best represent the linear combination of channels in the output sequences.

As seen from Fig. 1 and Eq. 2, the first layer only performs an upmixing operation in traditional neural codecs with a single-channel audio input. It is not obvious how a multichannel audio input should be processed for maximum efficiency. In addition to a brute-force solution of running a mono model separately in each audio channel, an alternative method could be to capture the inter-channel structure via two-dimensional kernels (as in Conv2D layers). However, this would increase the computational complexity, and introduce many separate notions of channels in the model, which are arguably more difficult to interpret. The present solution is rather using Conv1D layers and the standard arithmetic of Eq. 2, while increasing the input and output channels counts to match those of the source audio.

3.2 Perceptual Considerations

Justifications for the method of Sec. 3.1 being able to handle the inter-channel aspects of audio can be argued when considering human spatial perception [37]. Much of auditory localization can be explained by interaural time- and level differences, and the overall spatial envelopment is tied to interaural correlation and coherence [37]. A simple model to characterize the relationship between the audible ear signals, to which the above psychoacoustic results apply, and the codec input content is to consider them being related by a linear transformation.

If $X \in \mathbb{C}^{n\times l}$ is a (frequency-domain) multichannel input signal matrix of n channels of length l, the ear signals can be modeled as $E = AX$, where $A \in \mathbb{C}^{2\times n}$ is a (time- and frequency dependent) transform. Processes that can be reasonably approximated by this model include typical gain-based loudspeaker rendering systems for various content [38], and their combination with loudspeaker room impulse responses (RIRs). RIRs are

generally considered to represent linear time-invariant (LTI) systems. Similarly, binaural headphone renderers also typically operate as a combination of virtual loudspeakers, decorrelators and LTI head-related transfer functions [39].

Assuming such a simple LTI model, it can be seen that preserving the covariance structure of the input multichannel signal (with a reasonable time- and frequency resolution) would also result in approximating the audible interaural covariance:

$$R_E = E\overline{E} \approx AX\overline{(AX)} = AX\overline{XA} = AR_X\overline{A}, \quad (3)$$

where R_E and R_X represent the covariance matrices of the listener-audible-, and the original-format audio, and $\overline{\ }$ indicates the conjugate transpose. It is simple to see that the covariance matrix encompasses information about the inter-channel temporal and level differences, as well as correlations.

Accordingly, it has been found that such preservation of the original audio convariance structure is a highly useful target in state-of-the-art spatial coding systems, and it also accounts for possible rendering of the output afterward [9, 10, 34]. Traditional codecs also show that spatial attribute parameters can be quantized heavily, and are typically assigned less bitrate compared to waveform coding, and can be more generative in nature. Signal covariance is also discussed further in Sec. 3.3 in relation to loss functions.

With the above reasoning that preserving inter-channel properties of the original format audio constitutes in good perceptual spatial quality, the question becomes can the neural network convolution arithmetic in Eq. 2 capture and represent R_X? While it is generally difficult to derive any formal proofs related to DNNs, simple thought experiments e.g. about the operation of the model first layer seem to corroborate this. The sum of the channel-dependent kernel filter values can represent different values of inter-channel temporal (phase) and level differences, as well as correlations, given suitably well learned kernels. This informal reasoning, along with the universal approximator theorem [20] are postulated to warrant the present research.

3.3 Loss Functions

The original Descript RVQGAN model was trained with the combination of adversarial-, feature matching-, VQ codebook-, and reconstruction losses [3]. Since our model output is multichannel, we adjust the relevant losses to operate on each channel independently and finally take the expected value over channels as the final loss. For the adversarial discriminator, this requires changes to the multi-scale (MSD) and multi-period and waveform discriminators (MPD), as well as the multi-resolution spectrogram discriminators (MRSD) to produce a multichannel output.

We found that despite not looking at the channel interactions specifically, the previous paradigm already produces a good spatial impression for the present HOA content. For fine-tuning of the spatial quality, we consider a loss for the mismatch of the interchannel covariance structure, which was in Sec. 3.1 connected to the perceived interaural coherence, and therefore an important descriptor of perceptual spatial impression of the rendered output [37, 40, 41]. Attributes this loss aims to preserve are e.g. envelopment and diffuseness of the material. For content where these are important, the covariance loss is a simple way to emphasize them. It is also more general than the specialized loss functions proposed for stereo audio [18, 42].

The covariance loss is calculated as the expected L1 loss between the normalized channel-wise covariance matrices of the original and the model f_θ reconstruction time-domain signals:

$$L_{cov} = \frac{1}{2}\sum_{i=0}^{n}\sum_{j=0}^{n}\left|\left|\frac{C_{ij}}{\sqrt{C_{ii}C_{jj}}} - \frac{\hat{C}_{ij}}{\sqrt{\hat{C}_{ii}\hat{C}_{jj}}}\right|\right|, \quad (4)$$

where $C = \mathrm{cov}(\mathbf{x})$ and $\hat{C} = \mathrm{cov}(f_\theta(\mathbf{x}))$ are the covariance matrices of the multichannel time-domain input and model output signal of n channels, respectively. The model architecture and parameters are noted by f and θ. Each element of the normalized matrix is then the Pearson correlation coefficient between channels i and j.

We evaluate the covariance broadband, but it is also possible to have the measure be frequency-dependent. We also experimented using non-normalized covariance instead of correlation coefficient, but found the latter more appealing for numerical stability, and the fact that the other losses seemed to also account for channel energies and level differences.

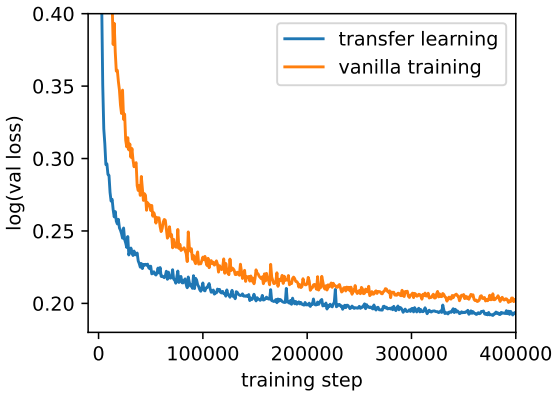


Fig. 2: Example comparison between proposed transfer learning utilizing pre-trained single-channel model weights, and vanilla random initialization. Plot shows the development of the reconstruction multi-scale mel validation loss [3].

3.4 Transfer Learning from Single-Channel Models

We propose applying transfer learning to the multichannel model from previously trained single-channel model weights. In input and output layers, where the number of channels is increased, we replicate the original convolutional weights of the original channel to the multiple channels of the new model identically. Thus, all input and output channels of the multichannel model are processed with identical convolutional kernels at the start of the training, which then evolve as needed.

In practice, we construct the new model and copy the appropriate parameters from the state dictionary of the published 16 kbps version of Descript single-channel model [3]. In addition to helping with RVQGAN training time consumption, transfer learning also helps if the fine-tuning stage is performed on less powerful hardware and smaller batch size. Fig. 2 illustrates the benefit of the proposed transfer learning in both faster conversion, and final error with reasonable amount of steps. In our experiments, we do not apply transfer learning to the adversarial discriminator model, it is only randomly initialized as in the original work [3].

4 Experiment

Immersive audio material available for data-driven modeling is not as abundant as traditional formats. In this paper, we have utilized the EigenScape database [43]. It consists of eight acoustic scenes recorded spatially in fourth-order Ambisonics that are ambient in nature, but do sometimes include e.g. people speaking. We omit the additional channels and process the signals to obtain 16-channel, third-order inputs at 16-bit / 44100 Hz resolution. For cross-validation, we utilize 7/8 of the samples on each scene for model training, and the keep the rest separate for validation and subjective testing. A multiple-round alternative of keeping each of the 8 individual scenes for validation, and testing them in turn was unfortunately not feasible due to subjective testing being used.

The model implementation was built on the Descript codebase [3] with the aforementioned multichannel extensions to the architecture and loss calculation (see Sec. 3). The only other changes were to omit the output Tanh-nonlinearity, and trivial modifications to make the code dependencies work with 16-channel audio. The training procedure remains largely the same as in the original work, with the exception of utilizing the transfer learning scheme proposed in Sec. 3.4. We use weighting of 1.0 for the covariance loss. The other loss weightings were kept as is (15.0 for the multi-scale mel loss, 2.0 for the feature matching loss, 1.0 for the adversarial loss and 1.0, 0.25 for the codebook and commitment losses, respectively). To illustrate a simple fine tuning procedure results, we trained for a modest number of steps (400,000) and batch size (24) of 5-second samples.

A MUSHRA [44] listening test was carried out. For quality reference, we utilize Opus channel mapping family 3 coding intended for HOA content [45], at 160 kbps rate, which is on the lower quality region of the Opus HOA, to demonstrate how conventional coding methods with 10x bitrate compare. In total, four listening conditions were compared:

- Low anchor (LA) - 3.5kHz low-pass filtered version of the reference
- Opus' Channel Family Mapping 3 (OPUS) compression at 160 kbps
- Proposed neural codec (PROP) at 16 kbps
- Hidden reference (H-REF)

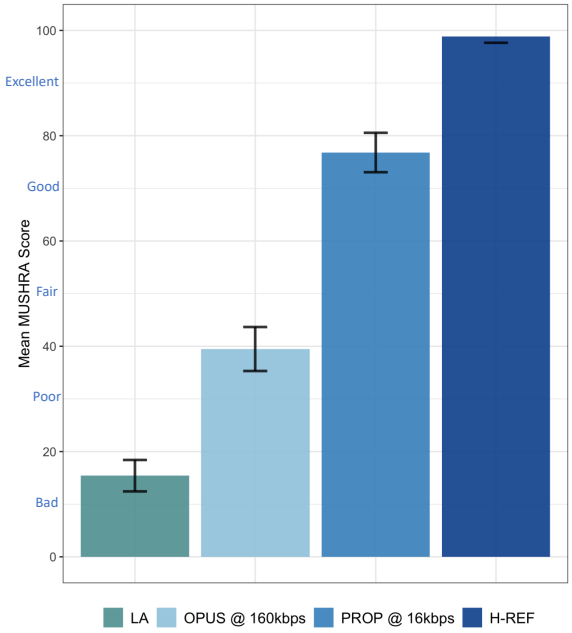


Fig. 3: MUSHRA score mean and 95% confidence interval over 8 tracks and 8 listeners, with 7.1.4 immersive loudspeaker listening.

Rendering the HOA signals to the 7.1.4 loudspeaker layout was done with the IAMF system [11] which utilizes a version of the EBU ADM Renderer [46]. Listening tests took place in a 7m (L) x 5.33m (W) x 3.05 (H) listening room equipped with a 7.1.4 playback system. The loudspeakers' and listener's positions were based on [47]. Loudspeakers were level-matched at the listener's position and meet the ITU-R specification [48].

Eight listeners completed the MUSHRA test. Participants were told to compare the four listening conditions to the known reference, and to rate the overall sound quality on the MUSHRA 100-point scale, while focusing on the audio quality and spatial impression correctness.

5 Results and Discussion

As can be seen from Fig. 3, the proposed neural codec operating on 16 kbps is able to achieve "good" quality on the MUSHRA scale, and outperform a traditional method with 10x less compression rate. The traditional Opus method [45] starts to break down at 160 kbps, and its quality would improve at higher rates. However, for good quality the Opus HOA codec rate needs typically to be considerably higher, around 32 kbps per channel [49]. Here we opted to illustrate the quality potential of the neural methods with a simple comparison.

The audio material used here is from a truncated distribution of ambient scenes, and also limited in amount in order to illustrate a typical use case, as discussed in Sec. 2.2. However, the use of transfer learning from a generic model arguably retains some level of baseline generality. Also, the listening test conditions did enable critical evaluation of the sound quality similarly as in typical immersive audio testing. Although the present material was ambient background in nature, we did not test mixed presentations with additional foreground audio elements [11], which would have made the HOA coding quality less critical.

It should be noted that informal listening with few samples of cinematic music outside the EigenScape database indicated that applying the proposed model trained only on scene-based material may not be as optimal for all content. Other content types or more generic models can also require an architecture with an increased bottleneck bitrate to achieve good quality. More general training and application of the model with e.g. musical and cinematic content is left as future work.

Although not tested here, the present modeling principle is also applicable to multichannel audio formats other than third-order Ambisonics (e.g. 5.1, 7.1, 7.1.4 etc.), as long as there is sufficient training data. Despite the promising results, the main caveats of the present method, and most neural codecs remain 1) the lack of scaling to different bitrates without retraining or model modification, and 2) the model computational complexity. For these, future work will hopefully provide solutions.

References

[1] Daniel, J., Moreau, S., and Nicol, R., "Further Investigations of High-Order Ambisonics and Wavefield Synthesis for Holophonic Sound Imaging," in *Audio Engineering Society Convention 114*, 2003.

[2] Zeghidour, N., Luebs, A., Omran, A., Skoglund, J., and Tagliasacchi, M., "SoundStream: An End-to-End Neural Audio Codec," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30, pp. 495–507, 2021.

[3] Kumar, R., Seetharaman, P., Luebs, A., Kumar, I., and Kumar, K., "High-Fidelity Audio Compression with Improved RVQGAN," 2023, arXiv 2306.06546.

[4] Zhang, Z. et al., "High-Fidelity Diffusion-Based Audio Codec," *2024 18th International Workshop on Acoustic Signal Enhancement (IWAENC)*, pp. 344–348, 2024.

[5] Roman, R. S., Adi, Y., Deleforge, A., Serizel, R., Synnaeve, G., and D'efossez, A., "From Discrete Tokens to High-Fidelity Audio Using Multi-Band Diffusion," 2023, arXiv 2308.02560.

[6] Gu, Y. and Diao, E., "ESC: Efficient Speech Coding with Cross-Scale Residual Vector Quantized Transformers," 2024, arXiv 2404.19441.

[7] Pedro J. Villasana T., Villemoes, L., Klejsa, J., and Hedelin, P., "Audio Decoding by Inverse Problem Solving," 2024, arXiv 2409.07858.

[8] Robinson, C., Tsingos, N., and Mehta, S., "Scalable Format and Tools to Extend the Possibilities of Cinema Audio," *SMPTE Motion Imaging Journal*, 121(8), 2012.

[9] ETSI TS 103 190 parts 1 2, "Digital Audio Compression (AC-4) Standard," Standard, European Telecommunications Standards Institute, Geneva, CH, 2018.

[10] ISO/IEC 23008-3:2022, "Information Technology - High Efficiency Coding and Media Delivery in Heterogeneous Environments - Part 3: 3D audio," Standard, International Organization for Standardization, Geneva, CH, 2022.

[11] "Immersive Audio Model and Formats," `https://aomedia.org/iamf/`, 2024.

[12] ITU-R BS.2051-3, "Advanced Sound System for Programme Production," Standard, International Telecommunication Union, Geneva, CH, 2022.

[13] D'efossez, A., Copet, J., Synnaeve, G., and Adi, Y., "High Fidelity Neural Audio Compression," 2022, arXiv 2210.13438.

[14] Kleijn, W. B., Chinen, M., Lim, F. S. C., and Skoglund, J., "Multi-Channel Audio Signal Generation," in *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5, 2023.

[15] Xu, Z., Xu, Y., Kothapally, V., Wang, H., Yang, M., and Yu, D., "SPATIALCODEC: Neural Spatial Speech Coding," in *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1131–1135, 2024, doi:10.1109/ICASSP48485.2024.10445853.

[16] Ratnarajah, A., Zhang, S.-X., and Yu, D., "M3-audiodec: Multi-Channel Multi-Speaker Multi-Spatial Audio Codec," *arXiv preprint arXiv:2309.07416*, 2023.

[17] Copet, J. et al., "Simple and Controllable Music Generation," 2024, arXiv 2306.05284.

[18] Evans, Z., Carr, C., Taylor, J., Hawley, S. H., and Pons, J., "Fast Timing-Conditioned Latent Audio Diffusion," 2024, arXiv 2402.04825.

[19] Evans, Z., Parker, J. D., Carr, C., Zukowski, Z., Taylor, J., and Pons, J., "Long-Form Music Generation with Latent Diffusion," 2024, arXiv 2404.10301.

[20] Zhou, D.-X., "Universality of Deep Convolutional Neural Networks," *Applied and Computational Harmonic Analysis*, 48(2), pp. 787–794, 2020.

[21] Hellerud, E., Solvang, A., and Svensson, U. P., "Spatial redundancy in Higher Order Ambisonics and its use for lowdelay lossless compression," in *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 269–272, 2009.

[22] Zamani, S., Nanjundaswamy, T., and Rose, K., "Frequency Domain Singular Value Decomposition for Efficient Spatial Audio Coding," in *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pp. 126–130, 2017.

[23] Zamani, S. and Rose, K., "Spatial Audio Coding with Backward-Adaptive Singular Value Decomposition," in *Audio Engineering Society Convention 145*, Audio Engineering Society, 2018.

[24] Zamani, S. and Rose, K., "Spatial Audio Coding Without Recourse to Background Signal Compression," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 720–724, 2019.

[25] Mahé, P., Ragot, S., and Marchand, S., "First-Order Ambisonic Coding with Quaternion-Based Interpolation of PCA Rotation Matrices," in *EAA Spatial Audio Signal Processing Symposium*, pp. 7–12, Paris, France, 2019, doi:10.25836/sasp.2019.19.

[26] Xu, J., Niu, Y., Wu, X., and Qu, T., "Higher order Ambisonics Compression Method Based on Independent Component Analysis," in *Audio Engineering Society Convention 150*, Audio Engineering Society, 2021.

[27] Namazi, M., Elshafiy, A., and Rose, K., "Spatial Audio Compression with Adaptive Singular Value Decomposition using Reconstructed Frames," in *Audio Engineering Society Conference: 2022 AES International Conference on Audio for Virtual and Augmented Reality*, Audio Engineering Society, 2022.

[28] Namazi, M., Elshafiy, A., and Rose, K., "On Ultra Low-Delay Compression of Higher Order Ambisonics Signals," in *2024 Data Compression Conference (DCC)*, pp. 512–521, 2024.

[29] Namazi, M. and Rose, K., "Ultra-Low Delay Lossless Compression of Higher Order Ambisonics," in *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 791–795, 2024.

[30] Hold, C., McCormack, L., Politis, A., and Pulkki, V., "Perceptually-Motivated Spatial Audio Codec for Higher-Order Ambisonics Compression," in *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1121–1125, 2024.

[31] Namazi, M., *Advancements in Higher Order Ambisonics Compression and Loss Concealment Techniques*, Ph.D. thesis, UC Santa Barbara, 2024.

[32] Namazi, M., Hirvonen, T., Bharitkar, S., and Souza-Blanes, E., "Pre- and Post-processing of Ambisonics Signals for Legacy Audio Codecs," in *AES 156th Convention*, Audio Engineering Society, 2024.

[33] Riedmiller, J., Mehta, S., Tsingos, N., and Boon, P., "Immersive and Personalized Audio: A Practical System for Enabling Interchange, Distribution, and Delivery of Next-Generation Audio Experiences," *SMPTE Motion Imaging J*, 124(5), pp. 1–23, 2015.

[34] Engdegård, J. et al., "MPEG Spatial Audio Object Coding - The ISO/MPEG Standard for Efficient Coding of Interactive Audio Scenes," in *AES 129th Convention*, Audio Engineering Society, 2010.

[35] Deutsch, T., Vollmer, L. J., and Fels, J., "An Assessment of Spatial Fidelity of Higher-Order Ambisonics Reproduction Using the SCaLAr System," in *47th Anual Conference*, Deutsche Gesellschaft für Akustik e.V. (DEGA), 2021.

[36] Dumoulin, V. and Visin, F., "A Guide to Convolution Arithmetic for Deep Learning," 2018, arXiv 1603.07285.

[37] Blauert, J., "Spatial Hearing: The Psychophysics of Human Sound Localization (Revised Edition)," 1997.

[38] ITU-R, "Audio Definition Model Renderer for Advanced Sound Systems," Recommendation BS.2127-1, International Telecommunication Union, Geneva, 2023.

[39] European Broadcasting Union, "Binaural EBU ADM Renderer (BEAR) for object-based sound over headphones," *European Broadcasting Union, Geneva, Switzerland, Tech. Rep*, 2023.

[40] Blauert, J. and Lindemann, W., "Spatial Mapping of Intracranial Auditory Events for Various Degrees of Interaural Coherence," *The Journal of the Acoustical Society of America*, 79(3), pp. 806–813, 1986.

[41] Luke, R., Innes-Brown, H., Undurraga, J. A., and Mcalpine, D., "Human Cortical Processing of Interaural Coherence," *iScience*, 5(25), 2022.

[42] Steinmetz, C. J., Pons, J., Pascual, S., and Serrà, J., "Automatic Multitrack Mixing with a Differentiable Mixing Console of Neural Audio Effects," 2020, arXiv 2010.10291.

[43] Green, M. C. and Murphy, D., "EigenScape," `https://doi.org/10.5281/zenodo.1012809`, 2017.

[44] ITU-R, "Method for the Subjective Assessment of Intermediate Quality Level of Audio Systems," Recommendation BS.1534-2, International Telecommunication Union, Geneva, 2014.

[45] Skoglund, J. and Graczyk, M., "Ambisonics in an Ogg Opus Container," `https://www.rfc-editor.org/rfc/rfc8486.html`, 2018.

[46] ITU-R, "Audio Definition Model Renderer for Advanced Sound Systems," Recommendation BS.2127-1, International Telecommunication Union, Geneva, 2023.

[47] ITU-R, "Advanced Sound System for Programme Production," Recommendation BS.2051-3, International Telecommunication Union, Geneva, 2022.

[48] ITU-R, "Methods for the Subjective Assessment of Small Impairments in Audio Systems," Recommendation BS.1116-3, International Telecommunication Union, Geneva, 2015.

[49] Souza-Blanes, E., Tejeda-Ocampo, C., and Bharitkar, S., "A Closer Look on Bitrate Requirements for Opus with First Order Ambisonics and All-RAD rendering to 5.1 and 7.1.4," in *AES 153th Convention*, Audio Engineering Society, 2022.

Audio Engineering Society

Conference Paper 4

Presented at the AES International Conference on Machine Learning and Artificial Intelligence for Audio 2025 September 8–10, London, UK

This paper was peer-reviewed as a complete manuscript for presentation at this conference. This paper is available in the AES E-Library (http://www.aes.org/e-lib), all rights reserved. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Journal of the Audio Engineering Society.

Extraction and neural synthesis of low-order spectral components for Head-Related Transfer Functions

Mary Pilataki[1, 2], Chris G. Buchanan[1], and Cal Armstrong[1]

[1] *PlayStation, Sony Interactive Entertainment, London UK*
[2] *Queen Mary University of London, London, UK*

Correspondence should be addressed to Cal Armstrong (calum.armstrong@sony.com)

ABSTRACT

Audio spatialisation is a crucial component of immersive technologies, shaping the quality of experiences in VR, AR and gaming. Spatial audio rendering relies on Head-Related Transfer Functions (HRTFs), which encode both spatial localisation cues and listener-dependent spectral characteristics associated with perceptual attributes such as sound colouration. However, research has largely prioritised localisation accuracy, leaving the remaining subject-specific spectral features underexplored. Meanwhile, 3D audio renderers often employ HRTFs with entangled spectral colouration profiles and localisation cues, leading to cases where localisation is accurate, but the resulting sound coloration is perceived as unnatural or undesirable. In this paper, we introduce a novel signal processing method, which we term the Delta Spatial Transfer Function (DSTF) which aims to separate low-order spectral characteristics from directional cues. We also present a conditional autoencoder model trained to reconstruct and synthesise new DSTF profiles. To ensure disentanglement between localisation and DSTF characteristics, we incorporate adversarial learning. The resulting model enables (1) synthesis of novel location-agnostic spectral characteristics, (2) neural compression of databases of such features, and (3) analysis and exploration of HRTF variation beyond spatial encoding.

1 Introduction

Head-Related Transfer Functions (HRTFs) are frequency- and time-dependent filters that represent the stereo anechoic acoustic transfer function between a positional sound source and a listener's ears. In the time domain, these are referred to as Head-Related Impulse Responses (HRIRs). An individual's HRTF is commonly measured at various angles around their head, typically referenced with respect to azimuth and elevation. The left and right ear responses differ and are both encoded into the HRTF. Examples of the specialised equipment used to measure HRTFs can be found in [1, 2, 3].

HRTFs provide critical spatial cues that enable listeners to localise sounds in three-dimensional (3D) space. These cues include interaural time differences (ITD) and interaural level differences (ILD), primarily associated with lateral localisation, and the frequency-dependent peaks and notches in the ear's response, which are key for elevatory localisation [4]. By convolving an audio signal with an HRIR and presenting the result directly to a listener's ears (typically via

headphones), a sound source can be simulated as if it originates from the direction in which the HRTF was measured [1]. This binaural rendering process has become a fundamental technique for simulating immersive acoustic environments in augmented reality (AR), virtual reality (VR), gaming, and entertainment.

In addition to localisation, HRTFs are associated with other perceptual attributes, including realism, externalisation, and colouration [4, 5]. While early efforts in HRTF personalisation predominantly focused on improving localisation accuracy, subject-specific spectral characteristics associated with such perceptual attributes, particularly those related to perceived coloration, have been under-explored. When an HRTF is applied to an audio signal, distinct changes in sound colouration are perceived. This colouration is primarily shaped by low-order spectral components that vary across listeners, influenced by factors such as the shape of the head, pinnae, and torso, as well as early reflections from the shoulders and torso [4]. Clearly, these physical variations not only affect the localisation cues within the HRTF but also impact sound coloration. Furthermore, listening test results have shown that the sound colouration introduced by a different set of HRTF may be more pleasant for some people than that of their own HRTF [1]. Spatially accurate but perceptually unpleasant renderings can disrupt immersion, suggesting that non-localisation-related spectral variation should be treated as a first-class perceptual attribute in spatial audio design.

We propose a novel signal processing method to analyse HRTFs and extract listener-dependent low-order spectral features. Using these features, we train a conditional autoencoder model in an adversarial learning framework to disentangle localisation and non-spatial spectral information such that the latent space exclusively encodes direction-independent spectral coloration components. The decoder module can then be used as a synthesiser for novel spectral characteristics, providing separate control over localisation and coloration effects. This enables exploration of the non-directional spectral variation in HRTFs. By sampling latent embeddings, we can generate a set of spectral filters across a 3D sphere, corresponding to different directions. Our model can be employed as a tool for personalising HRTFs by substituting spectral colouration filters within existing HRTFs to modify their perceptual character without altering localisation cues.

Finally, the learned latent space can serve as a neural compression tool, enabling efficient storage of a pre-calculated selection of such low-order spectral filters. The paper is organised as follows. Section 2 reviews state-of-the-art research on HRTF personalisation and spectral modelling using autoencoders. In Section 3, we describe the Sonicom HRTF database used in our experiments. Section 4 details the experimental setup, including feature extraction, dataset creation, and the deep learning model architecture and training details. Section 5 presents the results, discussing spectral reconstruction, synthesis, and neural compression, followed by the conclusion in Section 6.

2 Related Work

2.1 Head-Related Transfer Functions

Although HRTF personalisation is an active area of research, the analysis and synthesis of non-localisation-related spectral characteristics as an exclusive feature has not been extensively explored in the literature. Marentakis and Saitis [6] analyze the changes in the timbre descriptors *log attack time* and *spectral centroid* between anechoic and HRTF-filtered sounds. Some comparison is also presented between an individual's left and right ear HRTFs; however, variations in HRTFs between different subjects are beyond the scope of that work. Marggraf-Turley et al. [7] propose a method for HRTF recommendation based on Predicted Binaural Colouration (PBC) [8], where the goal is to minimise the colouration perceived by a target subject when compared to their own individual HRTF. Although this method does account for sound colouration, it is only relative to an individual's HRTF, does not allow for further personalisation, and requires the estimation of an individualised HRTF based on anthropometric features.

In this work, we focus on the perceptual attribute of sound coloration by removing dominant localisation-related information and considering only the residual subject-variant elements.

2.2 Spectral modeling with autoencoders

Variational autoencoders (VAEs) have been proposed for music audio coding, synthesis of novel timbres, and timbre exploration, where a low-dimensional latent space is learned through a guided process to achieve specific goals. For example, Esling et al. [9] propose

regularisation to ensure the latent space matches timbre qualities identified in timbre research. Colonel and Keene [10] propose conditioning using harmonic features so that the latent space becomes uniform in terms of pitch and timbre. Puche and Lee [11] also use conditioning combined with adversarial learning to disentangle pitch and timbre, ensuring that the latent space exclusively encodes non-pitch-related spectral information.

Inspired by this work and motivated by the findings in [12], which highlight the effectiveness of disentangling qualities in an autoencoder's latent space, we guide the learning process of our framework so that the latent space exclusively encodes the low-order, location-invariant spectral components of HRTFs that mainly affect coloration. Following a similar approach to [11], we employ adversarial learning to separate direction-dependent localisation cues from subject-dependent spectral coloration. This framework allows us to generate novel spectral features and provides separate control over both directional and subject-specific spectral components, enabling targeted manipulation of perceptual coloration in HRTFs.

3 HRTF Database

For our experiments, we use the Sonicom database [3], which consists of 200 subjects, each with 828 HRTF measurements. Azimuths are sampled in 5° increments between -175°to 180°. Elevations are sampled every 10° between -30° to 30°, and every 15° below and above that (-45°to 90°).

We use free-field compensated HRTFs, recorded at a sampling rate of 48 kHz. Each HRTF in the database is preprocessed as follows: 1) We remove unwanted floor reflections by applying a 75-sample unity pad followed by a 50-sample half-Hann closing window, 2) we apply diffuse field equalisation using the method proposed in [1], 3) we apply loudness normalisation according to EBU R128 [13], 4) we re-interpolate the measurement grid to a new set of azimuths ranging from -172.5° to 180° and elevations between -60° to 60°. The interpolation is performed in the frequency domain using the barycentric weighted Delaunay triangle method.

4 Method

4.1 Dataset

Each ear in the database is treated as a distinct data instance, with a unique Delta Spatial Transfer Function (DSTF) feature associated with every ear. To enforce left-right symmetry prior to feature extraction (detailed in Section 4.2), all right-ear measurements are horizontally flipped to represent left-ear equivalents. This results in a total of $200 \times 2 = 400$ unique DSTFs across the dataset. As described in Section 4.2 and shown in Figures 1 and 4, each DSTF feature corresponds to a single ear at a specific azimuth and covers 17 elevation angles from -60°to 60°. With 48 azimuth angles and 400 DSTFs, the full dataset comprises $48 \times 400 = 19{,}200$ DSTF features.

The dataset is split into training and testing subsets using an approximate 85%–15% ratio. To construct the test set, we randomly select 10 azimuth angles for 146 unique left-ear instances and 146 right-ear instances (flipped to left-ear equivalents), resulting in 2,920 DSTF features for evaluation. This yields $2 \times 146 \times 10 = 2{,}920$ DSTF features for evaluation, with the remaining features used for training.

Each DSTF feature is annotated with one-hot encoded labels for both subject identity and azimuth. Subject labels are 400-dimensional binary vectors, where a single 1 indicates which subject (i.e., ear) the feature belongs to. Azimuth labels are 48-dimensional vectors representing azimuths in ascending order: index 0 corresponds to an azimuth of -172.5°, index 1 corresponds to -167.5°, and so on, up to index 47, which corresponds to 180°.

4.2 Feature extraction

In this work, we focus on capturing low-order spectral colourations in HRTFs, which we refer to as the *Delta Spatial Transfer Function* (DSTF). The goal of DSTFT feature extraction is twofold: (1) to remove the diffuse-field response, which may account for residual measurement equalisation or approximate physiological constants such as ear canal resonance and (2) to eliminate features that are known from prior literature to primarily support sound localisation. The remaining components form a filter set that represents the uncharacterised residual elements of the HRTF. We consider the inter-subject variation of this filter set and propose interpreting it as subject-specific sound coloration not directly related to localisation cues.

To extract the DSTF, we apply the following steps:

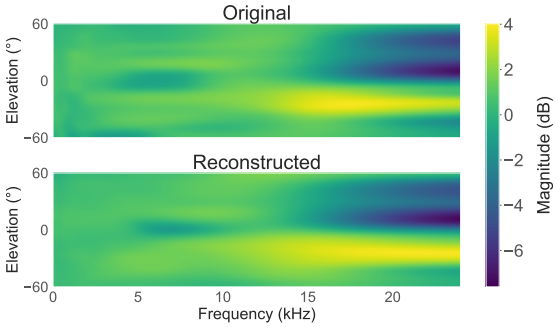


Fig. 1: Comparison between input feature (Original) and model output (Reconstructed) for subject P0001 at azimuth 0°. The x-axis represents frequency (0 Hz up to 24 kHz) and the y-axis represents elevation (-60 up to 60°). Colour represents magnitude values in dB.

1. **Notch Removal**: We first remove spectral notches from each HRTF measurement (per ear, per angle). We use monaural magnitude spectra to ensure ITDs are excluded. Approximate notch frequencies are identified using a Linear Predictive Coding (LPC)-based signal processing method, as described in [14] and [15]. These notch frequencies are determined by detecting local minima (the actual notch frequencies), inverting the spectrum and using a peak prominence detection algorithm (e.g *scipy.peak_prominences* [16]), which returns the left and right bounds of each peak. To finetune this process, we can alternatively calculate various notch width frequencies (e.g. *scipy.peak_widths* [16]), which calculates the notch width, allowing for more accurate definition of the start and end of each notch. After identifying the notch regions, we remove the corresponding frequency bins (by setting them to NaN) and reinterpolate the data over these samples using the *pchip* interpolation algorithm [17]. We refer to the output of the notch removal process as an HRTF′.

2. **ASTF calculation**: We compute the average HRTF′ at each spatial direction across all subjects, yielding the Average Spatial Transfer Function (ASTF). It is important here to distinguish ASTF from Diffuse Field Equalisation. While Diffuse Field Equalisation calculates the average filter for a single subject across all angles, the ASTF is the average filter for a single angle, across all subjects. As a result, the ASTF effectively captures ILDs and average head-related features such as head shadowing.

3. **DSTF calculation**: The Delta Spectral Transfer Function (DSTF) for each subject is defined as the difference between their HRTF′ and the ASTF, as shown in Equation 1, where ϕ denotes the azimuth angle.

$$DSTF_\phi = HRTF'_\phi - ASTF_\phi \tag{1}$$

Table 1: Glossary of key terms used in the feature extraction process.

Term	Definition
HRTF′	HRTF after notch removal and interpolation.
ASTF	Average notchless HRTF across subjects at each azimuth.
DSTF	Deviation from ASTF representing low-order spectral characteristics per subject at each azimuth.
PCHIP	Interpolation algorithm used to fill in spectral notches after removal [17].

For input into our autoencoder model, we treat the DSTF as a series of 2D matrices, where each matrix corresponds to the DSTF at a single azimuth ϕ. Each matrix represents the frequency-domain magnitude data of the DSTF across elevation and frequency. Figures 1 and 4 show example DSTFs at 0° azimuth for subjects P0001 and P0005. Each matrix has dimensions $[257 \times 17]$, corresponding to 257 frequency bins (0 to 24 kHz) and 17 elevation angles (from -60° to 60°). For training, all DSTF features are normalized in $[-1, 1]$ dB.

4.3 Model architecture and training details

The proposed model is a conditional autoencoder composed of four main components: a convolutional Encoder, a convolutional Decoder, a Fully Connected (FC) Subject Identity Classifier, and an FC Azimuth Discriminator. As shown in Figure 2, the architecture is designed to reconstruct DSTF features while disentangling subject identity (unique perceptual attributes of an HRTF, mainly sound coloration) from spatial direction (azimuth). The model is trained using a multi-task

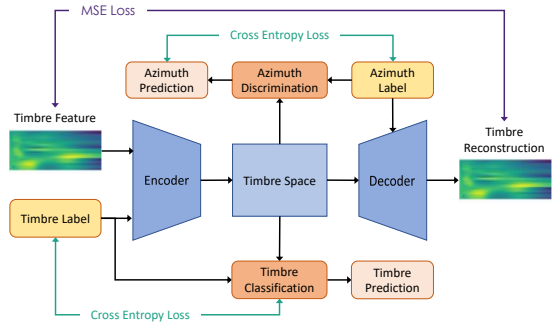


Fig. 2: Autoencoder architecture. The model consists of an Encoder, a Decoder, a Subject Identity Classifier and an Azimuth Discrimination module.

objective combining Mean Squared Error (MSE) for reconstruction and Cross-Entropy (CE) losses for classification. The overall objective is expressed as:

$$Loss = L_{AE} + L_{ID} - L_{AD} \tag{2}$$

L_{AE} denotes the MSE between the Decoder's output and the Encoder's input, both in dB-scaled spectral feature space. L_{TC} is the CE between the predicted and ground truth subject identity, and L_{AD} is the CE between the predicted and ground truth azimuth. During training, the goal is to minimize L_{AE} and L_{ID}, while maximizing L_{AD}, encouraging the Encoder to discard azimuth-specific cues and retain only low-order spectral information in the latent space, associated with perceptual attritrubes, maily sound coloration.

The Encoder is an 8-layer convolutional neural network that compresses input features of size $[257 \times 17](frequency \times elevation)$ into a 64-dimensional latent vector. Except for the first and last layers, each convolution is followed by a LeakyReLU activation with a slope of 0.2 and Batch Normalization.

The Decoder mirrors this structure using transposed convolutions to reconstruct DSTF features from the latent space, again applying LeakyReLU and Batch Normalization on all layers except the final two.

The Subject Identity Classifier is a 4-layer FC network. It takes the 64-dimensional latent vector as input and outputs a 400-dimensional one-hot subject prediction vector. The two hidden layers contain 128 neurons each. The Azimuth Discriminator has a similar structure, with the final output layer containing 48 neurons corresponding to the number of azimuth directions.

During training, input DSTF features are passed to the Encoder, which produces a latent representation. This latent embedding, along with azimuth labels, is fed to the Decoder to reconstruct the original feature. The MSE loss, L_{AE}, is calculated between the input features and the Decoder's reconstructions. The Identity Classifier attempts to predict the correct subject identity, while the Azimuth Discriminator tries to infer the azimuth. Adversarial training encourages the encoder to encode only information relevant to unique perceptual attributes such as sound coloration and suppress localisation cues.

The use of azimuth discrimination is specific to the structure of the features used, which encode data for a single azimuth across multiple elevations. Alternatively, if the dataset encoded a single elevation across multiple azimuths, an elevation discrimination module could be used analogously, by maximizing the classification error on elevation while minimizing reconstruction and subject identity classification losses.

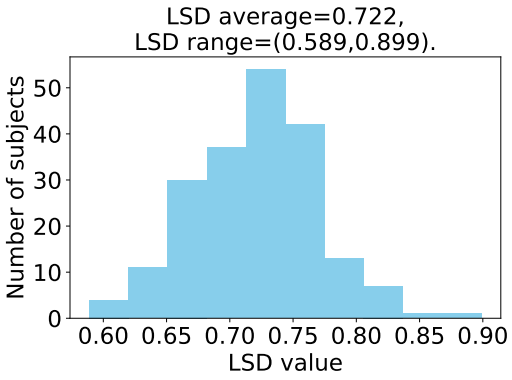
The model is trained for 2,000 epochs and evaluation is performed using the checkpoint of the last epoch. The batch size is set to 12 and the learning rate is set to 0.00005.

5 Results and Discussion

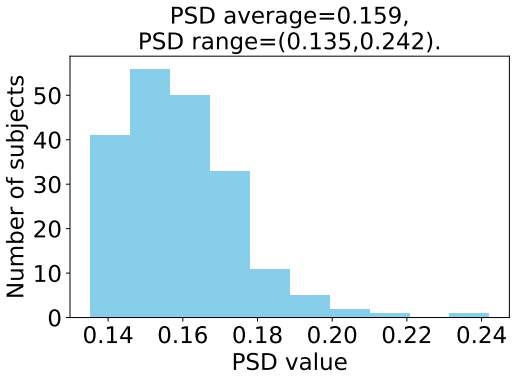
5.1 DSTF reconstruction and synthesis

To evaluate the model's performance, we first assess its reconstruction capability. Prior to evaluation, we smooth each FFT magnitude spectrum in the model's output using a moving average filter, where the window size increases linearly with respect to frequency and the window samples are weighted by a Hann function. Figure 1 shows a typical reconstruction of the left ear of subject P0001 at azimuth 0 (across all elevations). For each subject, we compute the Log Spectral Distance (LSD) [18, 19] and Perceptual Spectral Distance (PSD) [20] between the input and the reconstructed features at all spatial points.

LSD is a commonly used metric for evaluating spectral distortion. It measures the difference between two spectra on a logarithmic scale, reflecting perceptual similarity. As defined in [18], LSD is computed as the



(a) LSD histogram.



(b) PSD histogram.

Fig. 3: Evaluation of the model's reconstruction capability using the Log Spectral Distance (LSD) and Perceptual Spectral Distance (PSD) metrics.

root mean square (RMS) of the frame-wise differences between the log-magnitude spectra of a reference and a test signal, as expressed in Equation 3.

$$LSD(X,\hat{X}) = \sqrt{\frac{1}{K}\sum_{k=1}^{K}\left[20\log_{10}\left(\frac{|X(k)|}{|\hat{X}(k)|}\right)\right]^2} \quad (3)$$

In the equation above, K is the total number of frequency bins, X(k) is the magnitude of the reference spectra at frequency bin k and $\hat{X}(k)$ is the magnitude of the test spectra at frequency bin k.

PSD is a metric designed to quantify spectral differences between binaural audio signals by mimicking human auditory perception. It measures the perceptual dissimilarity between the reference and test signals

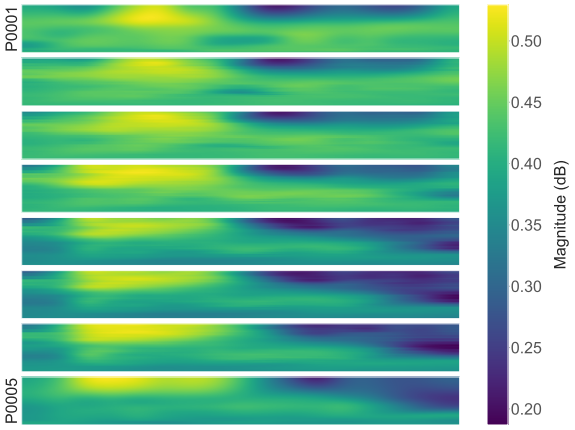


Fig. 4: DSTF feature interpolation between subjects P0001 and P0005 using the proposed model. The top image shows the DSTF feature of subject P0001, and the bottom image shows that of P0005. Intermediate features are generated by the model using linear interpolation between latent embeddings of the two subjects. The x-axis represents frequency (0 Hz to 24 kHz), and the y-axis represents elevation (-60 up to 60°).

by comparing their spectral envelopes after processing through an auditory-inspired model that accounts for frequency selectivity and binaural hearing effect [20]. This makes PSD sensitive to perceptually relevant spectral changes rather than simple numerical differences.

The LSD and PSD histograms for the entire database are shown in Figure 3. For 95.5% of subjects, the LSD values are lower than 0.775, and the PSD values are below 0.178.

In terms of synthesising novel DSTFs, Figure 4 illustrates an indicative example where such features are synthesised through linear interpolation between subjects P0001 and P0005. By leveraging the learned latent space, a range of novel DSTFs can be generated that combine DSTF features from multiple subjects within the database, showcasing the model's ability to generate new location-agnostic perceptual characteristics beyond the original data.

5.2 Latent space

The latent space is designed to capture the perceptual characteristics of each subject's HRTF while discarding azimuthal cues. To visualize the structure of this space,

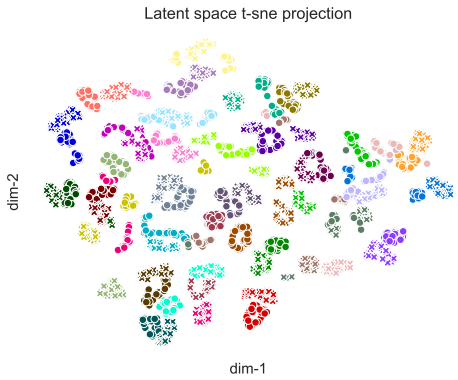


Fig. 5: Latent space visualisation using T-SNE dimensionality reduction. A subset of the subjects included in the database is depicted. Each point represents the latent vector of an DSTF feature at a specific azimuth. Different colours represent different subjects. Circles denote that a feature belongs to the left ear of a subject and crosses denote that a feature belongs to the right ear of a subject.

we apply T-SNE [21] to reduce the 64-dimensional latent vectors to two dimensions, as shown in Figure 5. Each point corresponds to the latent embedding of a DSTF feature measured at a specific azimuth. Colours indicate subject identity, while marker shape distinguishes ear: circles represent left-ear features, and crosses represent right-ear features.

The visualization reveals that features from the same subject are grouped into well-defined clusters, demonstrating that the model effectively encodes unique subject-specific perceptual characteristics while suppressing azimuth-related information. Interestingly, although left and right ears are treated as separate classes during training, the latent vectors for both ears of a subject tend to be clustered close together. This indicates that the unique perceptual characteristics of a subject's HRTF are similar for both ears, and the model is able to capture these relationships, even though it is not explicitly guided to do so. These observations support the claim that the encoder produces a latent representation that preserves subject identity while discarding spatial direction cues, a key goal of our adversarial training strategy.

5.3 Azimuth and DSTF disentanglement

Since DSTF features are calculated at specific azimuth locations, they inherently encode spatial information. To achieve independent control over localisation when synthesizing novel DSTF filters from the autoencoder's latent space, the model must disentangle azimuth qualities from the given features. The Encoder module must discard any correlation with azimuthal information, while the Decoder must rely solely on azimuth conditioning to synthesize a DSTF filter at the desired azimuth. This is accomplished by training the model in a multi-task learning framework, as described in Equation 2, and using adversarial learning for azimuth discrimination, as detailed in Section 4.3.

As outlined in Section 4.1, our dataset consists of 48 unique azimuths, and thus a complete DSTF is composed of 48 features (or DSTF filters). Given that our model can successfully disentangle DSTF and azimuth, a single vector sampled from any point in the latent space is sufficient to generate features at all 48 azimuths around the head. This can be achieved by feeding a latent vector to the Decoder along with the desired azimuth location and repeating this process for all 48 azimuths while keeping the latent vector fixed.

Although reconstructing an existing DSTF feature requires a specific latent vector, synthesising novel DSTFs does not impose this restriction. While azimuthal information is discarded from the latent space, some elevation-specific information remains, as it is encoded within the DSTF features themselves. The further disentanglement of elevation is left for future work.

5.4 Neural compression

The learned DSTF space can be utilised as a compressed representation of a predefined selection of DSTF profiles. An example database of 100 profiles, 1000 measurements per profile and 256-sample stereo filters per measurement with 32-bit floating point precision would require approximately 204.8 MB of disk space. The components required for real-time use of our model are the Decoder module (2.60 MB) and the latent space encodings (2.46 MB). In this example we offer a x40 compression ratio.

6 Conclusion

This work addresses the sensitive and often-overlooked issue of sound coloration introduced by 3D audio processing, advocating for greater emphasis on perceptual modeling in spatial audio, beyond localization accuracy alone. While the boundary between localisation and spectral colouration is inherently blurred, our findings suggest that HRTFs contain rich, underexplored features unrelated to spatial localisation that remain an open area for research.

We introduce a novel signal processing method for extracting Delta Spectral Transfer Function (DSTF) features from HRTFs, and present a conditional autoencoder that compresses and reconstructs these features as well as synthesising novel ones. Through multitask and adversarial training, our model successfully disentangles azimuth information from the perceptual characteristics encoded in DSTFs, enabling independent control of spatial localisation during synthesis. This disentanglement leads to a more structured latent space, where DSTF features from the same subject are consistently clustered, demonstrating the model's ability to preserve subject-specific perceptual characteristics. These results prove the potential of DSTFs as a perceptually meaningful representation and offer a step towards more customisable and perceptually optimised 3D audio experiences.

7 Acknowledgment

This research was conducted during an internship at PlayStation London and is supported by the EPSRC UKRI Centre for Doctoral Training in Artificial Intelligence and Music (Grant no. EP/S022694/1).

References

[1] Armstrong, C., Thresh, L., Murphy, D., and Kearney, G., "A Perceptual Evaluation of Individual and Non-Individual HRTFs: A Case Study of the SADIE II Database," *Applied Sciences*, 8(11), 2018, ISSN 2076-3417, doi:10.3390/app8112029.

[2] Armstrong, C., Chadwick, A., Thresh, L., Murphy, D., and Kearney, G., "Simultaneous HRTF Measurement of Multiple Source Configurations Utilizing Semi-Permanent Structural Mounts," in *Audio Engineering Society Convention 143*, 2017.

[3] Engel, I., Daugintis, R., Vicente, T., Hogg, A. O. T., Pauwels, J., Tournier, A. J., and Picinali, L., "The SONICOM HRTF Dataset," *J. Audio Eng. Soc*, 71(5), pp. 241–253, 2023.

[4] Simon, L. S. R., Zacharov, N., and Katz, B. F. G., "Perceptual attributes for the comparison of head-related transfer functions," *The Journal of the Acoustical Society of America*, 140(5), pp. 3623–3632, 2016, ISSN 0001-4966, doi:10.1121/1.4966115.

[5] Lindau, A., "Spatial Audio Quality Inventory (SAQI). Test Manual. v1.2," *Acta Acustica united with Acustica*, 2014, doi:https://doi.org/10.3813/AAA.918778.

[6] Marentakis, G. and Saitis, C., "Timbre in Binaural Listening: A Comparison of Timbre Descriptors in Anechoic and HRTF Filtered Orchestral Sounds," in *Forum Acusticum*, pp. 1047–1052, Lyon, France, 2020, doi:10.48465/fa.2020.0992.

[7] Marggraf-Turley, N., Lovedee-Turner, M., and De Sena, E., "HRTF Recommendation Based on the Predicted Binaural Colouration Model," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1106–1110, 2024, doi:10.1109/ICASSP48485.2024.10448092.

[8] McKenzie, T., Armstrong, C., Ward, L., Murphy, D. T., and Kearney, G., "Predicting the Colouration between Binaural Signals," *Applied Sciences*, 12(5), 2022, ISSN 2076-3417, doi:10.3390/app12052441.

[9] Esling, P., Chemla-Romeu-Santos, A., and Bitton, A., "Generative timbre spaces with variational audio synthesis," in *Proceeedings of the 21st International Conference on Digital Audio Effects (DAFx-18)*, 2018.

[10] Colonel, J. and Keene, S., "Conditioning Autoencoder Latent Spaces for Real-Time Timbre Interpolation and Synthesis," in *Proceeedings of the International Joint Conference on Neural Networks (IJCNN)*, pp. 1–7, 2020, doi:10.1109/IJCNN48605.2020.9207666.

[11] Puche, A. V. and Lee, S., "Caesynth: Real-Time Timbre Interpolation and Pitch Control with Conditional Autoencoders," *2021 IEEE 31st Interna-*

tional Workshop on Machine Learning for Signal Processing (MLSP), pp. 1–6, 2021.

[12] Hawley, S. H. and Steinmetz, C. J., "Leveraging Neural Representations for Audio Manipulation," in *Audio Engineering Society Europe Convention*, 2023.

[13] European Broadcasting Union, "Loudness normalisation and permitted maximum level of audio signals," EBU Recommendation R 128, 2023, version published 21 Nov 2023.

[14] Raykar, V. C., Duraiswami, R., and Yegnanarayana, B., "Extracting the frequencies of the pinna spectral notches in measured head related impulse responses," *The Journal of the Acoustical Society of America*, 118(1), pp. 364–374, 2005, ISSN 0001-4966, doi:10.1121/1.1923368.

[15] Spagnol, S. and Avanzini, F., "Frequency Estimation Of The First Pinna Notch In Head-Related Transfer Functions With A Linear Anthropometric Model," in *Proceeedings of the 18th International Conference on Digital Audio Effects (DAFx-15)*, 2015.

[16] Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, 17, pp. 261–272, 2020, doi: 10.1038/s41592-019-0686-2.

[17] Fritsch, F. N. and Butland, J., "A Method for Constructing Local Monotone Piecewise Cubic Interpolants," *SIAM J. Sci. Stat. Comput.*, 5(2), p. 300–304, 1984, ISSN 0196-5204, doi:10.1137/0905021.

[18] Gray, A. H. and Markel, J., "Distance measures for speech processing," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 24, pp. 380–391, 1976.

[19] Lee, G. W. and Kim, H. K., "Personalized HRTF Modeling Based on Deep Neural Network Using Anthropometric Measurements and Images of the Ear," *Applied Sciences*, 8(11), 2018, ISSN 2076-3417, doi:10.3390/app8112180.

[20] Armstrong, C., McKenzie, T., Murphy, D., and Kearney, G., "A Perceptual Spectral Difference Model for Binaural Signals," in *Audio Engineering Society Convention 145*, 2018.

[21] van der Maaten, L. and Hinton, G., "Visualizing High-Dimensional Data Using t-SNE," *Journal of Machine Learning Research*, 9(nov), pp. 2579–2605, 2008, ISSN 1532-4435, pagination: 27.

Audio Engineering Society

Conference Paper 5

Presented at the AES International Conference on Machine Learning and Artificial Intelligence for Audio 2025 September 8–10, London, UK

This paper was peer-reviewed as a complete manuscript for presentation at this conference. This paper is available in the AES E-Library (http://www.aes.org/e-lib), all rights reserved. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Journal of the Audio Engineering Society.

Hybrid Learning-based Active Noise Control in Encapsulated Structures

Alkahf Aboutiman[1], Hamid Reza Karimi[1], and Francesco Ripamonti[1]

[1]*Politecnico di Milano, Department of Mechanical Engineering, Milan, Italy*

Correspondence should be addressed to Alkahf Aboutiman (alkahfahmat.aboutiman@polimi.it)

ABSTRACT

Active noise control (ANC) in enclosed environments such as vehicle cabins presents substantial challenges due to complex acoustic paths. To address this, a hybrid ANC strategy is proposed, combining adaptive filtering with deep learning techniques. A finite element model (FEM) was developed to simulate the vibroacoustic dynamics of a closed test structure referred to as the "Noise Box" which reproduces the acoustic conditions inside a car. This model highlights the presence of non-minimum phase characteristics, which complicate control by introducing delays, undershoots, or initial noise amplification before attenuation occurs. To improve control performance under these conditions, a two-dimensional convolutional neural network (2D CNN) is used to select the most appropriate pre-trained control filter based on the spectral features of the incoming noise. Experimental results show that this learning-based hybrid ANC approach achieves superior noise reduction compared to conventional adaptive algorithms.

1 Introduction

Encapsulated environments such as trains, cars, and airplanes can significantly benefit from active noise control (ANC) systems, which provide an advanced solution for reducing unwanted sound [1, 2]. These systems operate by detecting noise through strategically placed sensors and generating anti-noise signals via speakers, resulting in effective noise cancellation in targeted areas. This technology enhances passenger comfort by mitigating the disturbances caused by external and internal sound sources. However, implementing ANC in such environments presents several challenges, particularly in maintaining high performance across a broad frequency spectrum. Addressing these challenges requires sophisticated techniques, including precise acoustic modeling, advanced path identification, and adaptive control algorithms [3, 4].

A fundamental obstacle to effective ANC is the intrinsic behavior of the control environment. In systems exhibiting non-minimum phase characteristics, the initial control signal may unintentionally induce undershoot, phase shift or delay in the control. This phenomenon complicates the convergence of adaptive algorithms and imposes fundamental limitations on overall system performance [5, 6]. Additionally, complex acoustic paths and resonances within the structure can introduce further constraints on the effectiveness of traditional control techniques, necessitating the development of more robust solutions [7].

This study introduces an ANC algorithm specifically

designed for an encapsulated structure referred to as the "noise box," which replicates the acoustic properties of a car interior. The proposed algorithm, SFANC-NLMS, builds upon previous research to enhance noise reduction efficiency in challenging acoustic environments. Inspired by [8], it incorporates a lightweight two-dimensional convolutional neural network (2D CNN) running on a co-processor to select the most appropriate pre-trained control filter for different noise types. Concurrently, the NLMS algorithm continuously updates the filter coefficients at the sampling rate for each noise frame, ensuring real-time adaptability. This hybrid approach enables the system to maintain effective noise suppression despite the complexity of the control environment.

2 Noise Box FEM model

2.1 Model and geometry

In the finite element model, a cavity was selected to simulate the interior of a car, referred to as the "Noise Box." To replicate the sound field within this enclosed structure, modeling both the internal and external fields was avoided due to computational cost constraints. Instead, only the air cavity and the vibrating part of the plate are modeled. The cavity takes the form of a pentagonal prism, with an additional compartment where the plate is located. Its dimensions are $0.825\,m \times 1.018\,m \times 0.645\,m$. The cavity is closed off with a rectangular aluminum plate, measuring $0.825\,m \times 1.018\,m$, and the physical properties of the plate are provided in Table 1.

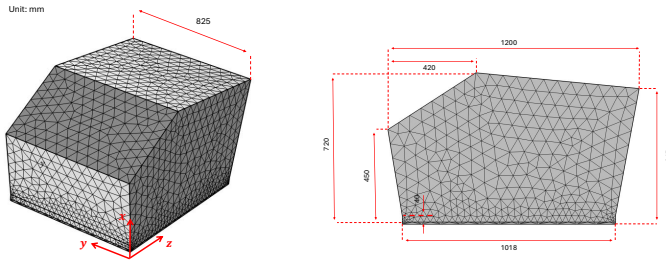
Table 1: Physical properties of the plate material

Property	Value
Thickness, t (m)	0.0004
Density, ρ (kg/m^3)	2700
Modulus of elasticity, E (Pa)	70e9
Poisson's ratio, ν	0.33

Figure 1 shows the dimensions of the noise box used in the model. COMSOL Multiphysics® was used for finite element modeling [9].

2.2 Physics definition

The aluminum plate was modeled in 3D using the Shell physics module, with quadratic shell elements by default. These elements follow the Mindlin-Reissner



(a) (b)

Fig. 1: Geometry of the Noise Box FEM models in (a) 3D and (b) 2D.

theory, and the interpolation method chosen is MITC (Mixed Interpolation of Tensorial Components), which allows for modeling both thin (Kirchhoff theory) and thick (Mindlin theory) shells. Given the thinness of the plate, with a thickness-to-span ratio of $h/a = 0.0048$, much smaller than 1/20 and less than 1/200, Kirchhoff's theory is most appropriate for this study, as suggested in the literature [10, 11].

The shell elements possess three degrees of freedom (DOF) in the x, y, and z directions. For damping, an isotropic structural loss factor $\eta = 0.003$ was selected, based on median values for aluminum plates from [12]. The edges of the plate are fixed to the cavity.

For meshing, triangular elements are applied to the plate with a maximum size of 0.024 m. The air cavity is meshed with tetrahedral elements, where the maximum element size, λ_{max}, is determined by the following equation:

$$\lambda_{max} = \frac{c}{N \cdot f_{max}} \tag{1}$$

Here, $c = 343\,m/s$ is the speed of sound, and $f_{max} = 1000\,Hz$ is the maximum frequency to be solved. N represents the number of mesh elements per wavelength, and it is recommended to use 7 to 10 elements per wavelength [13]. This vibroacoustic problem falls under multiphysics studies, requiring acoustic-structure coupling due to the interaction between the structural plate and the acoustic cavity. The cavity is modeled acoustically using the "Acoustics Frequency Domain" study. The air properties are set by $\rho_0 = 1.2043\,kg/m^3$ and $c = 343.2\,m/s$ at 20°C, assuming static pressure remains constant at 1 atm. The surrounding walls are

considered acoustically rigid, with $\bar{v}_n = 0$. To simulate a reverberation room, impedance is defined by $Z_0 = 100 \times \rho_0 \times c$.

2.3 Computation of the frequency response function

To determine these paths, the frequency response of the Noise Box is extracted for three distinct paths. The disturbance is modeled as a plane wave, applying its corresponding acoustic pressure to the aluminum plate. This pressure is the sum of the incident wave pressure p_{inc} and the reflected pressure p_{refl}, as defined in equations 2, 3, and 4.

$$p_{inc} = Ae^{i\phi} e^{-i(k_x x + k_y y + k_z z)} e^{i\Phi} \tag{2}$$

$$p_{refl} = Ae^{i\phi} e^{-i(-k_x x + k_y y + k_z z)} e^{i\Phi} \tag{3}$$

$$p_{total} = p_{inc} + p_{refl} \tag{4}$$

Here, k_x, k_y, k_z are the wave numbers, while Φ denotes the wave's phase. The angles θ and ϕ, which define the propagation direction, are both equal to $\pi/2$. The frequency response computation begins with a Single-Input-Single-Output (SISO) system. Sensors are modeled as 'Point Probes,' each measuring different physical entities: a reference accelerometer, an error microphone, and a canceling loudspeaker. The finite element model developed in previous section is used to compute the frequency response functions (FRFs) of all control paths in the active noise control (ANC) algorithm. These paths include the reference path, the primary path, and the secondary path in our case.

- **Reference Path:** The frequency response is the ratio of the incident acoustic pressure (Pa) to the plate's normal acceleration (m/s^2), yielding units of $(m/s^2)/Pa$.
- **Primary Path:** The response is the ratio of the incident acoustic pressure to the error microphone's measured pressure (Pa), making it dimensionless.
- **Secondary Path:** The canceling loudspeaker is modeled as a monopole with a volume flow rate of $1\,m^3/s$. Its unit is $Pa/m^3 \cdot s$.

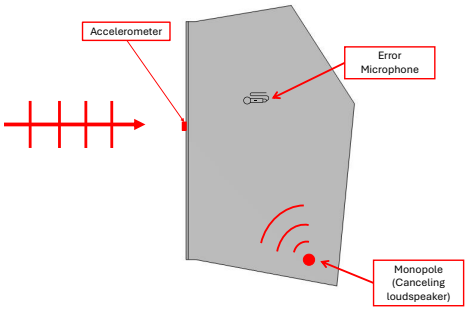


Fig. 2: A plane wave diagram represents the external disturbance entering the cavity, while an internal monopole simulates the canceling loudspeaker for the secondary path.

This setup allows for the analysis of the system's frequency response in noise control applications. Figure 2 presents the method used to compute theses FRFs.

Accurate FRF computation is crucial for deriving the state-space model required for control implementation. The transfer function is estimated using a residue-pole representation:

$$H(s) = \sum_n \frac{C_n}{s - A_n} + D \tag{5}$$

where $s = i\omega f$, C_n are residues, A_n are poles, and D is the direct term. The coefficients are approximated to match the computed frequency response using the AAA (Adaptive Antoulas-Anderson) algorithm [14]. The state-space model is first derived in the continuous domain and then discretized using the Zero-Order Hold (ZOH) method, assuming constant input between sampling intervals. This conversion, performed at 16 kHz, enables digital control design [15]. The resulting FRFs are shown in Figures 3, 4, and 5.

3 The SFANC-NLMS Algorithm

3.1 Presentation of the algorithm

The Selective Fixed-Filter Active Noise Control (SFANC) method enhances noise cancellation by pre-training a database of control filters and selecting the most suitable one in real-time [16]. It operates in two

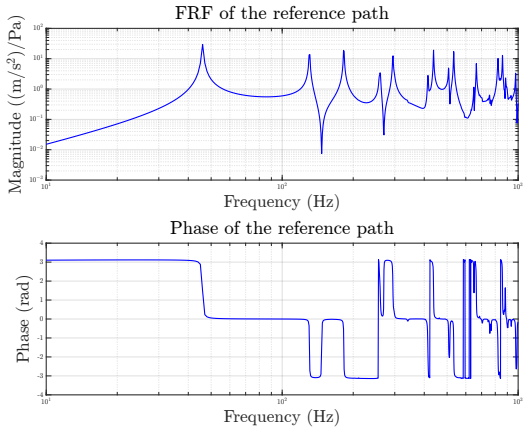


Fig. 3: Frequency response of the reference path

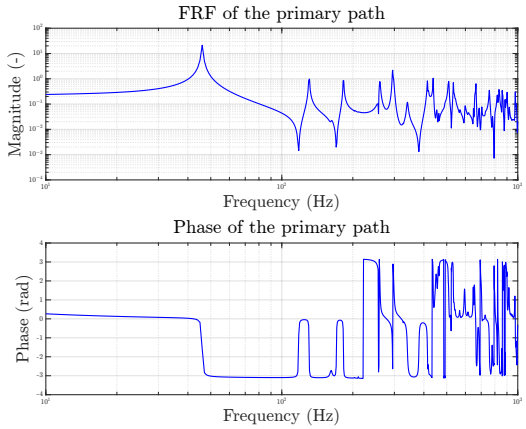


Fig. 4: Frequency response of the primary path

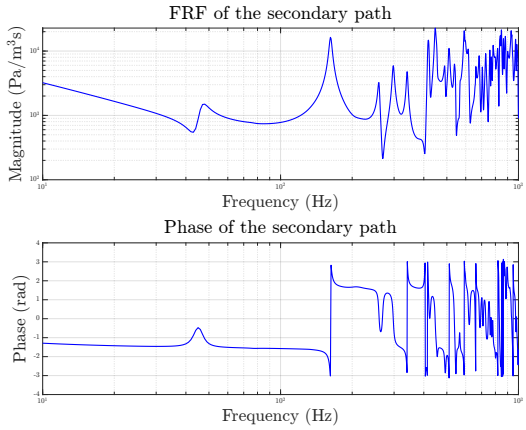


Fig. 5: Frequency response of the secondary path

stages: offline pre-training and online control. During pre-training, filtered white noise samples covering various frequency ranges (20 Hz–1000 Hz) are used to generate optimal control filters via the Normalized Least Mean Square (NLMS) algorithm, as shown in Table 2. These filters are stored in a database.

Table 2: Frequency ranges for each control filter C_i.

Control filter (C_i)	Frequency Range (Hz)
C_1	20 – 1000
C_2	20 – 500
C_3	500 – 1000
C_4	20 – 250
C_5	250 – 500
C_6	500 – 750
C_7	750 – 1000

During online operation, the system extracts noise characteristics and selects the optimal filter from the database, eliminating the need for continuous adaptation. This reduces computational complexity and enhances response time. SFANC-NLMS combines SFANC with adaptive filtering to refine the selected filter in real time. A co-processor uses a convolutional neural network (CNN) to analyze noise and choose the best filter, while an adaptive controller fine-tunes it dynamically. This hybrid approach ensures efficient and responsive noise reduction, making it suitable for real-time applications in dynamic environments. Figure 6 presents the algorithm diagram.

The refinement process is handled by the Normalized Least Mean Square (NLMS) algorithm, which updates the adaptive filter coefficients to minimize the residual error between the desired signal and the output. The NLMS algorithm uses the reference signal $r(n)$, and the error signal $e(n)$ to adjust the filter $w(n)$ through a normalized gradient descent. This update rule ensures stability even when the input signal power varies, making it effective in dynamic acoustic environments. The key equations governing the NLMS update are:

$$e(n) = d(n) - s(n) * \left(w^T(n) r(n)\right) \tag{6}$$

$$y(n) = w^T(n) r(n) \tag{7}$$

$$w(n+1) = w(n) + \mu\, e(n)\, \frac{r(n)}{r^T(n) r(n) + \varepsilon} \tag{8}$$

$$0 < \mu < \frac{1}{\lambda_{\max} D_s} \tag{9}$$

where μ is the step size, ε is a small regularization constant, D_s is the group delay of the secondary path, and $\lambda_{\max}$ is the maximum eigenvalue of the autocorrelation matrix of the reference signal. $d(n)$ is the primary noise and $s(n)$ is the secondary path impulse response.

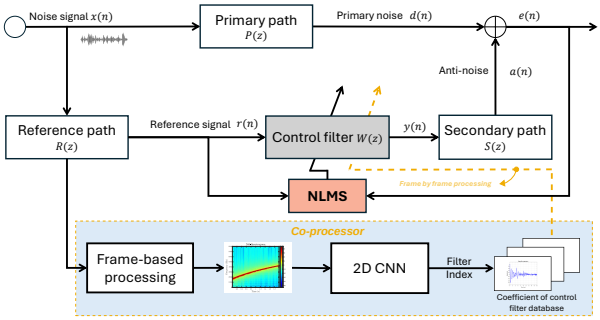


Fig. 6: Block diagram of the hybrid SFANC-NLMS algorithm. For each frame, the 2D CNN running on a co-processor selects the most appropriate pre-trained control filter. The NLMS algorithm then refines and updates the coefficients of the selected filter in real time at the sampling rate.

3.2 2D Convolutional Neural Network Architecture

The model architecture used is a modified version of a 2D CNN presented in [17]. The architecture is optimized to reduce the number of parameters by adjusting the convolutional layers and removing the Softmax layer in favor of direct classification. The detailed architecture of the 2D CNN is provided in Table 3. The input to the network consists of Mel spectrograms computed from audio signals with a sampling rate of 16 kHz. Each spectrogram is generated using a Fast Fourier Transform (FFT) size of 1024, a hop length of 512 , and a number of Mel filter banks set to 64. This results in spectrograms with a fixed frequency resolution of 64 Mel bands. The temporal resolution depends on the duration of the input audio. For a 1-second signal, the spectrogram has a width of 31 time frames. The network begins with a convolutional layer with a 5×5 kernel, extracting basic spectral patterns and producing 16 feature maps. This is followed by a ReLU activation and a 3×3 max pooling operation. The second convolutional block consists of two 3×3 convolutional layers, increasing the feature maps to 32 while preserving spatial resolution. A 6×6 max pooling operation further reduces the dimensionality. The extracted features are then flattened, batch-normalized, and passed through a fully connected layer of 15 neurons. Finally, a second fully connected layer produces the final classification output.

3.3 Training

To train the 2D CNN model, we generated a dataset of 80,000 Mel spectrograms for training and 2,000 for validation and testing. The spectrograms were computed from broadband noise recordings sampled at 16 kHz, filtered through the previously described reference path using randomly varying frequency bands. Each spectrogram corresponds to a 1-second noise segment, resulting in an input size of $1 \times 64 \times 31$. The model was trained using the Adam optimizer with a learning rate of 0.0001, a batch size of 250, and for 30 epochs. The training objective was to minimize the CrossEntropyLoss function. To prevent overfitting, L2 regularization with a factor of 10^{-4} was applied to all trainable parameters. Figure 7 shows the evolution of loss and accuracy during training.

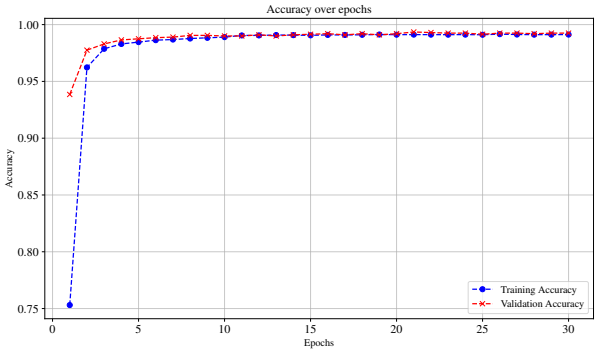
The performance of the proposed 2D CNN was compared against several other 2D network architectures, including the 2D CNN introduced in [17], the ShuffleNet model presented in [18], MobileNet model proposed in [19], and DenseNet presented in [20]. Table 4 summarizes the comparison of these networks. Based on the results, the proposed 2D CNN stands out as the most promising architecture, considering both its accuracy and the number of parameters it requires. To enable this comparison, all architectures were retrained using the constructed dataset.

4 Results

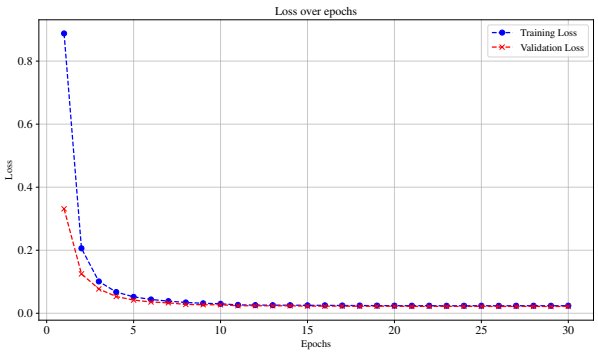
In this section, numerical simulations are performed using a single-channel ANC framework. The system operates at a sampling frequency of 16 kHz, with the control filter consisting of 8192 coefficients. The state-space coefficients introduced in Section 2.3 define the

Table 3: Summary of the proposed 2D CNN network architecture. The input consists of Mel spectrograms with 64 Mel bands and a variable number of time frames, depending on the duration of the input signal. For a 1-second signal.

Layer Index	Layer Type	Configuration	Output Shape
1	Convolutional Layer	Kernel: 5×5, Channels: $1 \rightarrow 16$, Stride: 1, Padding: 2	$[16, 64, 31]$
2	Activation (ReLU)	-	$[16, 64, 31]$
3	Max Pooling	Kernel: 3×3, Stride: 3	$[16, 21, 10]$
4	Convolutional Layer	Kernel: 3×3, Channels: $16 \rightarrow 32$, Stride: 1, Padding: 1	$[32, 21, 10]$
5	Activation (ReLU)	-	$[32, 21, 10]$
6	Convolutional Layer	Kernel: 3×3, Channels: $32 \rightarrow 32$, Stride: 1, Padding: 1	$[32, 21, 10]$
7	Max Pooling	Kernel: 6×6, Stride: 3	$[32, 5, 3]$
8	Flatten	-	$[384]$
9	Batch Normalization	Features: 384	$[384]$
10	Fully Connected Layer	Input: 384, Output: 15	$[15]$
11	Fully Connected Layer	Input: 15, Output: 7	$[7]$



(a) Training Accuracy



(b) Training Loss

Fig. 7: Accuracy and loss curves for the training of the proposed 2D CNN.

reference, primary, and secondary paths. Moreover, a total of seven pre-trained control filters are employed.

Table 4: Test accuracy and parameter comparison for various networks.

Network	Test Accuracy	Parameters
Proposed method	99.35%	20,959
2D CNN [17]	96.30%	25,855
ShuffleNet [18]	98.80%	246,110
MobileNet [19]	97.45%	2,457,721
DenseNet [20]	99.40%	6,954,759

As detailed in Table 2, these filters are obtained by training on seven distinct white noise signals, each constrained to a specific frequency range. The NLMS algorithm is used to compute the optimal control filters for these training noises, with a step size set to 0.1. To assess the performance of the algorithm, broadband noise is utilized as the disturbance signal. Stationary noise is modeled as band-limited random noise spanning 20 to 450 Hz. The SFANC-NLMS method is benchmarked against both the NLMS and SFANC algorithms.

The stationary noise is generated as a zero-mean, unit-variance Gaussian process. This signal undergoes band-pass filtering between 20 Hz and 450 Hz, using a filter designed with a Hamming window. The window characteristics determine the passband ripple and stopband attenuation, ensuring precise frequency selection. The sampling frequency is maintained at 16 kHz across all simulations, while the control filter length remains fixed at 8192 taps. The simulations are conducted on 10-second-long signals. The results obtained under

these conditions are illustrated in Figure 8, depicting the time-domain response of the controlled signals.

Examining the results depicted in Figure 8 for the different algorithms, we observe that for broadband noise, SFANC exhibits superior performance compared to NLMS. This enhanced effectiveness significantly influences the overall control achieved with SFANC-NLMS. Notably, SFANC-NLMS achieves a lower error amplitude due to its adaptive structure. Additionally, in the case of SFANC, active control is initiated after approximately 1 second. This delay allows the convolutional neural network (CNN) to analyze the characteristics of the primary noise and determine the most suitable pre-trained controller. Conversely, SFANC-NLMS applies control immediately from the start, leveraging its adaptive capabilities to adjust in real time.

Figure 9 presents the Power Spectral Density (PSD) of each controlled signal. The lowest levels are achieved by the signal controlled with SFANC-NLMS (represented by the red curve). To evaluate its effectiveness, we compare it with the PSD of the signal controlled by SFANC (the yellow curve). These results indicate that SFANC-NLMS consistently outperforms all other algorithms at each resonance frequency of the system. Additionally, at anti-resonance frequencies, such as around 110 Hz or 370 Hz, the signal levels controlled by SFANC and SFANC-NLMS exceed that of the primary noise. This phenomenon, known as the waterbed effect, describes a trade-off in control systems, where reducing energy in certain frequency bands inevitably causes an increase in others. This effect is particularly pronounced in non-minimum phase systems, where the presence of unstable zeros imposes fundamental limitations on the achievable performance of control algorithms.

Figure 10 illustrates the noise reduction achieved over time. At the beginning of the process, the noise reduction remains limited due to the initial response delay induced by the non-minimum phase (NMP) characteristics of the system and the time required by SFANC to initiate control. However, after approximately 2 seconds of computation, optimal control is established.Beyond this point, SFANC initially achieves a greater noise reduction compared to SFANC-NLMS. This discrepancy may stem from variations in the controller coefficients, which could momentarily mislead the adaptive algorithm in minimizing the error. Nevertheless, for the remainder of the signal, SFANC-NLMS provides superior noise reduction. On average, the noise reduction achieved by each algorithm is as follows: 1.66 dB for NLMS, 4.02 dB for SFANC, and 5 dB for SFANC-NLMS.

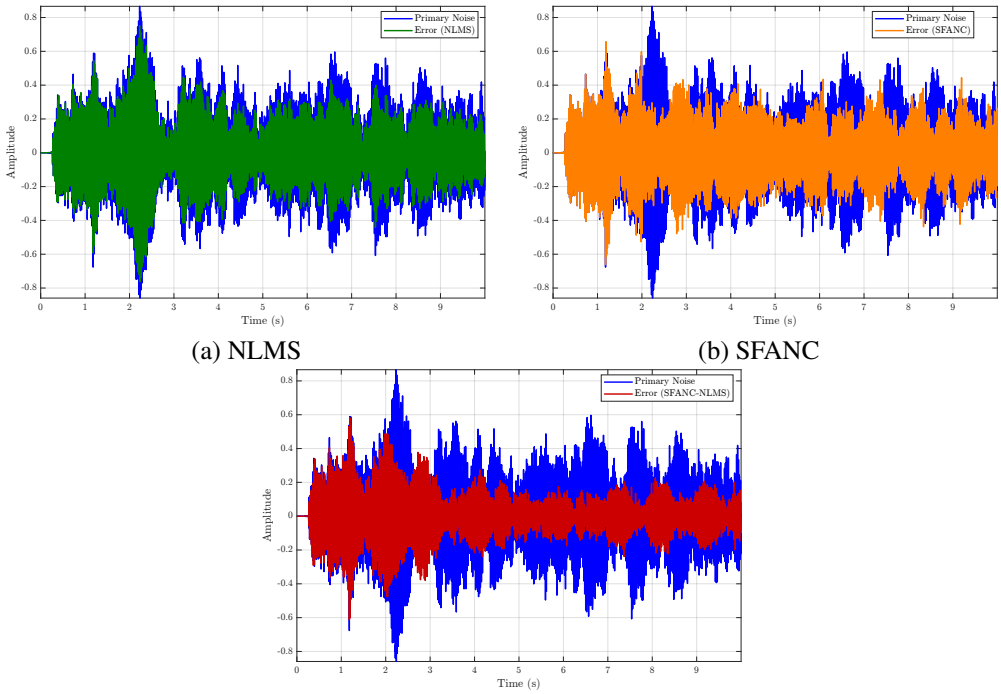
5 Conclusion

This study presents an experimental framework aimed at characterizing the frequency response function of an encapsulated structure, referred to as the Noise Box, which emulates the acoustic properties of a car interior. Given its non-minimum phase behavior, this environment poses significant challenges for active noise control (ANC). The acquired frequency response data were leveraged to develop and implement a novel control strategy. The proposed SFANC-NLMS algorithm integrates deep learning into the ANC framework. Its approach relies on constructing a dataset of multiple pre-trained control filters and employing a 2D convolutional neural network (CNN) to dynamically select the most appropriate controller based on the characteristics of the incoming noise. Additionally, an adaptive algorithm is incorporated to refine the coefficients of the chosen pre-trained controller, ensuring optimal performance in real time. This dual mechanism enhances efficiency beyond conventional adaptive control strategies while mitigating the limitations of SFANC when the primary path undergoes variations over time. A key advantage of the proposed approach is the elimination of secondary path estimation, which traditionally introduces discrepancies between the modeled and actual system dynamics. By bypassing this constraint, SFANC-NLMS reduces computational complexity and demonstrates that, in this specific setup, directly employing NLMS instead of FxNLMS can yield competitive results.

Looking ahead, while deep learning currently plays a supporting role in optimizing the selection of control filters, future advancements could push its involvement further—potentially enabling a neural network to directly synthesize the anti-noise signal. Such a development would be particularly valuable for addressing nonlinear distortions associated with the acoustic environment and electro-acoustic interactions induced by the materials used in noise control implementations.

References

[1] Fuller, C., Hansen, C., and Snyder, S., “Active control of sound radiation from a vibrating rectangular panel by sound sources and vibration inputs:



(a) NLMS (b) SFANC

(c) SFANC-NLMS

Fig. 8: Graphics of the primary noise and error signals obtained using different ANC algorithms.

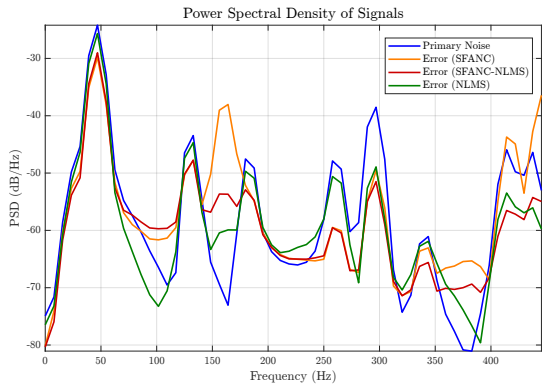


Fig. 9: Power Spectral Density (PSD) of the primary noise and the controlled noise achieved by NLMS, SFANC, and SFANC-NLMS

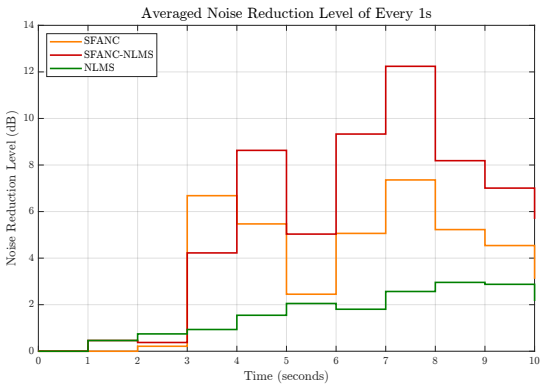


Fig. 10: Average noise reduction each second of the controlled noise using NLMS, SFANC, and SFANC-NLMS.

an experimental comparison," *Journal of Sound and Vibration*, 145(2), pp. 195–215, 1991.

[2] Misol, M., Algermissen, S., and Monner, H. P., "Experimental investigation of different active noise control concepts applied to a passenger car equipped with an active windshield," *Journal of Sound and Vibration*, 331(10), pp. 2209–2219, 2012.

[3] Belgacem, W., Berry, A., and Masson, P., "Active vibration control on a quarter-car for cancellation

of road noise disturbance," *Journal of Sound and Vibration*, 331(14), pp. 3240–3254, 2012.

[4] Aboutiman, A., Rachman, Z., Oberman, T., Aletta, F., Kang, J., Karimi, H. R., and Ripamonti, F., "Subjective perception analysis of active noise control algorithms in an encapsulated structure: An experimental study," *Applied Acoustics*, 239, p. 110823, 2025, ISSN 0003-682X, doi: https://doi.org/10.1016/j.apacoust.2025.110823.

[5] Milani, A. A., Kannan, G., and Panahi, I. M., "On maximum achievable noise reduction in ANC systems," in *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 349–352, IEEE, 2010.

[6] Wu, L., Qiu, X., and Guo, Y., "A generalized leaky FxLMS algorithm for tuning the waterbed effect of feedback active noise control systems," *Mechanical Systems and Signal Processing*, 106, pp. 13–23, 2018.

[7] Aboutiman, A., Shams, R., Karimi, H. R., Ripamonti, F., and Pawełczyk, M., "Active noise control in encapsulated structures with non-minimum phase characteristics using a Kalman filter approach," *Journal of Sound and Vibration*, 615, p. 119187, 2025, ISSN 0022-460X, doi:https://doi.org/10.1016/j.jsv.2025.119187.

[8] Luo, Z., Shi, D., and Gan, W.-S., "A hybrid sfancfxnlms algorithm for active noise control based on deep learning," *IEEE Signal Processing Letters*, 29, pp. 1102–1106, 2022.

[9] Liu, L., *Design, simulation and testing of a coupled plate-cavity system targeted for vehicle interior noise analysis and control*, Ph.D. thesis, Politecnico di Milano, 2021.

[10] Cremer, L. and Heckl, M., *Structure-borne sound: structural vibrations and sound radiation at audio frequencies*, Springer Science & Business Media, 2013.

[11] Chapelle, D. and Bathe, K.-J., *The finite element analysis of shells-fundamentals*, Springer Science & Business Media, 2010.

[12] Lenk, P. and Coult, G., "Damping of glass structures and components," in *Challenging Glass Conference Proceedings*, volume 2, pp. 341–350, 2010.

[13] Astley, R., Macaulay, G., and Coyette, J., "Mapped wave envelope elements for acoustical radiation and scattering," *Journal of Sound and Vibration*, 170(1), pp. 97–118, 1994.

[14] Nakatsukasa, Y., Sète, O., and Trefethen, L. N., "The AAA algorithm for rational approximation," *SIAM Journal on Scientific Computing*, 40(3), pp. A1494–A1522, 2018.

[15] Moir, T. J. and Moir, T. J., *Rudiments of Signal Processing and Systems*, Springer, 2022.

[16] Shi, D., Gan, W.-S., Lam, B., and Wen, S., "Feedforward selective fixed-filter active noise control: Algorithm and implementation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28, pp. 1479–1492, 2020.

[17] Shi, D., Lam, B., Ooi, K., Shen, X., and Gan, W.-S., "Selective fixed-filter active noise control based on convolutional neural network," *Signal Processing*, 190, p. 108317, 2022.

[18] Luo, Z., Shi, D., Ji, J., Shen, X., and Gan, W.-S., "Real-time implementation and explainable AI analysis of delayless CNN-based selective fixed-filter active noise control," *Mechanical Systems and Signal Processing*, 214, p. 111364, 2024.

[19] Adapa, S., "Urban sound tagging using convolutional neural networks," *arXiv preprint arXiv:1909.12699*, 2019.

[20] Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q., "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Audio Engineering Society

Conference Paper 6

Presented at the AES International Conference on Machine Learning and Artificial Intelligence for Audio 2025 September 8–10, London, UK

This paper was peer-reviewed as a complete manuscript for presentation at this conference. This paper is available in the AES E-Library (http://www.aes.org/e-lib), all rights reserved. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Journal of the Audio Engineering Society.

Flute Tone Quality Classification: a Machine-Learning-Based Evaluation Tool

Nikita Sane[1] and Jonathan Abel[2]

[1]*Henry M. Gunn High School*
[2]*CCRMA, Stanford University*

Correspondence should be addressed to Nikita Sane (nikitassane@gmail.com)

ABSTRACT

Music classification models have been used to accurately place audio recordings into different classes based on genre, instrument, and era, among other features. In this work, we introduce a machine learning algorithm that classifies flute recordings based on their tone quality. Flute tone quality is subjective, and its underlying parameters currently lack clear definitions in terms of objective audio signal processing features. Here we consider two quality descriptors, "focus" and "brightness," and present a largely "black box" algorithm to categorize recorded flute notes into quality descriptor classes. The model was built by first gathering and labeling a set of roughly 5000 high-quality flute note recordings derived from the Freesound database according to three levels each of focus and brightness. Then a pre-processing script that performs spectral analysis was developed. Finally, a convolutional neural network was trained, and its hyper-parameters, including the learning rate, batch size, and dropout rates, adjusted. An overall classification rate of 77% was achieved, with brightness and focus being correctly classified at rates of 83% and 86%, respectively.

1 Introduction

1.1 Background

Flute playing can be split into six major modalities: rhythm, articulation, dynamics, tone quality, intonation, and musicality [1]. Of these, the most subjective measures are musicality and tone quality (both go beyond the notes and symbols on the sheet music). In music auditions and competitions, judges make subjective, but still informed, decisions that may be subconsciously biased. To curb this bias and supplement decision-making, we propose creating an objective technology-based tool that rates a player's flute audition. We focused specifically on labeling tone quality as it is a playing modality that is highly nuanced.

We anticipate that our technology tool will be beneficial for not only audition judges but also young players, especially those without access to quality flute instruction. Learning what different tone qualities sound like is a crucial step in the early stages of learning the flute. With a tool that provides feedback on an audio recording, young players can develop their musical ear to hear when their tone is optimal.

Flutists typically change their tone quality by manipulating air speed, air direction, and the shape formed inside the mouth. Slower air speeds often result in

less bright tones. Larger mouth cavities result in fuller sounds. The specific manipulation of the embouchure (lip and mouth shape) depends on which register the flutist plays in.

Classification problems in the field of machine learning for audio analysis often use a convolutional neural network (CNN) fed with Mel Frequency Cepstral Coefficient data for each sample [2]. We used the basic model architecture of a music genre classifier [3] to begin classifying flute tone quality.

1.2 Related Work

Nykänen and Johansson [4] establish a connection between embouchure/mouth shapes and tone quality manipulation in saxophones, and this relationship is present for flute players too [5].

In the current field of music information retrieval, a more implicit approach is often used to analyze the spectral envelopes of different flute tone qualities. For instance, Clements and Yorita [6] explored the correlation between harmonic content and perceived flute tone quality, suggesting that specific spectral profiles might be indicative of certain tonal characteristics. They used descriptors such as "rich" or "dark" identified by expert flutists.

Advancements in machine learning have introduced more sophisticated approaches to this domain. Hernandez-Olivan and Beltran [7] developed a deep learning model employing a multi-head attention mechanism to classify timbres of orchestral instruments, achieving 62% classification accuracies across twenty instrument classes.

Similarly, recent studies by Agarwal and Greer [8] have used spectrogram-based deep learning and large language models to assess recorded flute performances. Their model is trained on labeled audition data and predicts category-specific scores (such as tone, intonation, and rhythm) that closely align with human adjudicators' assessments. CNNs specifically have been applied to optimizing flute playing and teaching systems [9]. By analyzing audio recordings, these models aim to strengthen students' musical literacy and performance skills.

Beyond the flute, Giraldo and Ramirez [10] have developed a system for users to train their own machine learning models that evaluate the tone quality of violin performances. Using a dataset of annotated violin recordings, they correlate different perceptual audio descriptors to build a real-time feedback system.

While the domains of instrument classification and performance assessment have seen significant progress, the subjective nature of tone quality, especially for instruments such as the flute, presents ongoing challenges. Our work builds upon these foundations. By using a labeled dataset and focusing on perceptual descriptors such as "brightness" and "focus," we aim to provide a more nuanced and objective evaluation tool for flutists and educators.

2 Methods

2.1 Dataset Labeling Process

We used the Freesound library [11] to train the proposed machine learning model. The library is a collaborative database of over 600,000 sounds uploaded by musicians, researchers, and audio engineers. The data we downloaded consisted of two packs labeled “Flute scales” and “Flute scales (played badly)” containing 76 and 281 audio files respectively. These packs were recorded by the Music Technology Group at the University of Pompeu Fabra in the context of their goodsounds.org project [12]. All of the audio files ranged from 10 to 30 seconds in length and were sampled at 48 kHz.

Note that we use “clips” and “recordings” interchangeably in this section. To label the data, ten flutists of high-intermediate to pre-professional level were selected. The 357 total audio clips were split into ten groups of thirty recordings each by simple random sampling. Then, the ten groups were further split into five pairs, and ten to twelve additional audio clips were given to both raters of each pair allowing for some overlap between assigned recordings. A total of 300 clips were be rated by one flutist, and fifty-seven clips were rated by two. Figure 1 shows a visual representation of this process.

We selected two descriptors of flute tone on which to rate each clip: “brightness” and “focus.” Brighter tones have dominant upper overtones and are rich in harmonics. Musicians often control the brightness of their tone to fit a certain part of the music. Focused tones are characterized by a stronger sinusoidal component of the sounds compared to the residual noise-like component. Musicians almost always strive for focused, fuller

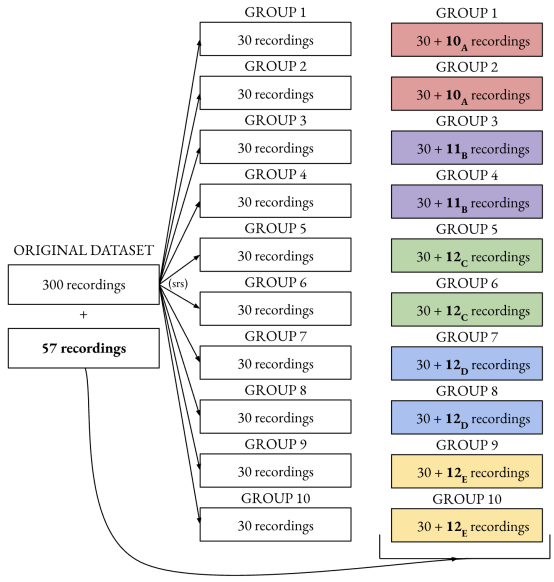


Fig. 1: Data-separation process used to assign audio clips to raters.

tones. Below are some words that are closely related to each of our selected descriptors that are sometimes used to describe flute tone quality in the music world.

- Bright: resonant, brilliant (antonyms: dark, dull)
- Focused: full, rich, clear, supported, centered (antonyms: hollow, airy, weak, thin)

Each participating flutist was assigned one group of recordings to rate. Each descriptor was rated on a scale of 1 to 3. For example, a clip rated "1_bright_3_focus" had a dark and centered tone. We selected these two descriptors because they are sufficiently independent of one another: a tone that is bright could be unfocused or full, and a tone that is full could be brilliant or dull.

Once each clip was rated, it was labeled accordingly. The clips that were rated by two flutists were used to ensure that the ratings were at least fairly consistent across raters. Table 1 shows the distribution of final labels after rating for the audio dataset. From the distribution of labels, we see it was common to rate brightness and focus as a "2" perhaps because the rating scale lacked granularity.

Table 1 also reveals that the dataset was imbalanced across different rating categories. We used class weighting to address this imbalance. Each label's respective weight was computed to be inversely proportional to its frequency in the dataset.

Table 1: Number of audio clips categorized into each label and their respective class weights.

Label	# of Audio Clips	Class Weight
1_bright_1_focus	51	0.778
1_bright_2_focus	53	0.747
1_bright_3_focus	23	1.718
2_bright_1_focus	37	1.071
2_bright_2_focus	74	0.536
2_bright_3_focus	45	0.881
3_bright_1_focus	14	2.825
3_bright_2_focus	31	1.280
3_bright_3_focus	29	1.364

2.2 Preprocessing

An essential stage of data collection involved cleaning the data. Analyzing the spectrograms of various audio clips, we identified that the power spectrum was highly unstable around the onset of notes and during changes in scale degrees. Thus, we constructed a preprocessing script in Python that split each audio clip at the note changes to create several sub-samples that only contained one pitch.

To write this script, we used Librosa [13] and NumPy [14]. Librosa is a Python library with functions to perform fast fourier transforms, onset detection, spectrogram analysis, etc. on audio files. NumPy is a Python library that supports computations with matrices and large arrays.

The first step of preprocessing was to identify the fundamental frequencies at several time indices along the recording to track when note changes occurred. So, we extracted frequencies and their corresponding magnitudes along the input file using `librosa.piptrack`, a pitch-tracking method. Within the resulting pitches array, we set a lower threshold at 240 Hz for fundamental frequencies, ignoring data outside the modern flute range (246 Hz–2093 Hz). Within the magnitudes array, we set a lower threshold

of 0.6, as the fundamental frequency is the strongest harmonic heard in the power spectrum. Out of the remaining frequencies in each time index, we set the lowest non-zero frequency present as the fundamental. The output matrices of `librosa.piptrack` were thus effectively flattened into a 1D array of fundamental frequencies with each index representing a time frame.

The next step of preprocessing was separating each recording into individual notes. This was achieved by noting the time indices where the fundamental frequency changed by greater than 20 Hz within the length of the audio. We called this new list "pitch_change_times."

Each range of audio data between adjacent time indices in our pitch_change_times list represented a single note within a scale being played in each audio recording. We call these sections of the recordings "samples." From the original dataset of 357 rated recordings, we extracted over 5000 labeled samples; each recording featured 16 notes of an ascending and descending one-octave scale.

Once the data was labeled and processed, we extracted discrete features from each audio sample. We used two approaches, one based on note MFCCs and partial amplitudes, and the other using note spectra. The following two sections describe the features computed as inputs for our machine learning model.

2.3 Method 1: Mel-Frequency Cepstral Coefficients (MFCCs) and Partial Amplitudes

Mel-Frequency Cepstral Coefficients (MFCCs) [15] are compact representations of the audio signal's spectral properties, obtained by applying a Discrete Cosine Transform (DCT) to the log-scaled Mel spectrogram, making them useful for speech and audio analysis. Equation (1) is used by the `librosa.feature.mfcc` function.

$$MFCC[m] = \sum_{k=1}^{K} S[k] \cdot \cos\left(m \cdot \left(k - \frac{1}{2}\right) \frac{\pi}{K}\right) \quad (1)$$

Here, $S[k]$ is the log-magnitude Mel spectrogram; K is the number of Mel filterbanks; m is the cepstral coefficient index. The DCT is applied to decorrelate the Mel spectrogram features.

We computed the first thirteen MFCCs for each sample using the `librosa.feature.mfcc` function and saved them along with their original rated labels.

Along with MFCC data, we selected values from the power spectrum for each audio clip, targeting each audio descriptor separately. We noted that a substantial portion of the sense of focus or brightness came from the steady-state middle portion of the note, away from the note onset and note release. Accordingly, in our classifier, we first isolated the middle 500 ms of each sample. We performed a Short-Time Fourier Transform (STFT) on the note segment using the `librosa.stft` function. We used a 2048-point-long Hann window [16], a 4096 bin DFT size, and a 512 sample hop length. Given that the sampling rate of the recordings was 48 kHz, we pulled 44 STFT frames for each sample.

We formed a periodogram spectral estimate for each note by averaging the STFT power over time,

$$X_{RMS}(k) = \sqrt{\frac{1}{44} \sum_{t=1}^{44} |X_k^{(t)}|^2} \quad (2)$$

In (2), $X_k^{(t)}$ is the magnitude of the DFT at frequency bin k and time frame t.

As an example, in Figure 2, two clips are shown, featuring a G4 to G5 major scale, articulated on every note, and with different focus ratings. As can be seen, the recording with lower rated focus has visible sub harmonics, especially in the higher note ranges, and has energy at odd integer multiples of half the fundamental frequency. To capture this observation, we computed the subharmonic frequencies $0.5f_0$, $1.5f_0$, $2.5f_0$, and $3.5f_0$ where f_0 is the fundamental frequency of a given sample as computed during preprocessing. Then, in the root mean square spectrum, we extracted the amplitudes of the nearest frequency bins to the subharmonics. The amplitudes were converted to decibels (dB) and normalized to the frequency of the fundamental before being added to the input data for each sample.

In Figure 3, two clips are shown featuring the same G4 to G5 major scale, articulated on every note, and with different brightness ratings. Note that the recording with higher rated brightness has a higher amplitude in the upper harmonic range for each note. To capture this observation, we computed the amplitudes of the frequency bins closest to the first seven harmonics in the root mean square spectrum of a given sample ($2f_0$, $3f_0$, etc.). These computed amplitudes were converted to dB, normalized to the fundamental frequency amplitude, and added to the input data for each sample.

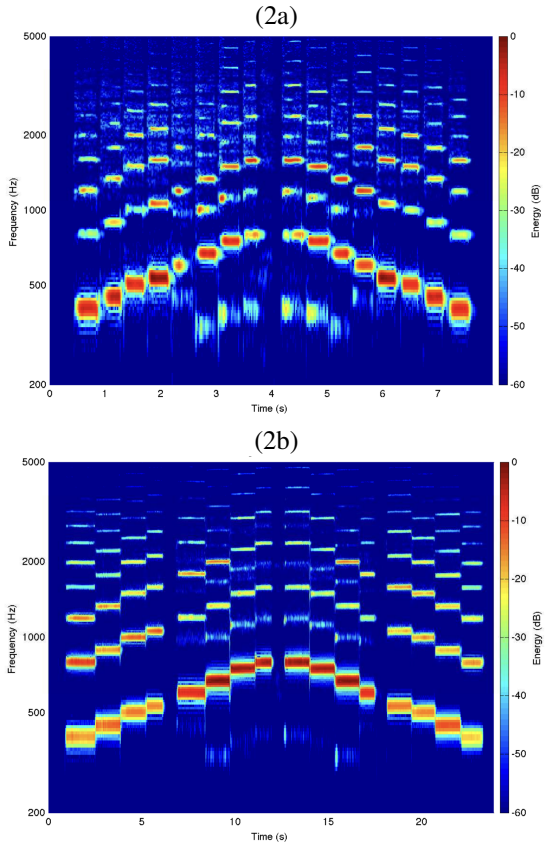


Fig. 2: Two spectrograms for different audio clips. (2a) was computed from a recording rated 2_bright_1_focus, and (2b) was computed from a recording rated 2_bright_3_focus.

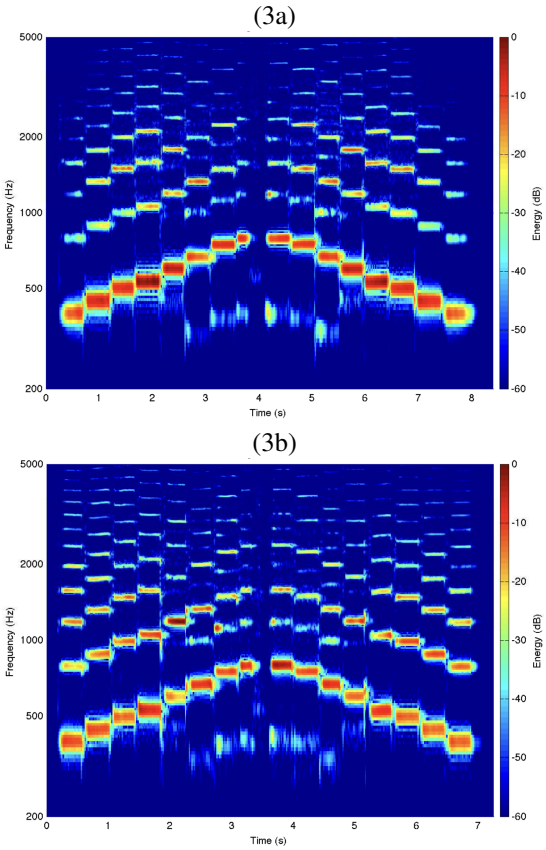


Fig. 3: Two spectrograms for different audio clips. (3a) was computed from a recording rated 1_bright_2_focus, and (3b) was computed from a recording rated 3_bright_2_focus.

The final input prepared for our model using this input data was a `.json` file containing a dictionary with three keys: "mapping," "mfcc + power spectrum data," and "labels." A visual representation of our dictionary is provided in Figure 4.

2.4 Method 2: Full Spectral Data

Method 1 of extracting data to input yielded an average classification rate of 71.9%. Another form of spectral data we tested inputting into the model was the entire root mean square spectrum for each sample instead of selected values. The entire spectrum would provide a more nuanced description of the full sound profile of each sample. For these spectra, we used frequency bins between 80 Hz and 8 kHz so as to capture the fundamental and subharmonic frequencies for all of the tones used, while avoiding any low-frequency noise, and to include high-frequency signal energy that was plainly visible in the note spectra. The spectra were normalized and clipped to range from -60 dB to 0 dB.

The final input prepared for our model using Method 2 was a `.json` file containing a dictionary with three keys: "mapping," "full power spectra," and "labels."

2.5 Basic Model Architecture

Before building our convolutional neural network (CNN), we tested a more simple k-means clustering [17] machine learning model with the Method 1 input data. It did not achieve notable clustering patterns.

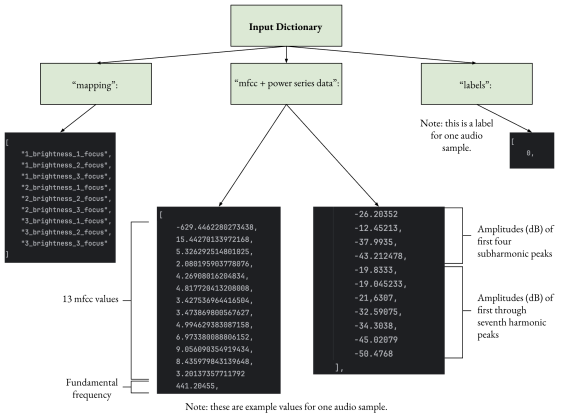


Fig. 4: Visual representation of the input dictionary using Method 1.

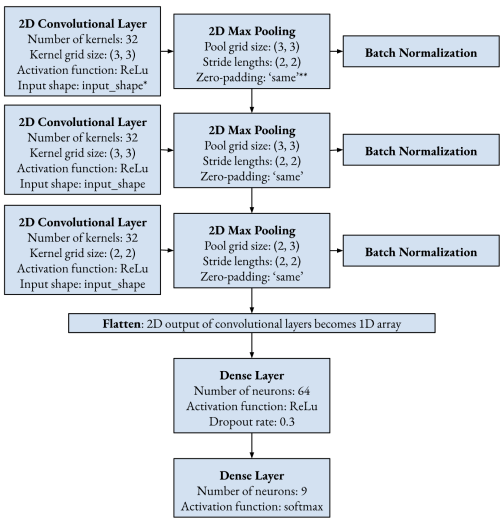


Fig. 5: Visual representation of the basic model architecture.

We used TensorFlow [18]—a software library for machine learning functions—and Keras [19]– a higher-level API for neural networks– to develop our Python model.

A common machine learning model used for genre classification problems is a CNN with three convolution layers and two dense, fully-connected layers. Thus, we began with the same architecture for our model before fine-tuning to improve convergence. For the first four layers, we used the Rectified Linear Unit (ReLU) activation function [20] largely due to its efficient convergence rate in stochastic gradient descent. ReLU returns the input if it is greater than zero, and zero otherwise according to (3) below.

$$f(x) = \max(0, x) \tag{3}$$

We use the softmax activation function in our last layer—a dense layer of nine neurons. The softmax activation function converts a vector of raw scores into a probability distribution by exponentiating each value and normalizing it so that the sum of all outputs equals 1 according to (4) below. Each of the nine neurons in the final layer represented one tone rating label.

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}} \tag{4}$$

In the fourth layer, we used a dropout rate of 0.3 to avoid over-fitting in the training process. A visual representation of the basic model architecture is provided in Figure 5.

For the test-train-validation split, 55% of the input data was training data, 25% was testing data, and 20% was validation data by random splitting performed in TensorFlow.

3 Training Details

We will discuss the effects on train and test set evaluation accuracy from varying the following hyperparameters: optimizer learning rate, number of epochs, batch size, and dropout rate. We varied one hyperparameter at a time and started testing from the following settings: 0.001 learning rate, 30 epochs, batch size of 32, and 0.3 dropout rate. Input data Method 2 of providing full spectra for each sample yielded overall higher classification rates, so we tested hyperparameters using this method.

3.1 Learning Rate

Figure 6 in Appendix A shows that learning rates of 0.01 and 0.001 result in overfitting as well as unstable convergence. Thus, we chose a learning rate of 0.0001.

3.2 Number of Epochs

Figure 7 in Appendix A shows that 15 and 50 epochs result in unstable learning curves and overfitting respectively. We selected 30 epochs for our model.

3.3 Batch Size

Figure 8 in Appendix A shows that a batch size of 32 offers the least overfitting and a proper learning curve.

3.4 Dropout Rate

Figure 9 in Appendix A shows that a dropout rate of 0.3 improves both train and test set accuracy substantially while avoiding overfitting.

4 Results

4.1 Data Note Splitting Strategy Outcome

As described in the Methods section, we expanded our data from 357 audio clips to 5000 samples by rating the clips before splitting the recordings into single-note samples. This strategy helped reduce noisy data from the onsets and tapers of articulated notes as well as breathing in the recordings. It was an effective way to isolate the pure tone of each note within each recording.

4.2 Final Model Parameters, Accuracy Outcomes

Our final model used the keras.Adam [21] optimizer which follows a stochastic gradient descent approach for efficient convergence. The optimizer used a learning rate of 0.0001 and the sparse categorical cross entropy loss function [22] that varies according to (5). Our main motivation for using sparse categorical cross entropy over categorical cross entropy was that our target labels were integers rather than one-hot encoded vectors.

$$L = \frac{-1}{m}\sum_{i=1}^{m} y_i \cdot log(\hat{y}_i) \tag{5}$$

In (5), each example has a true label vector y_i and a predicted probability distribution vector $\hat{y}_i$. We used 30 epochs and a batch size of 32 when training the model. The dense layer of 64 neurons used a dropout rate of 0.3. Using the outlined hyperparameters and model architecture, we achieved an average accuracy rate of 0.77 on the test set over ten trials.

Table 2: Test set classification rates for individual and combined descriptors using data input method 1 (mfcc + selected power spectrum data).

Descriptor	Average Classification Rate
Brightness	0.823
Focus	0.787
Brightness and Focus	0.719

Table 3: Test set classification rates for individual and combined descriptors using data input method 2 (full power spectrum data).

Descriptor	Average Classification Rate
Brightness	0.834
Focus	0.865
Brightness and Focus	0.772

5 Discussion

Our study aimed to develop a machine-learning-based tool for classifying flute tone quality, focusing on the descriptors of brightness and focus. The results of our model indicate that flute tone quality classification is a feasible problem for machine learning but presents challenges due to the subjective nature of tone perception and the limited size of labeled data available.

The classification rates for brightness and focus individually as well as the overall classification rates for the test set are shown in Table 2 and Table 3 as an average over ten full training cycles. Note that brightness and focus were treated as binary classification tasks each. It is interesting that the errors were highly correlated between classifying focus and brightness for both data input methods. While the second input method was more computationally demanding, providing full spectral data for each sample yielded the highest average test accuracy of 0.77.

Our results suggest that while the model successfully captured some aspects of flute tone quality, there is room for improvement. One contributing factor to the lower accuracy could be the granularity of our labeling system. Our three-level scale for brightness and focus

may not have been fine-grained enough to capture the full range of tone variations perceived by expert flutists. Future iterations of this study could explore a finer rating scale or continuous values for labeling to improve precision.

Another key factor influencing model performance was the size and diversity of the dataset. While segmentation increased the number of training samples, it did not increase the variety of recorded environments, microphones, or playing styles represented in the dataset. This limitation may have led to overfitting on the specific acoustic characteristics of our dataset. A more diverse dataset, including recordings from a wider range of flutists and recording conditions, would likely improve generalizability.

In terms of data processing methods, exploring other, simpler audio features such as spectral flatness could improve the model's ability to capture focus and brightness. Additionally, inputting data from note onsets may provide the model with important information on the articulation style and its effects on tone quality.

Hyperparameter tuning had a significant impact on the final model's accuracy. Our tests indicated that an optimizer learning rate of 0.0001, batch size of 32, and dropout rate of 0.3 provided the best balance between convergence stability and generalization. However, further optimization using automated hyperparameter search methods, such as Bayesian optimization [23], could refine the model further. In conclusion, our machine-learning-based flute tone quality classifier demonstrates promise but requires further refinement.

Future work could focus on expanding the dataset, improving the granularity of labels, and incorporating additional spectral and temporal features. With continued development, this tool could serve as a valuable supplement for musicians seeking objective feedback on their tone quality and for audition judges aiming to reduce bias in their evaluations.

References

[1] Toff, N., *The Flute Book: a Complete Guide for Students and Performers*, Oxford University Press, 2012.

[2] Rawat, P., Bajaj, M., Vats, S., and Sharma, V., "A Comprehensive Study Based on MFCC and Spectrogram for Audio Classification," *Journal of Information and Optimization Sciences*, 44(6), pp. 1057–1074, 2023.

[3] Ghildiyal, A., Singh, K., and Sharma, S., "Music Genre Classification Using Machine Learning," in *2020 4th International Conference on Electronics, Communication and Aerospace technology (ICECA)*, pp. 1368–1372, IEEE, 2020.

[4] Nykänen, A. and Johansson, Ö., "Development of a language for specifying saxophone timbre," in *Stockholm Music Acoustics Conference: 06/08/2003-09/08/2003*, volume 2, pp. 647–650, Kungl. Tekniska Högskolan, 2003.

[5] Hakala, R., "Changing my sound: Developing the Tone Quality on the Flute with Embouchure Change," *Gothenburg University Library*, 2023.

[6] Yorita, R. and Clements, J., "Using spectral analysis to evaluate flute tone quality," in *Proceedings of Meetings on Acoustics*, volume 23, AIP Publishing, 2015.

[7] Hernandez-Olivan, C. and Beltran, J. R., "Timbre classification of musical instruments with a deep learning multi-head attention-based model," *arXiv preprint arXiv:2107.06231*, 2021.

[8] Agarwal, M. and Greer, R., "Spectrogram-Based Deep Learning for Flute Audition Assessment and Intelligent Feedback," in *2023 IEEE International Symposium on Multimedia (ISM)*, pp. 238–242, IEEE, 2023.

[9] Xu, C., "Analysis and Optimization of Flute Playing and Teaching System Based on Convolutional Neural Network," *Mathematical Problems in Engineering*, 2022(1), p. 1846863, 2022.

[10] Giraldo, S., Waddell, G., Nou, I., Ortega, A., Mayor, O., Perez, A., Williamon, A., and Ramirez, R., "Automatic assessment of tone quality in violin music performance," *Frontiers in Psychology*, 10, p. 334, 2019.

[11] Font, F., Roma, G., and Serra, X., "Freesound technical demo," in *Proceedings of the 21st ACM international conference on Multimedia*, pp. 411–412, 2013.

[12] Bandiera, G., Picas, O. R., Tokuda, H., Hariya, W., Oishi, K., and Serra, X., "Good-sounds. org: A Framework to Explore Goodness in Instrumental Sounds." in *ISMIR*, pp. 414–419, 2016.

[13] McFee, B., Raffel, C., Liang, D., Ellis, D. P., McVicar, M., Battenberg, E., and Nieto, O., "librosa: Audio and music signal analysis in python." *SciPy*, 2015, pp. 18–24, 2015.

[14] Harris, C. R., Millman, K. J., Van Der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., et al., "Array programming with NumPy," *Nature*, 585(7825), pp. 357–362, 2020.

[15] Davis, S. and Mermelstein, P., "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE transactions on acoustics, speech, and signal processing*, 28(4), pp. 357–366, 1980.

[16] Testa, A., Gallo, D., and Langella, R., "On the processing of harmonics and interharmonics: Using Hanning window in standard framework," *IEEE Transactions on Power Delivery*, 19(1), pp. 28–34, 2004.

[17] Hartigan, J. A. and Wong, M. A., "Algorithm AS 136: A k-means clustering algorithm," *Journal of the royal statistical society. series c (applied statistics)*, 28(1), pp. 100–108, 1979.

[18] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al., "{TensorFlow}: a system for {Large-Scale} machine learning," in *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pp. 265–283, 2016.

[19] Gulli, A. and Pal, S., *Deep learning with Keras*, Packt Publishing Ltd, 2017.

[20] Agarap, A. F., "Deep learning using rectified linear units (relu)," *arXiv preprint arXiv:1803.08375*, 2018.

[21] Kingma, D. P. and Ba, J., "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[22] Jadon, S., "A survey of loss functions for semantic segmentation," in *2020 IEEE conference on computational intelligence in bioinformatics and computational biology (CIBCB)*, pp. 1–7, IEEE, 2020.

[23] Snoek, J., Larochelle, H., and Adams, R. P., "Practical bayesian optimization of machine learning algorithms," *Advances in neural information processing systems*, 25, 2012.

A Training Details

The following figures outline our hyperparameter testing stage. They track train and test set evaluation accuracy and error over each epoch of training as we manipulate the learning rate, number of epochs, dropout rate, and batch size. Note that we input data from preprocessing Method 2 for training.

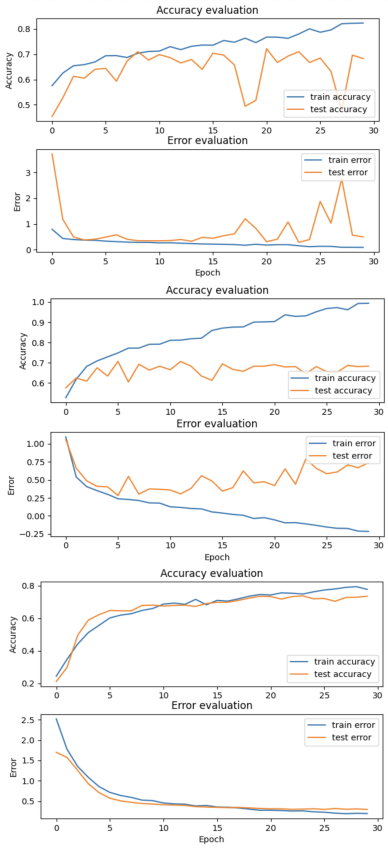


Fig. 6: From top to bottom, the learning rates used were 0.01, 0.001, 0.0001.

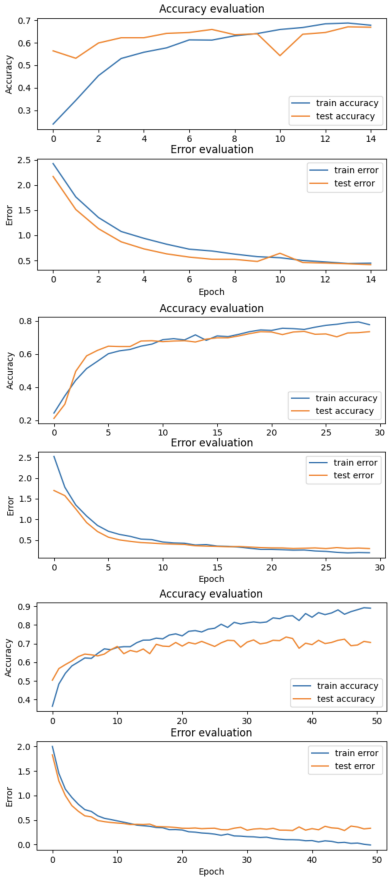
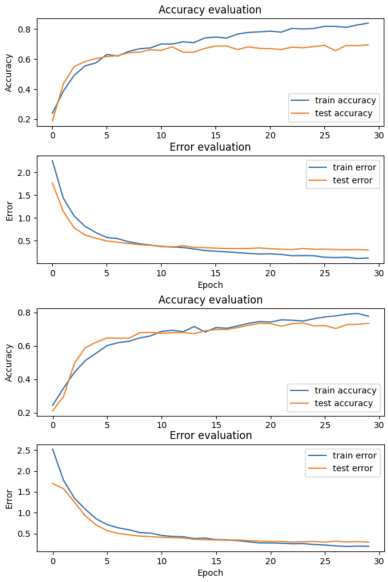


Fig. 7: From top to bottom, the number of epochs used were 15, 30, 50.



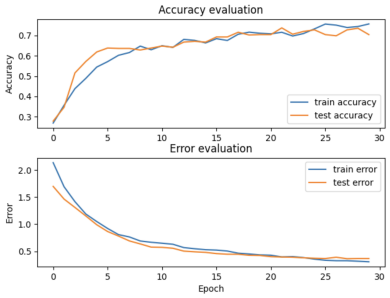


Fig. 8: From top to bottom, the batch sizes used were 16, 32, 64.

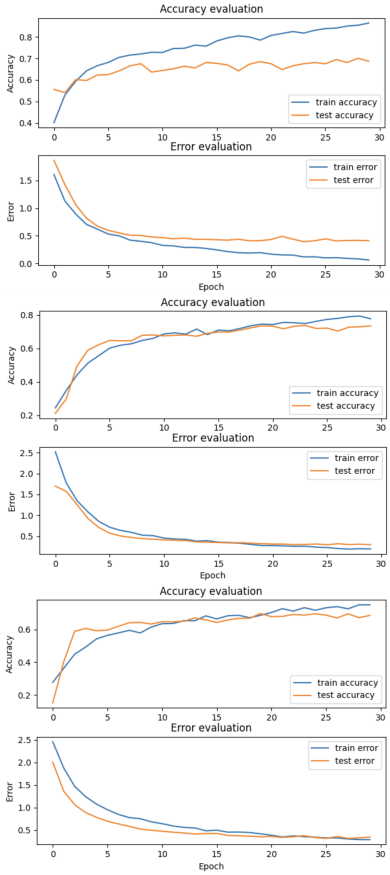


Fig. 9: From top to bottom, the dropout rates used were 0.2, 0.3, 0.5.

Audio Engineering Society

Conference Paper 7

Presented at the AES International Conference on Machine Learning and Artificial Intelligence for Audio
2025 September 8–10, London, UK

This paper was peer-reviewed as a complete manuscript for presentation at this conference. This paper is available in the AES E-Library (http://www.aes.org/e-lib), all rights reserved. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Journal of the Audio Engineering Society.

Automatic Audio Equalization with Semantic Embeddings

Eloi Moliner[1], Vesa Välimäki[1], Konstantinos Drossos[2], and Matti S. Hämäläinen[2]

[1]*Acoustics Lab, Department of Information and Communications Engineering, Aalto University, Espoo, Finland*
[2]*Nokia Technologies, Tampere/Espoo, Finland*

Correspondence should be addressed to Eloi Moliner (eloi.moliner@aalto.fi)

ABSTRACT

This paper presents a data-driven approach to automatic blind equalization of audio by predicting log-mel spectral features and deriving an inverse filter. The method uses a deep neural network, where a pre-trained model provides semantic embeddings as a backbone, and only a lightweight head is trained. This design is intended to enhance training efficiency and generalization. Trained on both music and speech, the model is robust to noise and reverberation. Objective evaluations confirm its effectiveness, and subjective tests show performance comparable to that of an oracle that uses true log-mel spectral features, indicating that the model accurately estimates the desired characteristics, with remaining limitations attributed to the filtering stage. Overall, the results highlight the potential of the method for real-world audio enhancement applications.

1 Introduction

Automatic blind equalization (EQ) of speech and music is a key aspect in audio processing. The goal is to adjust the spectrum of an audio signal to achieve a desired tonal balance without knowledge of the original recording conditions or the target equalization curve [1]. Such processing enhances the audio listening experience by improving clarity and consistency. By enabling data-driven adjustments, EQ can significantly improve sound quality across a wide range of contexts. Automatic equalization is used in music production [2, 3], hearing aids [4], and teleconferencing [5].

This problem has been studied extensively. A well-known approach involves estimating an equalization curve by averaging the spectral characteristics of a reference recording or collection of recordings [6, 7]. More recently, deep learning techniques have been explored, employing neural networks to model complex example-dependent spectral transformations. End-to-end convolutional networks have been applied to automatic EQ [8], while methods based on self-supervised frameworks have been proposed for blind parameter estimation [9, 10, 11]. Closer to this work, Mockenhaupt et al. [12] estimated EQ targets using a classification-based approach, though the reliance on manually defined targets limits flexibility. Generative models have also been explored, with Moliner et al. applying a diffusion-based model for equalization and audio restoration [13].

While deep learning has shown promise for EQ, practical deployment requires models that generalize across diverse signals and remain robust to noise and reverberation. This paper presents a data-driven approach to blind equalization by predicting log-mel spectral features and deriving an inverse filter, explicitly optimized for robustness and generalization.

This paper is organized as follows. Sec. 2 defines the blind EQ problem as an inverse filtering task. Sec. 3 details the proposed method, which combines a pre-trained feature extractor with a lightweight trainable head for parameter estimation. Sec. 4 describes the experimentals, including training on speech and music. Secs. 5 and 6 present objective and subjective evaluations. Finally, Sec. 7 summarizes key findings.

2 Problem Definition

The problem is how to automatically equalize the tonal balance of a degraded audio signal towards a reference. Let $\mathbf{x} \in \mathbb{R}^L$ represent a high-quality reference audio signal. We observe a degraded version of this signal, $\mathbf{y} \in \mathbb{R}^L$, generated through the forward model:

$$\mathbf{y} = \mathcal{A}(\mathbf{x}) + \mathbf{n}, \tag{1}$$

where $\mathcal{A}(\cdot) : \mathbb{R}^L \to \mathbb{R}^L$ is a degradation operator (e.g., filtering or nonlinear distortion), and $\mathbf{n} \in \mathbb{R}^L$ captures additive noise from environmental interference, equipment limitations, or other sources. Our objective is to estimate a reconstruction $\hat{\mathbf{x}} \approx \mathbf{x}$. Assuming that $\mathcal{A}(\cdot)$ is a linear time-invariant system, the degradation can be modeled as a convolution with a finite impulse response filter $\mathbf{h} \in \mathbb{R}^K$ of length K:

$$\mathcal{A}(\mathbf{x}) \approx \mathbf{h} * \mathbf{x}, \tag{2}$$

where $*$ denotes discrete convolution. We further assume that the problem is well-conditioned, implying that the filter is invertible, with inverse $\mathbf{h}^{-1}$. In the noiseless case, the reference can be approximated as

$$\mathbf{x} \approx \hat{\mathbf{x}} = \hat{\mathbf{h}}^{-1} * \mathbf{y}. \tag{3}$$

Thus, the task of estimating the reference is reduced to estimating the inverse filter $\hat{\mathbf{h}}^{-1}$. However, this approximation does not hold in the presence of noise $\mathbf{n}$. As we will explain later, this issue is addressed by pre-processing $\mathbf{y}$ with a denoiser. In cases where the problem is ill-conditioned, such as when portions of the signal (e.g., certain frequency bands or segments) are lost or severely degraded, our method is limited to restoring only the nondegraded parts of the signal.

3 Methods

3.1 Approximating the Inverse Filter

This work aims to estimate the inverse filter $\mathbf{h}_i^{-1}$ without any prior knowledge of the forward filter $\mathbf{h}_i$, making it a blind estimation task. The approach is performed in the frequency domain, where we estimate the inverse filter $\hat{\mathbf{H}}^{-1} \in \mathbb{C}^{N_{\text{FFT}}}$, based on a chosen FFT size N_{FFT}.

Equation 3 can be written in the frequency domain as

$$\mathbf{x} \approx \mathcal{F}^{-1}\left(\hat{\mathbf{H}}^{-1} \odot \mathcal{F}(\mathbf{y})\right), \tag{4}$$

where $\mathcal{F}(\cdot)$ and $\mathcal{F}^{-1}(\cdot)$ denote the short-time Fourier transform (STFT) and its inverse (iSTFT), respectively. Both transformations are computed using N_{FFT} frequency bins, with a window size and hop size defined in Sec. 4.3. Owing to the Hermitian symmetry of the Fourier transform for real-valued signals, only $N_{\text{FFT}}/2+1$ bins contain non-redundant information, and thus only these need to be estimated in practice.

We constrain the inverse filter $\hat{\mathbf{H}}^{-1}$ to be a zero-phase filter, and approximate its magnitude as

$$|\hat{\mathbf{H}}^{-1}| = \sqrt{\frac{\mathbf{X}^2_{\text{ref}}}{\mathbf{Y}^2_{\text{avg}}}}, \tag{5}$$

where $\mathbf{Y}^2_{\text{avg}}$ is the time-averaged Power Spectral Density (PSD) of the observed signal $\mathbf{y}$, computed as

$$\mathbf{Y}^2_{\text{avg}} = \frac{1}{M}\sum_{m=0}^{M-1} |\text{STFT}(\mathbf{y})_m|^2. \tag{6}$$

Here, M is the number of STFT frames, and $\mathbf{X}^2_{\text{ref}}$ denotes the target or reference spectral shape, which is typically content-dependent. The reference $\mathbf{X}^2_{\text{ref}}$ could be a user-defined spectral curve, or, more generally, the average power spectrum of the unknown clean source $\mathbf{x}$, given in an analogous way as in Eq. (6).

Since the clean source $\mathbf{x}$ is unknown, we estimate the target spectrum $\hat{\mathbf{X}}^2_{\text{ref}} \approx \mathbf{X}^2_{\text{ref}}$ using a deep neural network (DNN). The DNN denoted as $E_\theta(\mathbf{y})$ is trained to predict the target spectrum given the observed signal $\mathbf{y}$, where θ represents the trainable parameters of the network.

Instead of predicting the target spectral shape directly $\hat{\mathbf{X}}^2_{\text{ref}}$, we reduce its dimensionality by using a mel-scale triangular filterbank. The mel-transformed reference spectrum $\mathbf{z}_{\text{ref}} \in \mathbb{R}^{N_{\text{mel}}}$ is given by

$$\mathbf{z}_{\text{ref}} = 10\log_{10}(\mathbf{M}\mathbf{X}^2_{\text{ref}}), \tag{7}$$

where $\mathbf{M} \in \mathbb{R}^{N_{\text{mel}} \times (N_{\text{FFT}}/2+1)}$ represents the mel-scale filterbank, which compresses the non-redundant frequency bins into fewer mel-scaled bins N_{mel}. The

DNN is trained to predict the features $\mathbf{z}_{\text{ref}}$, such that $E_\theta(\mathbf{y}) = \hat{\mathbf{z}} \approx \mathbf{z}_{\text{ref}}$.

Once we have obtained $\hat{\mathbf{z}}$, the estimated time-averaged PSD in the original frequency domain can be computed by reversing the mel transformation:

$$\hat{\mathbf{X}}^2_{\text{ref}} = \mathbf{M}^T 10^{\hat{\mathbf{z}}/10}. \tag{8}$$

Here, $\mathbf{M}^T$ is the transpose of the mel filterbank, which interpolates the full FFT frequency resolution from the lower-dimensional mel representation.

3.2 Blind Parameter Estimation

The DNN model E_θ is designed to estimate the log-mel spectral features $\mathbf{z}(\mathbf{x})$ of an original recording $\mathbf{x}$, given a processed observation $\mathbf{y}$. The target parameters $\mathbf{z}(\mathbf{x})$ are extracted from the clean signal $\mathbf{x}$ using a procedure described in Sec. 3.1. During training, the observation $\mathbf{y}$ is generated by applying a forward transformation $f(\mathbf{x})$ to the same original recording. This transformation introduces a range of perturbations, including randomized spectral coloration, variations in loudness, room reverberation, and additive noise. This makes the spectral parameter estimation non-trivial.

We hypothesize that $\mathbf{z}(\mathbf{x})$, being both time-averaged and compressed through a mel filterbank, is correlated with the type of content in the input signal. For example, orchestral music tends to have a distinct spectral envelope, while speech has a different one, and characteristics like gender or age may further affect the spectral shape of speech signals. Based on this, the model $E_\theta(\mathbf{y})$ is expected to extract semantic information from the processed signal $\mathbf{y}$ and map it to the corresponding log-mel spectral features $\hat{\mathbf{z}}$.

To achieve this, we propose factorizing the DNN model E_θ into two components, such that

$$E_\theta(\mathbf{y}) = (\text{MLP}_{\theta_{\text{MLP}}} \circ \text{CLAP}_{\theta_{\text{CLAP}}})(\mathbf{y}), \tag{9}$$

where $\text{CLAP}_{\theta_{\text{CLAP}}}$ is a pretrained audio encoder [14], and $\text{MLP}_{\theta_{\text{MLP}}}$ is a small multi-layer perceptron (MLP) that learns the mapping from Contrastive Audio Language Pretraining (CLAP) embeddings to log-mel spectral features. CLAP is a joint audio-text encoder trained via contrastive learning, mapping audio and text pairs into a shared embedding space. We use the audio encoder of CLAP, with its parameters frozen during training, to extract semantic features from the signal $\mathbf{y}$.

Previous work has shown that embeddings from CLAP and similar models often fail to capture the details of audio effects and transformations, such as those introduced by spectral processing, limiting their sensitivity to these changes [15, 16]. This is beneficial in our case, as it suggests that the forward operator $f(\cdot)$, which introduces spectral coloration, may not significantly affect CLAP-derived embeddings. Thus, $\text{MLP}_{\theta_{\text{MLP}}}$ can focus on learning the relationship between the extracted embeddings and the log-mel spectral features.

The parameters of $\mathbf{z}(\mathbf{x})$ typically exhibit varying magnitudes, which generally decrease logarithmically as a function of frequency. Normalizing both the input and target data is beneficial during neural network training, as it ensures consistent scaling and improves model convergence. To achieve uniform error impact across all frequency bands, we first compute the average $\mathbf{z}_{\text{avg}}$ coefficients from a representative subset of the training data. The model is then trained to estimate the deviation of each instance from this average, ensuring balanced attention across the different frequency components.

The parameter estimation model E_θ is optimized using the following loss function:

$$\mathcal{L} = \mathbb{E}_{\mathbf{x}\sim p_x, f(\cdot)\sim p_f}\left[\|E_\theta(f(\mathbf{x})) - (\mathbf{z}(\mathbf{x}) - \mathbf{z}_{\text{avg}})\|_1\right], \tag{10}$$

where $\mathbf{x}$ represents clean audio data sampled from a distribution p_x (e.g., a dataset of recordings), and the forward operator $f(\cdot)$ is sampled from a distribution p_f, representing different types of data augmentations. As illustrated in Fig. 3, these degradations include room reverberation, randomized spectral coloration, additive noise, and gain randomization. Importantly, since the CLAP encoder is frozen during training, only the parameters of the MLP $\theta_{\text{MLP}} \subset \theta$ are optimized based on the loss objective. It is important to note that, due to the ℓ_1-norm minimization objective, at convergence the network E_θ approximates the conditional median of $\mathbf{z}(\mathbf{x})$ given the transformed input $f(\mathbf{x})$ under the training data distribution. Thus, the estimated power spectrum $\hat{\mathbf{z}} = E_\theta(f(\mathbf{x}))$ can be interpreted as

$$\hat{\mathbf{z}} \approx \text{median}_{\mathbf{z}\sim p(\mathbf{z}|f(\mathbf{x}))}[\mathbf{z} - \mathbf{z}_{\text{avg}}] + \mathbf{z}_{\text{avg}}, \tag{11}$$

where the median is computed implicitly over the target distribution defined by the training dataset.

The training process is illustrated in Fig. 1, and the signal processing conducted at inference time is summarized in Fig. 2. For simplicity, the detail of $\mathbf{z}_{\text{avg}}$ is omitted in both Fig. 1 and 2.

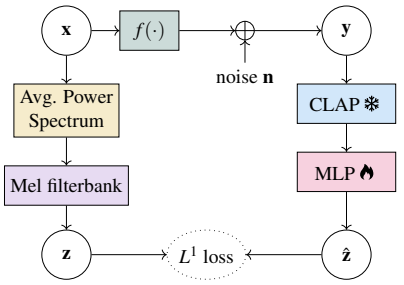


Fig. 1: Training diagram.

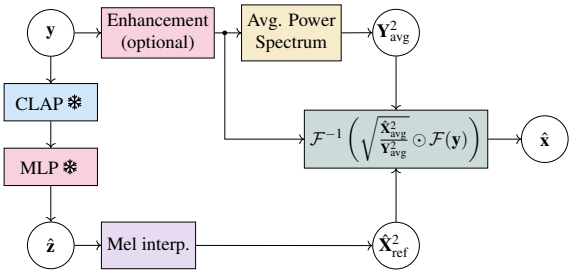


Fig. 2: Inference diagram.

4 Experiments

4.1 Training Data Pipeline

We develop a single model capable of processing both speech and music. The model is designed to exhibit robustness to background noise, gain variations, and reverberation, necessitating the inclusion of these disturbances within the training data pipeline. We utilize the speech datasets VCTK [17] and EARS [18], both of which consist of high-quality studio speech recordings sampled at 48 kHz, and which overall contain around 200 different speakers.

We choose MedleyDB [19] and DSD100 [20] as the music datasets, both offering studio-quality recordings across various genres such as rock, pop, classical, and jazz. These datasets are mixed by recording engineers, allowing us to consider the spectral balance as an appropriate target. Although the datasets provide isolated stems for every source in the microphone, we limit our use to the mixed tracks. Every audio segment undergoes loudness normalization to achieve a standardized volume level of -18 LUFS. This procedure guarantees a consistent loudness target, ensuring uniformity across different audio files.

The degradation process is outlined in Fig. 3, illustrating the practical implementation of $f(\cdot)$ as shown in Fig. 1. Initially, an audio segment with approximately 7 s in duration, containing either speech or music, is convolved with a room impulse response (RIR). We specifically use RIRs from the Arni dataset [21]. Subsequently, the audio is subjected to an equalization filter featuring random gains across various frequency bands. This filter is constructed in the frequency domain as a zero-phase FIR filter. The gain for a set of 26 bark-scaled frequency bands is stochastically sampled on a log-uniform scale ranging from -20 to 20 dB. In the next phase, noise is introduced. For noise sources, the TAU Urban Acoustic Scenes dataset [22] is employed during training. This dataset is integrated with speech and music by mixing with noise from this dataset, achieving an SNR that is log-uniformly sampled between 5 and 50 dB.

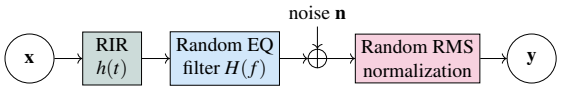


Fig. 3: Pipeline illustrating the applied degradations.

Finally, the gain of the mixture is adjusted through a random root-mean-square (RMS) normalization. The RMS normalization is implemented as $y = (\alpha\sqrt{L}/\|\mathbf{x}\|_2)\mathbf{x}$, with the parameter α being uniformly sampled within -5 dB and 30 dB.

4.2 Training Details

The training of the model consisted of conducting 100,000 iterations, utilizing the Adam optimization algorithm. The learning rate employed during this process was set to 1×10^{-4}, and the values of the beta momentum parameters were configured to be (0.9, 0.999). For the training data, audio segments of 7 s sampled at 48 kHz were used. The training was performed with a batch size of 16.

We chose a number of mel-frequency coefficients $N_{\text{mel}} = 32$, extracted using the STFT with FFT size 2048, a Hann window with size 1024, and a hop length of 256 samples. The employed MLP contains 0.34 M parameters and is based on three linear layers with a hidden size of 256 and Leaky-ReLU nonlinearities.

4.3 Evaluation Data

The model is evaluated on speech and music, and in different scenarios, according to the applied degradations.

We distinguish the following scenarios: 1. Speech equalization, 2. Music equalization, 3. Noisy speech equalization, and 4. Reverberant speech equalization.

For scenarios 1, 3, and 4, we evaluate on VCTK speakers "p351", "p360"–"p364", "p374", "p376" and EARS speakers "p100"–"p107". In scenario 2, we extract 8-s segments from separated test splits of MedleyDB and DSD100. For each of these scenarios, a test set comprising 1,000 samples is created. To each example, a random filter is applied following the same procedure used during training, and the gain is randomized with an RMS in the interval [0.005, 0.30], extending the range seen at training time.

In scenario 3, we introduce noise from the DEMAND dataset [23] and apply SNRs ranging between -5 dB and 30 dB, which also extends beyond the training range. Finally, in scenario 4, the signals are convolved with RIRs, utilizing simulated RIRs from pyroomacoustics [24], with a reverberation time T_{60} spanning [0.1, 1.0]. In both scenarios, we apply DeepFilterNet2 [25] to pre-process audio signals that are either noisy or reverberant. DeepFilterNet2 was selected due to its open-source availability, lightweight architecture, high processing speed, and acceptable performance. It is important to note that the original signal remains accessible to the model, while the pre-processing is integrated into the equalization pipeline, as shown in Fig. 2.

4.4 Metrics

To evaluate the performance of the model, we consider several metrics that provide a comprehensive analysis of its effectiveness in estimating the power spectrum and enhancing audio signals. These metrics capture both spectral accuracy and audio-domain fidelity.

We compute the L1 error between the target average power spectrum extracted from the reference high-quality recording and the estimated power spectrum. This metric corresponds directly to the loss function used during training, as defined in Eq. (10). It provides a measure of how closely the model's predictions align with the target power spectrum, reflecting its ability to learn accurate spectral representations.

To evaluate the fidelity of the equalized audio signal, we employ the Log-Spectral Distortion (LSD), which measures differences in spectral features between the reference and predicted audio signals. The LSD is computed as

$$\mathrm{LSD} = \frac{1}{MK} \sum_{m,k} \left| \log(|\mathcal{F}(\mathbf{x})| + \varepsilon) - \log(|\mathcal{F}(\hat{\mathbf{x}})| + \varepsilon) \right|, \tag{12}$$

where $\mathcal{F}$ represents the Short-Time Fourier Transform (STFT) operator, M is the number of time frames, K is the number of frequency bins, and ε is a small constant (e.g., 10^{-6}) to avoid logarithmic instabilities.

While these objective metrics are useful for evaluating model performance, they are not always directly indicative of perceptual quality. For example, the equalization applied to music examples may deviate from the median power spectrum the model predicts. This is because the model's estimation inherently reflects the statistical properties of the training data, whereas real-world audio signals often exhibit unique spectral characteristics. Thus, while the metrics provide quantitative insights, qualitative evaluation through listening tests is essential to fully assess the method's impact on perceptual audio quality.

4.5 Baselines

To contextualize the performance of our proposed method, we designed and evaluate the proposed method against several baselines. The first baseline, which we call *Average*, computes the average power spectrum of the training data, and uses that to compute the inverse filter in Eq. (5). We compute the average of the speech data and music data separately. By using these fixed average spectra as predictions, this baseline is meant to highlight the effectiveness of the proposed method at leveraging data-specific patterns compared to a naive global estimate.

The second baseline, which we call *End-to-end*, replaces the frozen CLAP encoder with a fully trainable encoder, allowing the model to jointly optimize feature extraction and power spectrum estimation. We used the same neural network architecture as in the setup from Steinmetz et al. [9]. This baseline aims to demonstrate the advantages of using a pre-trained representation over task-specific feature learning.

Finally, the *Oracle* baseline uses parameters directly estimated from the reference high-quality audio signal, bypassing the need for a model prediction. This serves as an upper bound on performance within our framework, illustrating the best possible outcome achievable when the target parameters are perfectly known.

5 Objective Evaluation

5.1 Speech Equalization and Gain Adjustment

In this scenario, speech signals undergo arbitrary EQ and random RMS processing. The outcomes are depicted in Fig. 4. Each plot displays the median curves represented by colored lines, with the InterQuartile Range (25%-75%) illustrated as shaded regions. Figure 4a illustrates L^1 errors on parameter estimation in relation to frequency, highlighting increased errors in higher frequency ranges. Both the Proposed and End-to-end models display comparable performance, surpassing the Average model. Figure 4b presents L^1 errors concerning the RMS of the altered signal, showing consistent performance across all RMS levels, despite training only within the [0.02, 0.18] interval. Performance seems similar even beyond this range. Figure 4c depicts LSD relative to frequency, indicating that Proposed, Average, and End-to-end perform similarly, though less effectively than the Oracle condition. In Figure 4d, LSD errors in relation to RMS reveal higher errors at lower RMS values, even in the Oracle scenario, underscoring a fundamental limitation of the employed inverse filter approach (see Eq. (5)).

5.2 Music Equalization and Gain Adjustment

In Scenario 2, musical signals were processed utilizing randomized equalization filters and gain levels. The outcomes, illustrated in Fig. 5 and mirroring the data from Fig. 4, exhibit patterns akin to those identified in Scenario 1. Nevertheless, here, both the Average and End-to-end baselines demonstrate significant performance reduction across all metrics. This deterioration in the Average baseline's effectiveness is probably attributable to the wider variability found in music as opposed to speech. The Proposed model sustains superior overall performance compared to the End-to-end baseline, suggesting that the use of pretrained CLAP embeddings facilitates robust feature extraction, even for the more complex music signals. This scenario is also evaluated through subjective listening in Sec. 6.

5.3 Noisy Speech Equalization

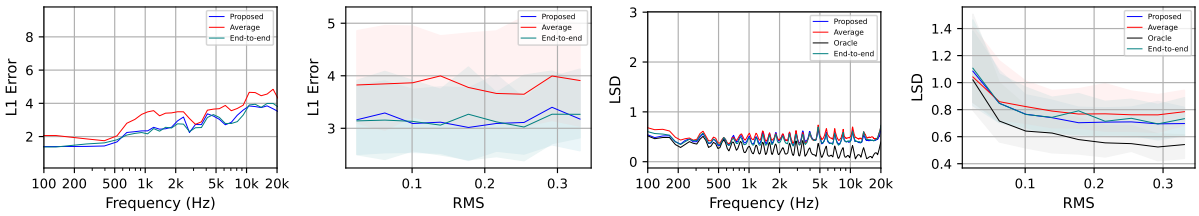
This scenario investigates the impact of background noise on inverse filter estimation by analyzing the relationship between the objective metrics and the input SNR. Figure 6a presents the L^1 error of the estimated parameters as a function of SNR. Both the Proposed and End-to-end methods exhibit similar performance, with a noticeable but relatively minor increase in error at lower SNRs, which aligns with expectations given the added noise.

The LSD metric, defined between audio waveforms, is affected by the speech enhancement model used for preprocessing. In Fig. 6b, we evaluate performance using DeepFilterNet2 [25] for speech enhancement prior to equalization. Here, all methods exhibit similar trends, heavily influenced by the SNR. To explore an idealized scenario, we apply the estimated equalization (derived from noisy measurements) to a noiseless version of the degraded signal, simulating perfect denoising while retaining spectral coloration. As shown in Fig. 6c, this setup significantly improves LSD performance, achieving near-uniform results across the SNR range. In this case, the *Proposed* and *Average* conditions perform comparably, while the *End-to-end* baseline shows a noticeable decline in performance.

5.4 Reverberant Speech Equalization

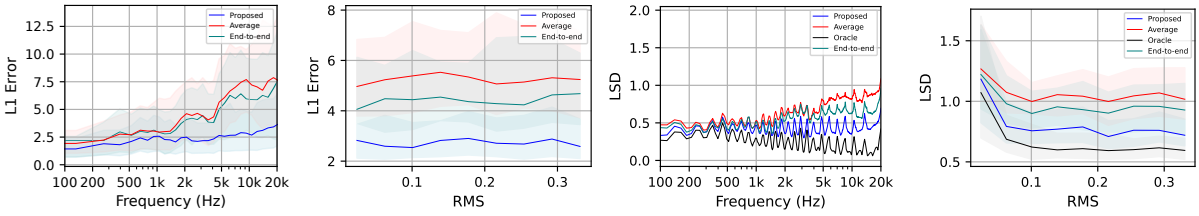
This scenario builds upon the analysis in Sec. 5.3, shifting the focus to robustness against reverberation. We examine how different metrics vary with the reverberation time applied to the measurements. Figure 7a shows the L^1 parameter estimation error as a function of the reverberation time, T_{60}. The Proposed and End-to-end methods perform similarly, with a slight increase in error at longer T_{60} values.

Figure 7b depicts the audio-domain LSD metric with respect to T_{60}, where the equalization is applied to signals enhanced by DeepFilterNet2. As in Sec. 5.3, the metrics are heavily influenced by the performance of the speech enhancement model, overshadowing the underlying robustness of the equalization methods. To address this limitation, we repeat the experiment under an idealized scenario where perfect dereverberation is assumed. Specifically, the reference dry signal is equalized using the inverse filter estimated from the reverberant measurement. The result, presented in Fig. 7c, shows that the correlation between the metrics and T_{60} becomes negligible under these conditions. In this idealized setup, the Proposed method outperforms End-to-end and demonstrates a marginal but consistent advantage over the Average method in terms of LSD, further highlighting its robustness to reverberation.



(a) L_1 error vs frequency **(b)** L_1 error vs RMS **(c)** LSD error vs frequency **(d)** LSD error vs RMS

Fig. 4: Objective metrics from Scenario 1: Equalization and gain adjustment for speech.



(a) L_1 error vs frequency **(b)** L_1 error vs RMS **(c)** LSD vs frequency **(d)** LSD vs RMS

Fig. 5: Objective metrics from Scenario 2: Equalization and gain adjustment for music.

6 Subjective Evaluation

The goal of the listening test was to evaluate the performance of the proposed method for music equalization, focusing on subjective preference. As music equalization is inherently subjective, we aimed to determine which type of equalization participants preferred. The primary question for the test was simple: "Which one do you prefer?". Participants were encouraged to choose randomly when they were unsure.

The test followed a pairwise comparison preference method, where participants were presented with two audio samples and asked to choose the one they preferred. This allowed for direct comparisons between different equalization conditions. We selected four music excerpts, each approximately 7 s long, from four different genres: *Jazz*, *Opera*, *Pop*, and *Rock*. These samples are available in the attached material.

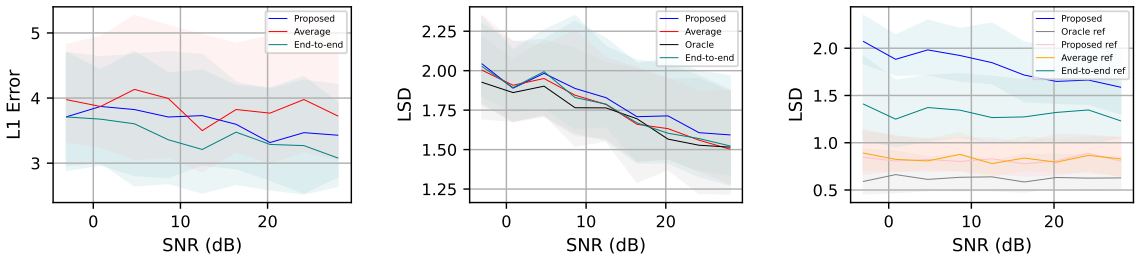
Participants assessed five distinct scenarios: the *Reference* condition, featuring the untouched audio; the *Distorted* condition, where the original audio sample was pre-processed with a random equalization filter before serving as input for subsequent methods; the *Proposed* condition, which was treated using the technique suggested in this study; the *Oracle* condition, which utilized the same equalization approach as the *Proposed* but with access to the reference for calculating the average power spectra; and the *Average* condition, which employed an average equalization approach based on the spectral average of the music training dataset. All samples were loudness-normalized preceding the tests to ensure that evaluations focused on spectral coloration rather than differences in loudness. Every pairwise comparison was conducted twice to eliminate potential biases that might occur if users tend to favor the letters A or B when uncertain. Additionally, all pages were displayed in a random sequence.

We implemented the test using the WebMUSHRA tool [26], with an unofficial version adapted specifically for preference tests [1]. The test involved 8 participants, who all completed it in less than 15 min. The test was intentionally kept brief to prevent participant fatigue. All participants had prior experience with listening tests. The average age was 28, with an equal gender distribution. No participants were excluded.

6.1 Listening Test Results

The counts are aggregated across all participants and trials. The combined results for all genres are provided in Table 1. Figure 8(e) also summarizes the aggregated results of the pairwise comparison of the proposed method with the other conditions. The tables show, for each possible pair, the number of wins, i.e., the times a participant chose one over the other. These tables illustrate, for each possible pair of conditions, the number of times participants preferred one option

[1] https://github.com/Simon-Stone/webMUSHRA/tree/preference_test

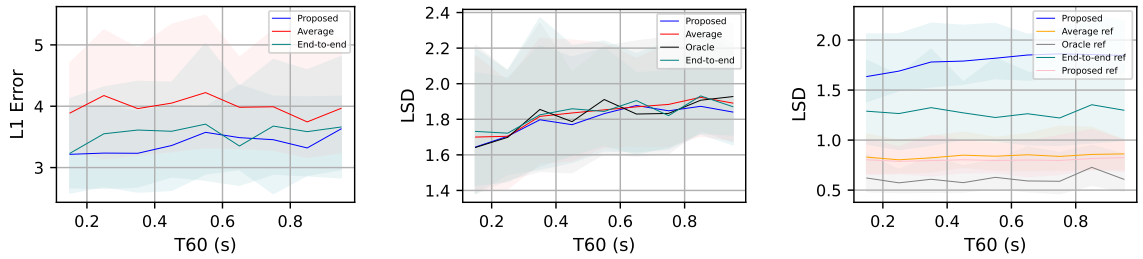


(a) L_1 error vs SNR

(b) LSD error vs SNR. Using speech enhancement.

(c) LSD error vs SNR. Using denoised reference.

Fig. 6: Objective metrics from Scenario 3: Equalization and gain adjustment for noisy speech



(a) L_1 error vs reverberation time T_{60}

(b) LSD vs reverberation time T_{60}. Using speech enhancement.

(c) LSD vs reverberation time T_{60}. Using dereverberated reference.

Fig. 7: Objective metrics from Scenario 4: Equalization and gain adjustment for reverberant speech

over the other. The results from the *Jazz*, *Opera*, *Pop* and *Rock* examples are represented in Table 2. Figures 8(a–d) also show the results of the comparison of the proposed method with the other four conditions.

For each pair, a statistical analysis was performed to determine whether the observed preference was significant. Specifically, a Z-test for proportions was conducted, comparing the proportion of times each condition was selected. The Z-statistic quantifies the difference between these proportions, taking into account the sample size and variability. A positive Z-statistic indicates that the first condition (e.g., “Option A") was preferred more often than the second (e.g., “Option B"), while a negative Z-statistic suggests the opposite. The p-value associated with the Z-statistic represents the probability that the observed difference in preferences could occur due to random chance. A p-value below 0.01 is typically considered statistically significant, suggesting that participants had a genuine preference rather than random variability. The tables use color coding to highlight statistically significant results. The condition preferred more frequently in significant comparisons is highlighted in green as the winner, while the less-preferred condition is highlighted in red.

Table 1: Aggregated preference listening test results across all genres. In the statistically significant cases ($P < 0.01$), the winning and losing methods are indicated with green and red font, respectively.

Option A	Option B	Wins A	Wins B	Z-Statistic	P-Value
proposed	average	51	13	6.72	1.85e-11
proposed	oracle	31	33	-0.35	0.72
proposed	distorted	60	4	9.90	4.18e-23
proposed	reference	15	49	-6.01	1.85e-09
oracle	average	40	24	2.83	0.0047
oracle	distorted	58	6	9.19	3.84e-20
oracle	reference	15	49	-6.01	1.85e-09
average	distorted	55	9	8.13	4.23e-16
average	reference	8	56	-8.49	2.15e-17
distorted	reference	6	58	-9.19	3.84e-20

6.2 Discussion

The results provide meaningful insights into the performance of the tested methods across all genres. As expected, Table 1 shows that the distorted condition was always rated the lowest, underscoring the detrimental impact of the applied random EQ filter on perceived audio quality. The proposed method outperformed or performed comparably to the average condition, with

Table 2: Results of the preference test across various genres. For brevity, paired comparisons between conditions other than the proposed method are omitted.

Genre	Option A	Option B	Wins A	Wins B	Z-Statistic	P-Value
Jazz	proposed	average	8	8	0.00	1.00
	proposed	oracle	8	8	0.00	1.00
	proposed	distorted	16	0	5.66	1.54e-08
	proposed	reference	5	11	-2.12	0.03
Opera	proposed	average	13	3	3.54	0.0004
	proposed	oracle	12	4	2.83	0.0047
	proposed	distorted	16	0	5.66	1.54e-08
	proposed	reference	7	9	-0.71	0.48
Pop	proposed	average	14	2	4.24	2.21e-05
	proposed	oracle	7	9	-0.71	0.48
	proposed	distorted	15	1	4.95	7.43e-07
	proposed	reference	2	14	-4.24	2.21e-05
Rock	proposed	average	16	0	5.66	1.54e-08
	proposed	oracle	4	12	-2.83	0.0047
	proposed	distorted	13	3	3.54	0.0004
	proposed	reference	1	15	-4.95	7.43e-07

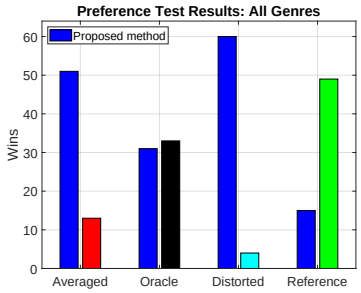


Fig. 8: Listening test results from pairwise comparison of the proposed method. The data correspond to the Wins A and B in the four first lines of Table 1.

Jazz being an exception where the proposed method and the average condition yielded similar results, as can be seen in Table 2. This highlights the capability of the proposed method to retrieve data patterns.

When compared to the *Oracle* condition, the proposed method demonstrated comparable results in the aggregated analysis of Table 1, with no significant differences observed. This is particularly noteworthy because it suggests that the blind power spectrum estimator employed in the proposed method nearly reaches the performance bound under the given conditions. Interestingly, in the *Opera* genre, the proposed method outperformed the Oracle condition, see Table 2. This outcome may stem from the fact that the median estimate of the average power spectrum (see Eq. (11)), used in the proposed method, is subjectively more pleasing than the one derived directly from the reference.

However, both the proposed method and the Oracle condition were consistently rated lower than the Reference condition, as seen in Table 1. This points to a limitation of the inverse filter approximation. The *Reference* condition serves as an idealized target, and the observed quality gap between the Oracle method and the Reference condition suggests that further refinement is needed. One promising avenue for improvement could involve increasing the resolution of the estimated coefficients by expanding the number of frequency bands in the set of parameters that the model estimates.

7 Conclusions

This paper introduces a method for automatic audio equalization using blind power spectrum estimation combined with inverse filtering, showcasing reliable performance across various challenging conditions. The approach has been designed to work with both speech and music and to be robust to noise and reverberation. The proposed method shows a distinct advantage when processing audio with varied semantic content, such as music, while its benefit over baseline methods is less pronounced for speech. Our listening test results indicate a marked preference for this method in different music genres, achieving results close to Oracle level without the need for reference data. However, the gap between the proposed method and the Reference baseline highlights opportunities for further improvement. Current limitations are primarily linked to the inverse filtering stage, particularly the use of a subsampled spectrum via a mel-scale filterbank.

8 Acknowledgment

This work has been financed in part by Nokia Technologies through the DeepMix project (Aalto University project no. 411116).

References

[1] V. Välimäki and J. D. Reiss. All about audio equalization: Solutions and frontiers. *Applied Sciences*, 6(5):129, 2016.

[2] B. De Man, J. Reiss, and R. Stables. Ten years of automatic mixing. In *Proc. 3rd Workshop on Intelligent Music Production*, Salford, UK, 2017.

[3] M. A. Martínez-Ramírez et al. Automatic music mixing with deep learning and out-of-domain data. In *Proc. ISMIR*, 2022.

[4] T. Sankowsky-Rothe, M. Blau, S. Köhler, and A. Stirnemann. Individual equalization of hearing aids with integrated ear canal microphones. *Acta Acustica united with Acustica*, 101(3):552–566, 2015.

[5] X. Liu, J. D. Reiss, and A. Mourgela. An automatic mixing system for teleconferencing. In *Proc. Audio Engineering Society Conv. 153*, 2022.

[6] T.G. Stockham, T.M. Cannon, and R.B. Ingebretsen. Blind deconvolution through digital signal processing. *Proc. IEEE*, 63(4):678–692, 1975.

[7] Z. Ma, J. D. Reiss, and D. Black. Implementation of an intelligent equalization tool using Yule-Walker for music mixing and mastering. In *Proc. Audio Engineering Society Conv. 134*, page 5, 2013.

[8] M. A. Martínez-Ramírez and J. D. Reiss. End-to-end equalization with convolutional neural networks. In *Proc. International Conference on Digital Audio Effects (DAFx)*, 2018.

[9] C. J. Steinmetz, N. J. Bryan, and J. D. Reiss. Style transfer of audio effects with differentiable signal processing. *Journal of the Audio Engineering Society*, 70(9):708–721, 2022.

[10] C. Peladeau and G. Peeters. Blind estimation of audio effects using an auto-encoder approach and differentiable digital signal processing. In *Proc. IEEE ICASSP*, pages 856–860, 2024.

[11] C. J. Steinmetz et al. ST-ITO: Controlling audio effects for style transfer with inference-time optimization. In *Proc. ISMIR*, 2024.

[12] F. Mockenhaupt, J. S. Rieber, and S. Nercessian. Automatic equalization for individual instrument tracks using convolutional neural networks. In *Proc. International Conference on Digital Audio Effects (DAFx)*, 2024.

[13] E. Moliner et al. A diffusion-based generative equalizer for music restoration. In *Proc. International Conference on Digital Audio Effects (DAFx)*, 2024.

[14] B. Elizalde, S. Deshmukh, and H. Wang. Natural language supervision for general-purpose audio representations. In *Proc. IEEE ICASSP*, pages 336–340, 2024.

[15] A. Gui et al. Adapting Frechet audio distance for generative music evaluation. In *Proc. IEEE ICASSP*, pages 1331–1335, 2024.

[16] S. H. Hawley and C. J. Steinmetz. Leveraging neural representations for audio manipulation. In *Proc. Audio Engineering Society Conv.*, 2023.

[17] C. Valentini-Botinhao et al. Reverberant speech database for training speech dereverberation algorithms and TTS models. *Univ. Edinburgh*, 2016.

[18] J. Richter et al. EARS: An anechoic fullband speech dataset benchmarked for speech enhancement and dereverberation. In *Proc. INTERSPEECH*, 2024.

[19] R. M. Bittner et al. Medleydb 2.0: New data and a system for sustainable data collection. *ISMIR Late Breaking and Demo Papers*, 36, 2016.

[20] A. Liutkus et al. The 2016 signal separation evaluation campaign. In *Proc. Latent Variable Analysis and Signal Separation: 13th International Conference, LVA/ICA*, pages 323–332, 2017.

[21] K. Prawda, S. J. Schlecht, and V. Välimäki. Calibrating the Sabine and Eyring formulas. *J. Acoust. Soc. Amer.*, 152(2):1158–1169, 2022.

[22] T. Heittola, A. Mesaros, and T. Virtanen. TAU Urban Acoustic Scenes 2022 Mobile, Development dataset, March 2022.

[23] J. Thiemann, N. Ito, and E. Vincent. The diverse environments multi-channel acoustic noise database (demand): A database of multichannel environmental noise recordings. In *Proc. Meetings on Acoustics*, volume 19, 2013.

[24] R. Scheibler, E. Bezzam, and I. Dokmanić. Pyroomacoustics: A Python package for audio room simulation and array processing algorithms. In *Proc. IEEE ICASSP*, pages 351–355, 2018.

[25] H. Schröter et al. DeepFilterNet2: Towards real-time speech enhancement on embedded devices for full-band audio. In *Proc. International Workshop on Acoustic Signal Enhancement (IWAENC)*, pages 1–5, 2022.

[26] M. Schoeffler et al. WebMUSHRA—A comprehensive framework for web-based listening tests. *Journal of Open Research Software*, 6(1), 2018.

Audio Engineering Society

Conference Paper 8

Presented at the AES International Conference on Artificial Intelligence and Machine Learning in Audio
2025 September 8–10, London, UK

This paper was peer-reviewed as a complete manuscript for presentation at this conference. This paper is available in the AES E-Library (http://www.aes.org/e-lib), all rights reserved. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Journal of the Audio Engineering Society.

Transfer Learning for Neural Modelling of Nonlinear Distortion Effects

Tara Vanhatalo[1,2,3], Pierrick Legrand[1], Myriam Desainte-Catherine[2], Pierre Hanna[2], Guillaume Pille[3], Antoine Brusco[3], and Joshua D. Reiss[4]

[1]*Centre Inria de l'Université de Bordeaux, Institute of Mathematics of Bordeaux, UMR 5251 CNRS, University of Bordeaux, France*
[2]*University of Bordeaux, CNRS, Bordeaux INP, LaBRI, UMR 5800, France*
[3]*Orosys SAS, Saint-Gély-du-Fesc, France*
[4]*Centre for Digital Music, Queen Mary University of London, UK*

Correspondence should be addressed to Tara Vanhatalo (tara.vanhatalo@inria.fr)

ABSTRACT

Modelling analogue distortion replicates the warmth, character, and nonlinear harmonics of classic gear often used in audio production, to be incorporated into modern, digital workflows. Creating such models can be challenging as it requires accurately capturing complex nonlinearities and dynamic responses while while ensuring real-time processing. We evaluate the performance of various transfer learning methods for improved modelling of distortion effects. These approaches take into account the choice of pretrained model, the choice of dataset for fine-tuning, and the method in which the knowledge is transferred from source to target task. We compare the impact of using a source effect closest to the target one for pretraining, determined either via feature extraction or using domain knowledge, to the use of a "general model". We evaluate how to transfer the knowledge from source to target task by assessing which weights to freeze and what data to use for the downstream training. Each approach is assessed on two architectures applied to three example distortion modelling tasks. All models obtained via transfer learning are compared to a baseline model trained from scratch with random initialisation to gauge the gain in accuracy due to the transfer. Results show that transfer learning can help improve model performance and reduce training times when the appropriate transfer approach is used.

1 Introduction

Many iconic audio devices, such as tube amplifiers, tape machines, and vintage pedals, have unique distortion characteristics that are highly sought after. Modelling these effects allows musicians and producers to access these sounds without needing expensive or hard-to-find hardware, and allows for them to be used within digital workflows seamlessly.

One approach to modelling audio distortion effects is to use neural networks in a black-box training pipeline. The training data used to create these models consists of audio signals - such as instrument recordings or synthetic test signals - sent through the target analogue devices, the output of which is recorded and used as ground truth. Reducing the amount of training data

and time has a number of benefits. These include: a reduced risk of error in the data generation portion that can allow for non-specialists to create their own models; less wear-and-tear of the equipment used to create the training data; and reduced environmental impact.

Transfer Learning (TL) is a commonly used method in machine learning, most often applied to cases where there is insufficient training data. It transfers the knowledge from one or more source tasks to a target task and can manifest in a variety of methods. We give a formal definition of transfer learning, adapted from Pan and Yang [1]. In our case there is only one source task but this can easily be extended to the general case.

Given a source domain $\mathbf{D}_S$ and task $\mathbf{T}_S$, a target domain $\mathbf{D}_T$ and task $\mathbf{T}_T$, transfer learning aims to improve the learning of the target predictive function $f(.)$ in $\mathbf{D}_T$ using the knowledge in $\mathbf{D}_S$ and $\mathbf{T}_S$ where $\mathbf{D}_S \neq \mathbf{D}_T$ or $\mathbf{T}_S \neq \mathbf{T}_T$. Here a domain is defined as $\mathbf{D} = \{\mathbf{X}, P(X)\}$ where $\mathbf{X}$ is a feature space and $P(X)$ is a marginal probability distribution, and a task is defined as $\mathbf{T} = \{\mathbf{Y}, f(.)\}$ where $\mathbf{Y}$ is a label space and $f(.)$ is a predictive function learned from training data. In the case of Deep Transfer Learning (DTL), this predictive function takes the form of a neural network [2], [3].

DTL started to arise when Deep Neural Networks (DNN) were used as feature extractors in TL models, a process referred to as "incomplete" transfer in Yu et al. [3]. Around the same time as the incorporation of DNNs into traditional TL, model-based DTL methods also started to emerge [3]. Model-based approaches consist of sharing and fine-tuning parameters of models. In this work, we focus on transfer learning via fine-tuning of the model weights for audio effects modelling.

Little work has explored the possibility of applying TL to neural networks for audio effects modelling. The neighbouring field of Music Information Retrieval (MIR) contains a number example works using TL for various tasks. In Hamel et al. [4], TL applied to genre classification, automatic tag annotating and music similarity is investigated. Oord et al. [5] exploit an available large-scale music dataset, the Million Song Dataset (MSD), for classification tasks on other datasets by reusing models trained on MSD for feature extraction. In Liang et al. [6], a multi-layer neural network trained for semantic tagging prediction is used as a content model for content-aware music recommendation. In Choi et al. [7], a tagging CNN trained on MSD is used to create explicit labels for the downstream task of playlist generation using RNN. In Choi et al. [8], TL takes the form of a pretrained feature extraction network for both music classification and regression tasks. Alfaro-Contreras et al. [9] examine the possible improvements owed to TL using either image or audio data for Optical Music Recognition and Automatic Music Transcription.

In this work, we apply model-based DTL to improve the performance of state-of-the-art static models for audio effects modelling. Although transfer learning is often used in contexts where the downstream (i.e. target) task lacks sufficient labelled data, it can also lead to faster learning, more accurate predictions and/or require less training data. This is of interest for virtual analogue modelling as we could, in theory, train one model on a large amount of data acquire from a chosen audio effect, and then use this trained network as a starting point for all other models of a variety of different, even unrelated, black-box audio effects we wish to create.

In this case the source task is training a model of a chosen audio effect S and the target task is training a model of a different audio effect T. The initial, large training dataset could even be synthetic, using circuit simulation techniques to create the data to avoid the wear-and-tear of electrical components in analogue devices and to minimise the risk of human error during the capture process. Thus, utilising transfer learning could allow for reduced training times in the target tasks which would incur shorter times to create new models and reduce the required computational resources, and thus environmental impact.

These benefits are amplified in the case of parametric models. When we wish to model the behaviour of the user controls' influence on the output audio and have the simulation adapt to different control settings, the amount of training data required increases exponentially when sampling combinatorically which is often the method employed [10]. Reduced amounts of necessary training data are therefore doubly important in such cases.

We therefore evaluate the impact on overall performance of applying transfer learning to this modelling task. We also determine the most effective methods of conducting the pretraining and fine-tuning phases of the knowledge transfer. A case study is carried out on

three example distortion effects with increasing degrees of nonlinearity: an overdrive, a distortion and a fuzz which we aim to model with two example networks, a simple LSTM recurrent network and the feedforward WaveNet, from Wright et al. [11].

The organisation of this paper is as follows. In Section 2, we present the methodology behind both the pretraining and fine-tuning experiments. In Section 3, we present the experimental results and comparison and discuss the findings. All the data used in this work is made available at: `https://www.math.u-bordeaux.fr/~plegra100p/NN/TL`.

2 Methodology

For the pretraining, illustrated at the top right of Figure 1, we compare which type of starting model produces the highest accuracy between a model most similar to the target device and a general-purpose model that can be used regardless of what the target device is. In order to determine the similarity between devices, we use a Convolutional Neural Network (CNN) pretrained for distortion effects classification, dubbed FxNet, to extract features from a number of devices [12]. The closest model is chosen as the one whose features minimise the L_2-distance with the features of a given target effect. Similarly, the general-purpose model chosen is the one whose features have the smallest distance from an average feature vector computed from all the effects in the dataset. More detail on this is given in Subsection 2.1. In order to ascertain how effective this approach is, we compare results with those obtained using closest and general-purpose models obtained with domain knowledge.

For the fine-tuning phase, illustrated at the bottom right of Figure 1, we evaluate the impact of the type of signal used for the training, the length of this signal (in order to determine a minimum amount of data required), and which weights to freeze for optimal results with both networks. All results obtained after fine-tuning are compared to a baseline model trained from scratch using random initialisation.

2.1 Pretraining

Pretraining-based TL describes the case where a model initially trained for a particular source task with a given data distribution, is adapted to a related task with a different training data distribution. This method generally leads to improved model performance and has the benefit of requiring less data for the target model fine-tuning.

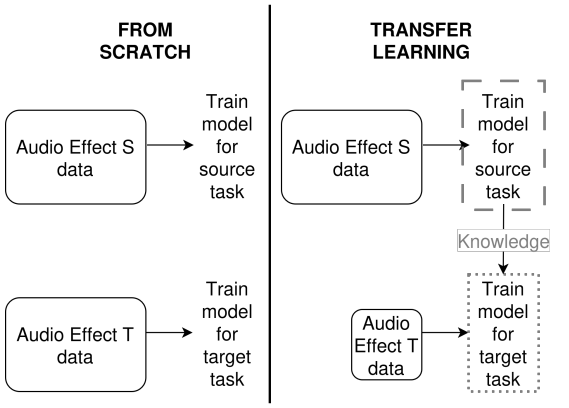


Fig. 1: *Training from scratch vs transfer learning. The pretraining is circled by the dashed line and the finetuning by the dotted line.*

Here, we evaluate the impact on performance that the choice of pretrained source model can have. For this, comparisons are carried out between a source effect that is closely related to the target effect and a source effect that is more generally applicable to various target effects. We call these two choices of pretrained model *closest model* and *general model*.

2.1.1 Choice of Pretrained Model

The transfer of knowledge from source to target task is expected to improve modelling performance. Regarding the choice of pretrained source model, we compare fine-tuning a model deemed closest to the target one with a single model chosen as the starting point for all target model training. This is done in order to gauge whether it is preferable to have very closely related source and target tasks or whether a generally applicable source model is sufficient for the fine-tuning of all effects.

For both experiments, the choice of what constitutes the closest model or the most general model is performed with two methods: one is a choice informed by domain knowledge and another is chosen via feature extraction.

We first present the approach based on domain knowledge and then present the feature extraction in the following subsection. For the closest model, we chose an

emulation of the same pedal as the target one but from a different designer: Knowledge is transferred from the MultiDrive Pedal Pro (MPP) TS808 to the Nembrini Audio (NA) TS808, from the MPP RAT to the NA RAT, and from the Distorque Face Bender to the MPP Fuzz Face. Table 1 describes all plugins used and the devices they are based on.

Table 1: *Plugins used for all data generation.*

Designer	Plugin	Emulation of
Audified	MultiDrive Pedal Pro	Ibanez TS808 Ibanez TS9 Boss BD2 Boss OD1 Boss SD1 Boss DS1 Boss MT2 ProCo RAT MXR Distortion + Arbiter Fuzz Face E-H Big Muff
Mercuriall	Greed Smasher Metal Area TSC 808 Core	M-B Grid Slammer Boss MT2 TS808
Analog Obsession	Pig Pie Zupaa	E-H Russian Big Muff Vox Tone Bender
Nembrini Audio	BLACK Clon Minotaur 808 Overdrive	ProCo RAT Klon Centaur Ibanez TS808
Distorque Audio	Face Bender	Arbiter Fuzz Face Tone Bender MKII

These starting models are compared to those obtained via feature extraction: The closest model to the NA TS808 is the MPP OD1, the closest model to the NA RAT is the NA Klon Centaur, the closest model to the MPP Fuzz Face is the MPP MT2. For the general-purpose pretrained model selected using domain knowledge, we chose the NA Klon Centaur as it is known for its tonal transparency that does not colour the signal [13]. The general model obtained via feature extraction is the MPP MXR distortion emulation.

2.1.2 Feature Extraction

We utilise another form of transfer learning for the feature extraction method presented here. A pretrained CNN designed for distortion effects recognition is used to extract features of the training data for the source model choice. This approach bears similarities to other such methods employed in MIR, examples of which were presented in Section 1.

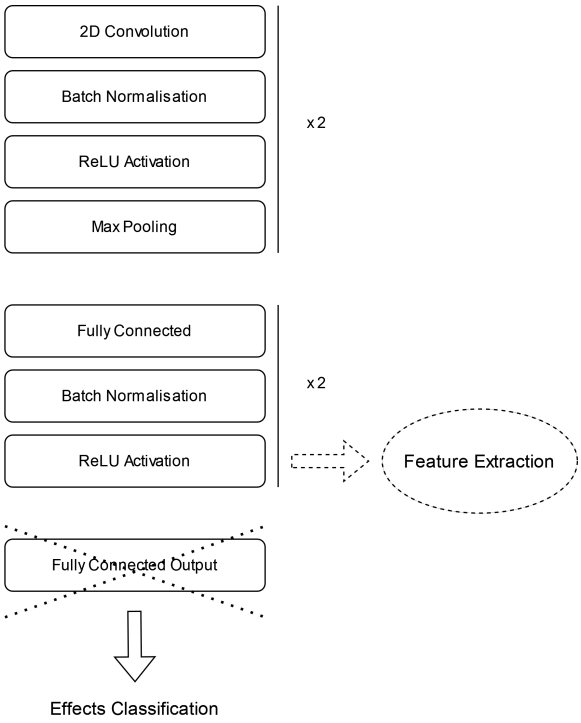


Fig. 2: *The FxNet architecture from [12] adapted for feature extraction.*

The pretrained CNN used in our experiments is the FxNet which comprises two convolutional and three fully connected layers, with batch normalisation at each level. The FxNet model is trained specifically on a number of distortion effects for their classification. We adapt FxNet by removing the output classification layer, which consists of a fully connected layer used to map the learnt feature representation to a vector whose size equals the number of effects in their dataset, as illustrated in Figure 2. The details of the adapted FxNet architecture are presented in Table 2. FxNet takes as input mel power-spectrograms. We extracted these spectrograms from various parts of the recorded signals in our dataset using the Librosa library [14].

We now detail the methods used to determine the closest and general models. We have a full training dataset S_i for each audio effect $i \in N$ listed in Table 1, where N is the total number of audio effects. We denote $\mathbb{D}$ the collection of each of these source datasets $S_i, i \in N$.

Table 2: *Feature Extraction CNN (FxNet) adapted from [12]. #Fmaps is number of feature maps output from the convolutional layers.*

Layer	Size	#Fmaps	Activation
2D Convolution	5×5	6	Linear
Batch Norm.	···	···	···
Activation	···	···	ReLU
Max Pooling	2×2	···	···
2D Convolution	5×5	12	Linear
Batch Norm.	···	···	···
Activation	···	···	ReLU
Max Pooling	2×2	···	···
Fully Connected	120	···	Linear
Batch Norm.	···	···	···
Activation	···	···	ReLU
Fully Connected	60	···	Linear
Batch Norm.	···	···	···
Activation	···	···	ReLU

We apply the modified FxNet model to each source dataset to extract the features $F(S_i)$ of each dataset S_i, as illustrated in Figure 2. We then choose the i^{th} audio effect to be our target effect, and the S_j providing the smallest distance as the source data for the pretraining of the closest model:

$$\underset{F(S_j)}{\arg\min} ||F(S_i) - F(S_j)||_2, \quad j \neq i \tag{1}$$

The target dataset T_i corresponding to the i^{th} audio effect is chosen depending on the fine-tuning method.

Similarly, for the general model the same feature extraction process is applied to obtain all the features of each dataset in $\mathbb{D}$. We then compute an average feature vector:

$$F(A) := \frac{1}{N} \sum_{j=1}^{N} F(S_j) \tag{2}$$

For each $S_j \in \mathbb{D}$, we then compute the Euclidean distance between each potential source dataset features and the average feature vector. The closest feature vector $F(S_j)$ to the average $F(A)$ gives us the source data S_j used to train the general model.

$$\underset{F(S_j)}{\arg\min} ||F(A) - F(S_j)||_2 \tag{3}$$

Again, the target dataset T_i corresponding to the i^{th} audio effect is chosen depending on the fine-tuning method, a number of which are also evaluated in this work.

2.2 Fine-Tuning

In the fine-tuning phase, we adapt the pretrained models to a new target effect by retraining on the target task data and optionally freezing certain network parameters.

2.2.1 Choice of Dataset

For this study, we assembled a dataset similar to the one proposed by Comunità et al. [12]. The input data comprises samples from the IDMT-SMT-Guitar dataset [15] and a number of artificial test signals including Dirac delta functions, step functions, exponential sine sweeps and sawtooth waveforms of varying amplitude. The guitar signals were all taken from the IDMT-SMT-Guitar dataset 4 and vary in genres and tempos. These signals were then processed using the same plugins as those used in Comunità et al. [12] and three additional distortion effect plugins, all of which are summarised in Table 1. All pretrained models were obtained using the whole dataset whereas the type and length of the data used for fine-tuning was varied in order to gauge its impact on target task performance.

We evaluate the impact of the type of training data employed for the fine-tuning by comparing the use of the artificial test signals to the recorded guitar and by comparing input data seen during pretraining to unseen data. All the data used for this comparison is of same length in order to isolate the impact of the signal content. We also study the impact of signal length on the fine-tuning accuracy in order to gauge a minimal amount of data necessary for this phase. Only the recorded guitar was used for this experiment, with length varying from five seconds to nine minutes.

2.2.2 Weight Freezing

It is common practice in transfer learning approaches to freeze certain weights of the pretrained model for the fine-tuning. We study the impact of weight freezing for the transfer by varying the layers whose weights are fixed in our example networks.

The recurrent model is fairly shallow and contains only two layers that can be frozen. Therefore we compare the impact of freezing the recurrent layer's weights, the fully connected layer's weights, and complete retraining. Yosinksi et al. observe experimentally that DNNs appear to exhibit more general features that are more

applicable to many datasets and tasks in their initial layers whereas more specific features tend to appear in the later layers [16]. Therefore for the example convolutional model we have chosen, we compare freezing the initial block to freezing all convolutional layers and to complete retraining.

3 Results

We now present the results of a case study using two illustrative architectures for audio effects modelling: A simple single-layer LSTM network with a fully connected output and 32 recurrent units; and the feedforward WaveNet from Wright et al. [11] with two blocks of ten convolutional layers with a dilation factor of two. Each convolutional layer has 16 filters and a kernel size of three. We evaluate various methods of TL using these two models on three different distortion effects using a variety of methods in order to encourage significance of the results presented here.

3.1 Choice of Pretrained Model

All pretraining was carried out using the entire dataset which comprises around ten minutes of guitar recordings and a number of artificial test signals as described previously. This dataset was split, using 90% for training and 10% for validation. Both the recurrent and convolutional models were trained using the same optimisation and data hyperparameters for a total of 50 epochs. The models are then fine-tuned using the same hyperparameters for another 50 epochs. In the case of the training from scratch, all models are trained on the fine-tuning data for 100 epochs. Mean Absolute Error (MAE) was used as the loss function for all training, as well as the Adam optimiser with an initial learning rate of 0.005 and weight decay of 1e-4. A batch size of 50 was used for the recurrent model and 40 was used for the convolutional one. All errors reported are also expressed in terms of MAE after training the models for 100 epochs and are presented in Table 3. All LSTM models showed an improvement when using transfer learning however the WaveNet models did not always present an improvement. In most cases, the general-purpose pretrained model outperforms other methods, with the feature extraction-based general-purpose model showing the lowest error most consistently.

3.2 Weight Freezing

In Table 4 and Table 5, we report the average test losses of the different weight freezing approaches again after 100 epochs training.

Freezing the weights of the recurrent layer of the LSTM model for fine-tuning leads to a significant increase in error. Freezing the fully connected layer of this model can lead to an improvement in accuracy. However, for the distortion emulation this freezing method leads to an overall decrease in performance. Retraining the models consistently leads to a decrease in test MAE for the LSTM model. Transfer learning rarely leads to decreased error for the convolutional models when compared to training from random initialisation for the same number of epochs. This could be due to the dataset size as increased labelled data or the use of data augmentation at fine-tuning has been found to reduce the effects of the knowledge transfer in certain convolutional architectures [17].

3.3 Subjective Evaluation

To better evaluate the impact of transfer learning on model performance for equal training length, we implement subjective listening tests using the Multiple Stimuli with Hidden Reference and Anchor (MUSHRA) methodology with the webMUSHRA framework [18]. In these listening tests, participants are asked to rate, out of 100, the similarity of audio extracts compared to a reference. The audio clips evaluated contain both simulations obtained with and without transfer learning, a hidden reference and two anchors. The MUSHRA methodology allows for statistical significance of results with fewer participants [19]. This is to assess whether the slight decreases in error observed with the best performing models in Tables 4 and 5 are audibly significant. A total of 40 assessors were used for the subjective evaluation. These assessors are a mix of audio professionals with experience in critical listening and guitarists.

The results are presented in Figures 3 and 4. Certain outliers were omitted from these results if the assessor assigned a score lower than 90 for the reference or higher than 50 for either of the anchors. For both the LSTM and WaveNet models, a relatively high variance in responses can be observed. This is most likely due to the fact that the figures show an aggregate of responses, evaluating all effects types including the fuzz effects

Table 3: *Final model MAE with respect to the choice of pretrained model for each class of distortion effect. Pretrained model chosen either using Domain Knowledge (DK) or via Feature Extraction (FE).*

	Overdrive		Distortion		Fuzz	
	LSTM	WaveNet	LSTM	WaveNet	LSTM	WaveNet
Closest Model *(DK)*	0.0307	0.0281	0.0381	0.0193	0.0744	0.0557
Closest Model *(FE)*	0.0297	0.0288	0.0260	0.0268	0.0817	**0.0518**
General Model *(DK)*	**0.0280**	0.0307	0.0260	0.0268	0.0719	0.0576
General Model *(FE)*	0.0290	**0.0267**	**0.0135**	0.0162	**0.0652**	0.0651
From Scratch	0.0308	0.0295	0.0278	**0.0146**	0.0902	0.0530

Table 4: *Average weight freezing MAE (LSTM).*

LSTM			
	Overdrive	Distortion	Fuzz
Freeze rec. layer	0.0512	0.2693	0.1765
Freeze dense layer	**0.0290**	0.0313	**0.0724**
Retrain	0.0293	**0.0259**	0.0773
From scratch	0.0308	0.0278	0.0902

which are known to be difficult to model with these approaches [20].

A Student's t-test was used over ANOVA to compare the means of the groups as there are only two present here. A one-sided test was used because we assume there will be an improvement in performance using TL for the alternative hypothesis. Regarding the LSTM model, the mean score of the TL simulations is 16% higher than that of the from scratch results with a p-value of 0.003 computed with Student's t-test from the SciPy Python library [21]. The listening tests back up the objective measures that showed improvements in simulation accuracy when using TL on LSTM. The lack of improvement seen in the WaveNet predictions is also reflected in the listening test results. The mean score sees a decrease of 2.4% when using TL but the p-value computed equals 0.57 and therefore this result was not deemed statistically significant. The listening tests back up the objective measures that showed

Table 5: *Average weight freezing MAE (WaveNet).*

WaveNet			
	Overdrive	Distortion	Fuzz
Freeze conv. layers	0.0396	0.1734	0.1558
Freeze first block	0.0311	0.0328	0.0887
Retrain	**0.0286**	0.0223	0.0581
From scratch	0.0295	**0.0146**	**0.0530**

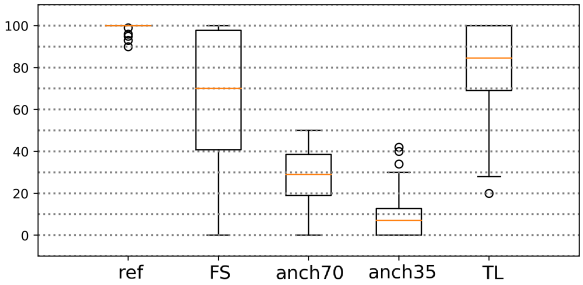


Fig. 3: *MUSHRA results for the LSTM model. The scores from the reference (ref), simulations obtained using Transfer Learning (TL) and training From Scratch (FS) as well as from the hidden anchors are shown.*

no improvements in accuracy when using TL on the WaveNet. The interquartile range for both simulations from TL and trained from scratch remains about the same, with the median seeing a 10% decrease with TL.

3.4 Fine-Tuning Data

We then evaluate what data to use for the fine-tuning phase, both in terms of signal content and dataset length. All results presented hereafter were obtained when fine-tuning the general-purpose model chosen via feature extraction without weight freezing as this provided the most consistent improvements in the previous experiments. In Table 6, we compare two subsets containing recorded guitar with a dataset containing exclusively artificial test signals. These results were obtained by retraining the models during fine-tuning as this is the method that performed best for both example models in the previous experiments. In almost all instances, except the distortion emulation using the WaveNet model, the guitar datasets outperformed the use of test signals. Moreover, in all

Table 6: *Fine-tuning dataset type. All datasets are of equal length (37 seconds).*

Signal Type	Overdrive		Distortion		Fuzz	
	LSTM	WaveNet	LSTM	WaveNet	LSTM	WaveNet
Recorded Guitar *(seen at pretraining)*	0.0452	0.0301	0.0758	0.1015	**0.0886**	0.0853
Test Signals *(seen at pretraining)*	0.0656	0.0655	0.2202	**0.0480**	0.1133	0.1466
Recorded Guitar *(unseen at pretraining)*	**0.0305**	**0.0290**	**0.0344**	0.0583	0.0958	**0.0772**

Table 7: *Fine-tuning dataset length. The from scratch results were trained on the full guitar dataset, lasting 546 seconds. Training times for LSTM recorded on an Nvidia GeForce GTX 1650 GPU, training times for WaveNet recorded on an Nvidia GeForce RTX 2060 GPU.*

Signal Length (s)	**Overdrive**		**Distortion**		**Fuzz**		**Training Time(s)**	
	LSTM	WaveNet	LSTM	WaveNet	LSTM	WaveNet	LSTM	WaveNet
5	0.0577	0.0534	0.1451	0.0762	0.1457	0.1369	0.51	3.66
30	0.0359	0.0299	0.0767	0.1081	0.1378	0.1098	2.94	19.97
210	**0.0286**	0.0252	0.0355	0.0328	0.0803	0.0788	21.09	190.89
540	0.0301	**0.0225**	**0.0126**	0.0264	**0.0582**	0.0559	42.83	396.85
From Scratch	0.0308	0.0295	0.0278	**0.0146**	0.0902	**0.0530**	57.80	381.28

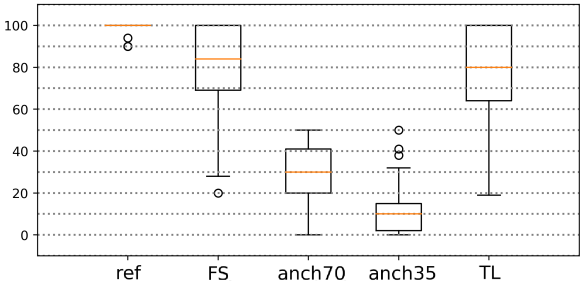


Fig. 4: *MUSHRA results for the WaveNet model. The scores from the reference (ref), simulations obtained using Transfer Learning (TL) and training From Scratch (FS) as well as from the hidden anchors are shown.*

cases except one, training on unseen data outperformed training on a subset of the source data.

Our final experiment aims to find a trade off between training time and model accuracy. All models obtained via transfer learning were fine-tuned on recorded guitar signals of various lengths. We compare these to from scratch training on the full guitar subset in Table 7.

For the simpler emulation task of the overdrive effect, we obtain comparable error for both the recurrent and convolutional models when using transfer learning on a significantly reduced dataset. We go from full training on nine minutes and six seconds of data to 30 seconds resulting in a decrease in training time by around 95% for both the LSTM model and WaveNet for a slight increase in MAE. For more complex effects, the increase in MAE is more pronounced. It is still possible to cut training time in half for a maximum increase in error of 2.6%. Moreover, transfer learning for the fuzz emulation using LSTM leads to a 1% decrease in error while reducing training time by over 50%.

4 Conclusion

In this paper we studied the impact of transfer learning and its implementation for distortion effects modelling. We evaluated this approach on three example modelling tasks, using two illustrative networks. Transfer learning shows a slight decrease in error for the recurrent model compared to training from scratch for an equal number of epochs. However, the convolutional network did not show any significant improvements. This was also reflected in the subjective results from listening tests. Using a general-purpose pretrained model obtained via feature extraction presents the most consistent improvements in performance. It was found that fine-tuning on guitar recordings unseen during pretraining lead to the best performance and significant model training speed-ups can be achieved using for minimal to no increase in error.

Appropriate weight freezing should be employed in order to avoid any increase in error. Namely, for consistent improvements, complete retraining is preferred. In

order to increase the statistical significance of these results, it would be of interest to rerun these experiments a number of times while varying the random seed. This case study could also be extended to other audio effects, especially those presenting different audio behaviour as we have constrained ourselves to distortion effects in this work.

Finally, extending this work to parametric models that incorporate user-facing controls is of great interest as the benefits of transfer learning are especially prominent in this case.

5 Acknowledgments

This research was hosted by the SCRIME of the University of Bordeaux (Studio of Creation and Research in Computer Science and Experimental Music) which is funded by the french ministry of culture. Experiments presented in this paper were carried out using the PlaFRIM experimental testbed, supported by Inria, CNRS (LABRI and IMB), Université de Bordeaux, Bordeaux INP and Conseil Régional d'Aquitaine (see `https://www.plafrim.fr`).

References

[1] Pan, S. J. and Yang, Q., "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, 22, pp. 1345–1359, 2010, ISSN 10414347, doi:10.1109/TKDE.2009.191.

[2] Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., and Liu, C., "A Survey on Deep Transfer Learning," in *Artificial Neural Networks and Machine Learning – ICANN*, 2018.

[3] Yu, F., Xiu, X., and Li, Y., "A Survey on Deep Transfer Learning and Beyond," *Mathematics*, 10, 2022, ISSN 22277390, doi:10.3390/math10193619.

[4] Hamel, P., Davies, M. E. P., Yoshii, K., and Goto, M., "Transfer Learning in MIR: Sharing Learned Latent Representations for Music Audio Classification and Similarity," in *Proceedings of the 14th International Society for Music Information Retrieval Conference*, pp. 9–14, 2013.

[5] Oord, A. V. D., Dieleman, S., and Schrauwen, B., "Transfer Learning by Supervised Pre-Training for Audio-Based Music Classification," in *Proceedings of the 15th International Society for Music Information Retrieval Conference*, pp. 29–34, 2014.

[6] Liang, D., Zhan, M., and Ellis, D. P. W., "Content-Aware Collaborative Music Recommendation Using Pre-Trained Neural Networks," in *Proceedings of the 16th International Society for Music Information Retrieval Conference*, pp. 295–301, 2015.

[7] Choi, K., Fazekas, G., and Sandler, M., "Towards Playlist Generation Algorithms Using RNNs Trained on Within-Track Transitions," in *Workshop on Surprise, Opposition, and Obstruction in Adaptive and Personalized Systems*, 2016, doi:10.1145/1235.

[8] Choi, K., Fazekas, G., Sandler, M., and Cho, K., "Transfer learning for music classification and regression tasks," in *Proceedings of the 18th International Society for Music Information Retrieval Conference*, pp. 141–149, 2017.

[9] Alfaro-Contreras, M., Valero-Mas, J. J., Iñesta, J. M., and Calvo-Zaragoza, J., "Insights Into Transfer Learning Between Image and Audio Music Transcription," in *19th Sound and Music Computing Conference*, pp. 295–301, 2022.

[10] Juvela, L., Damskägg, E.-P., Peussa, A., Mäkinen, J., Sherson, T., Mimilakis, S. I., Rauhanen, K., and Gotsopoulos, A., "End-to-End Amp Modeling: from Data to Controllable Guitar Amplifier Models," in *2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5, Institute of Electrical and Electronics Engineers, 2023, doi:10.1109/icassp49357.2023.10094769.

[11] Wright, A., Damskägg, E. P., Juvela, L., and Välimäki, V., "Real-time Guitar Amplifier Emulation with Deep Learning," *Applied Sciences (Switzerland)*, 10(3), 2020, ISSN 20763417, doi:10.3390/app10030766.

[12] Comunità, M., Stowell, D., and Reiss, J. D., "Guitar Effects Recognition and Parameter Estimation With Convolutional Neural Networks," *Journal of*

the Audio Engineering Society, 69, pp. 594–604, 2021, ISSN 15494950, doi:10.17743/jaes.2021.0019.

[13] World, G., *Guitar World Presents 200 Stompbox Reviews*, Hal Leonard, 2014, https://www.halleonard.com/product/123825/iguitar-worldi-presents-200-stompbox-reviews.

[14] McFee, B., Raffel, C., Liang, D., Ellis, D. P., McVicar, M., Battenberg, E., and Nieto, O., "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th python in science conference*, volume 8, 2015, https://librosa.org/.

[15] Kehling, C., Männchen, A., and Eppler, A., "IDMT-SMT-Guitar," Technical report, Fraunhofer Institute for Digital Media Technology IDMT, 2014, https://www.idmt.fraunhofer.de/en/publications/datasets/guitar.html.

[16] Yosinski, J., Clune, J., Bengio, Y., and Lipson, H., "How transferable are features in deep neural networks?" in *Advances in Neural Information Processing Systems 27 (NIPS '14)*, 2014.

[17] Zoph, B., Ghiasi, G., Lin, T.-Y., Cui, Y., Liu, H., Cubuk, E. D., and Le, Q. V., "Rethinking Pre-training and Self-training," in *34th Conference on Neural Information Processing Systems (NeurIPS 2020)*, 2020.

[18] Schoeffler, M., Bartoschek, S., Stöter, F.-r., Roess, M., Edler, B., and Herre, J., "webMUSHRA — A Comprehensive Framework for Web-based Listening Tests," *Journal of Open Research Software*, 6, p. 8, 2018.

[19] ITU-R, "ITU-R BS.1534-3: Method for the Subjective Assessment of Intermediate Quality Level of Audio Systems," *Intern. Telecom. Union Radiocom. Sector*, BS Series, p. 34, 2015.

[20] Comunità, M., Steinmetz, C., Phan, H., and Reiss, J., "Modelling Black-Box Audio Effects with Time-Varying Feature Modulation," pp. 1–5, 2023, doi:10.1109/ICASSP49357.2023.10097173.

[21] Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, 17, pp. 261–272, 2020, doi:10.1038/s41592-019-0686-2.

Audio Engineering Society

Conference Paper 9

Presented at the AES International Conference on Machine Learning
and Artificial Intelligence for Audio
2025 September 8-10, London, UK

This paper was peer-reviewed as a complete manuscript for presentation at this conference. This paper is available in the AES E-Library (http://www.aes.org/e-lib), all rights reserved. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Journal of the Audio Engineering Society.

Perceptions of an Artificial Intelligence Musical Collaborator

Becky Allen[1], Ronald Mo[1,]

[1]University of Sunderland

Correspondence should be addressed to Becky Allen (Becky.Allen@sunderland.ac.uk)

ABSTRACT

Generative Artificial Intelligence (GenAI) has the potential to change the landscape of music creativity through new creative practices and tools to assist in all aspects of music making. However, the acceleration of this technology has also prompted concerns relating to a range of ethical issues. Discourse surrounding AI within music has included limited consultation with creative practitioners surrounding potential disruption to working practices. This study centres the concept of 'musicking' to identify participants to gather data pertaining to views of Generative AI within music. Findings indicate that some musicians are already incorporating AI within their practice as a creative assistant. However, a clear mental model of how AI works and how this can be ethically implemented within practice is lacking, potentially exacerbating concerns around AI adoption. Embedding a human-centred AI approach to the development of AI music tools, through stakeholder engagement and alignment with existing music technology may harness the positive potential of AI without devaluing musicianship.

1 Introduction

Generative Artificial Intelligence (GenAI) is at the forefront of development within AI and has been a main driver pertaining to the awareness of this technology. Advances such as ChatGPT [1] have brought discourse surrounding this technology firmly into the public domain when in 2022 it was made widely available. GenAI utilises deep learning to generate artifacts using existing content including video, images, audio and text by analysing examples in the training dataset and learning their distribution and patterns [2]. GenAI is gaining traction within audio generation, for example AIVA [3] is an AI music generation assistant allowing users to generate compositions encompassing a range of genres. Other examples of this technology include AudioCraft which generates audio based on user text input [4], MuseNet [5] and Suno [6]. These systems often operate on either a text-to-music-based system or through the selection of pre-specified criteria such as genre. The ability to customise or augment the output is currently limited.

The focus on generative audio/ music by big tech corporations such as Meta and OpenAI (e.g. AudioCraft, MuseNet) demonstrates commercial viability in such technology. However, the impact the ubiquity of such technology may have on the creative community if wider adoption is enacted may be far-reaching. Implications of the cultural impact and potential ethical issues are relatively unexplored, as are the views of both creatives and music consumers.

Determining perceptions of this technology is key to gaining an understanding of its infiltration within the music community and potential misconceptions of AI. Understanding concerns relating to the inclusion of AI within music is crucial to examining how this technology can be made more equitable. A human-centred approach to AI (HCAI) is gaining prominence as a potential way to embed a more ethical approach.

This approach endeavours to build "intelligent systems that are designed with social responsibility in mind," encompassing an understanding of humans and assisting humans in understanding AI [7]. Therefore, to be able to adopt a HCAI approach, an understanding of individuals' thoughts on this technology is essential.

The perspectives of both musicians and music enthusiasts are core to this research to determine any heterogeneity in opinion. For the purpose of clarity we define a musician as "a person talented in the art of music" [8]. This definition encompasses professionals who may undertake this as a job role as well as individuals who play instruments and compose as a hobby. We define a music enthusiast as a person who "pursues an activity or objective with passion or zeal" [9]. Both of these terms encompass 'musicking'[10] and will be used throughout the paper. The study aims to address the following research questions:

RQ1: What are the perceptions of an AI musical collaborator from the perspective of both musicians and enthusiasts of music?
RQ2: What are the main concerns regarding the integration of AI within a creative musical context?
RQ3: Does the inclusion of AI within the composition process affect musician's and enthusiast's perceptions of the music?

To address the questions, a rigorous methodology has been implemented to ensure a consistent approach to knowledge discovery. Determining existing research relating to these questions is core to appreciate the current state of the art relating to this domain.

2 Background

2.1 Existing Generative AI Systems

GenAI models can create high-quality artefacts based on their training data [11], including images, text and audio. It can be identified as a subdomain of deep learning due to foundational algorithms such as Generative Adversarial Networks (GANs) [12] and Variational Autoencoders (VAEs) [13]. Diffusion models, transformers and riffusion have also demonstrated promise in relation to symbolic and structured music generation [14], [15], [16].

There are a range of existing commercial GenAI music options, including MusicGen [4], MuseNet [5], Loudly [17] and Soundraw [18]. These tools have limited marketisation towards musicians, mainly for users who require functional music to accompany their content (e.g. games, social media, etc). Most systems generate the composition based on text input from the user outlining the required genre and other high level descriptive aspects such as tempo. There are limited genre options, which may have adverse implications for the diversity of music being produced. Specific plug-ins are also readily available for integration into a user's digital audio workstation (DAW), examples include Magenta Studio [19] and Orb Producer [20]. These provide users with greater control and editability of the generated material, for example Drumify within Magenta Studio can "create grooves based on the rhythm of any input" [19], assisting the musician in composition if they do not have the pre-requisite skills.

This technology presents both positive and negative challenges from a societal and ethical perspective [21]. Current approaches to the development of GenAI often fail to recognise the inherent human activity of music making which encompasses "sociological, historical and physio-psychological dimensions" [22]. There is a need to embed a more sociotechnical approach, potentially through integration of stakeholder engagement which is currently lacking [22], [23]. Through stakeholder engagement, comprehension of the artistic process can be understood by system developers. Co-creation can assist musicians in developing a cognitive model of AI, which has been shown to engender trust and encourage integration into creative practice [23], [24]. If GenAI is used critically by musicians, they may hone a specific craft in applying AI to music, creating a shared "creative musicking intelligence" [24].

2.2 Current Use of AI by Musicians

Determining the extent to which musicians currently use AI is core to comprehend the acceptance level. A study by Ditto Music surveyed 1299 artists who were actively producing and releasing music to determine the extent to which they employed AI [25]. 59.5% of respondents already use AI within their music, most commonly for production and mastering purposes. 11% of respondents used this for songwriting. From the artists surveyed who currently do not use AI, 47.1% said they would use it for songwriting, 61.5% said they would use it for production and 66% for mastering. Only 28.5% of participants responded that they would never use AI. Artists disclosed they were dissuaded from using this technology through a lack of access to AI tools and the lack of creativity [25]. This survey indicates that some artists are already

employing AI within their creative process and those that currently aren't may be open to its inclusion. Issues raised pertaining to AI access and perceived lack of creativity of these technologies pose interesting questions relating to the level of awareness and understanding of AI.

A survey of 1141 artists from across the UK, US and Germany discovered a growing embrace of AI [26]. However, this study revealed that a proportion of musicians who use AI within the creative process would be concerned about disclosing this to their audience for fear of the perception of a "loss of authenticity" [26]. Participants were asked at which stage of their practice AI was the most useful, responses included "songwriting & composition" with participants describing it as a "jumping off point" [26].

2.3 Human-Centered AI

Concomitant to the increase in AI system use is the rise in the frequency of misuse [27]. Poor ethical practices not only erode trust in this technology but lead to discriminatory outcomes through the perpetuation of bias. To attempt to mitigate against unethical practices, a HCAI approach to this technology has been proposed, centring the context of social responsibility and value-guided approaches, alongside core computational approaches [28].

Data ethics is a core concern within GenAI, particularly issues around copyright. Music publishers including Universal Music sued Anthropic, accusing the tech firm of misusing copyrighted song lyrics to train its chatbot [29]. Major record companies have also sued AI music start-ups Suno and Udio for training their systems using the labels' recordings without permission [30]. Some artists, however, have embraced this technology. The Beatles utilised AI to work on previously unheard material, resulting in the release of Now and Then [31]. AI was used to clean and enhance John Lennon's voice to achieve "a cleaner and more refined sound" [32]. The use of this technology as a tool to assist in the achievement of artistic outcomes emphasises the benefits of AI when utilised for a specific purpose.

HCAI systems designed for co-creativity within music have highlighted the role of AI as a musical collaborator, users personified AI as a "second person" to bounce ideas off [23] and assisted in the development of new ideas [33]. However, adoption into practice can be difficult with some users becoming overwhelmed by the amount of content generated and the non-deterministic nature of this output [34]. However, other musicians appreciate this as it allows them to "generate a large repository of content to remix" [23]. This juxtaposition of views reflects the personal process of composition and the importance of a co-creative approach ensuring the musician can utilise AI to suit their practice.

3 Methodology

To answer the research questions, interviews and focus groups were conducted with participants. A core set of questions were asked in both methodologies to ensure consistency in approach. Ethical approval was certified by authors institution.

3.1 Geographical Context

The study is situated within the North-East of England consisting of Tyne and Wear, County Durham, Northumberland and Tees Valley. The region has undergone extensive culture-led regeneration through major investment in music and performance venues [35],[36]. The musical landscape within the North-East has a rich heritage from border ballads and progressive folk music to modern day rock and indie bands. A range of music making activities and formal educational courses are offered in both music and music technology at varying educational institutes in the region. The city of Sunderland has recently been awarded Music City status in recognition of its transformative impact [37].

3.2 Participant Recruitment

Participation in the study is limited to individuals situated within the North-East who are either music enthusiasts or professional musicians. Encompassing this approach is the concept of musicking. The term 'musicking' or '*to music'* describes the action of participating in any capacity, in a music performance, either by performing, listening or composing [10]. Gathering the opinions of musicians and music enthusiasts is core to building an understanding of a range of views. Purposive sampling was used to recruit participants for interview; self-selection sampling was used for the focus groups. Financial incentives were not provided. Participants were requested to complete a questionnaire pertaining to demographic information, including age and gender (categories included female, male and prefer to self-describe). Participants were also asked whether they play a musical instrument, their years of experience and whether they compose music. A factsheet was

provided to all participants to enable a baseline understanding of AI.

3.3 Interviews

Semi-structured interviews were deemed the most applicable format as they are based around a set of topics or pre-set questions, however, sequencing is participant led [38] with scope for additional discussion. The interview guide enabled a clear structure, ensuring that the questioning aligned with the research aim. Ensuring that the interview guide had similarity with the focus group guide (see section 3.4) enabled parity in terms of consistency of approach. Interviews lasted approximately one hour. The interview commenced with a question to determine the participant's overarching view of AI. Questioning turned to participant's views of the use of AI within music, including whether they have ever used it. More specific questions were then asked regarding an AI musical collaborator and whether participants would use this and within what context. For participants who are not musicians, questioning included whether an AI musical collaborator would encourage them to attempt music making. All participants were asked whether the use of AI within music composition would alter their opinion of the product and if they had any overarching concerns.

3.4 Focus Group

To widen participation in the study, two focus groups were completed, lasting approximately 90 minutes, enabling individuals often excluded from academic research to participate. A guide for the focus groups ensured that the discussion remained pertinent to the research questions whilst also allowing exploration of topics of interest. Pre-prepared questions mirror those within the interviews to ensure a consistent approach.

3.5 Analysis

Transcription of all audio recordings were completed. Thematic analysis used a reflexive approach [39] and included iterative independent coding to identify patterns of shared meaning and initial themes which were then subjected to inter-rater reliability analysis. This compared the themes generated by each author to determine the percentage agreement around thematic generation.

4 Results

Fourteen individuals participated in the research, consisting of 7 interview participants and 7 focus group participants. Focus group 1 consisted of 4 participants and focus group 2 had 3 participants. Table 1 contains participant demographics. Participants 4, 13 and 14 did not complete the demographic questionnaire. Based on the 11 participants who provided demographic information, 7 identified as male and 3 as female (1 participant did not answer). Nine respondents play a musical instrument and 10 compose their own music. The results will be presented based on the themes generated from both interviews and focus groups. Due to the use of a subject guide for both data collection methods, the results have been collated by theme. The inter-rater reliability analysis score between authors was 95.3%.

Participant (P)	Gender	Age	Play Musical Instrument?	Which Instrument?	Years of Experience?	Compose Music?
1	M	64	Y	Trumpet, Piano	50	Y
2	F	54	Y	Keyboard, Vocals	"Since childhood"	Y
3	-	-	Y	Piano, Bass Guitar	33	Y
5	F	39	Y	Vocals, Piano, Guitar	25	Y
6	F	34	N	-	-	N
7	M	37	Y	DJ, Music Production	20	Y
8	M	29	Y	DJ, Music Technology	11	Y
9	M	24	N	-	7	Y
10	M	54	Y	Guitar	10	Y
11	M	-	Y	Piano, Drum Machine	23	Y
12	M	46	Y	Guitar, Keyboard, Drums	35	Y

Table 1. Participant Demographics

4.1 AI Literacy

Six out of the 14 participants made a comment aligned with a lack of AI literacy and how this affected their feelings towards AI. P1 stated that they were not *"massively informed*", P4 stated that they had "*limited interaction*" and P14 advised that they were "*new to it all*". Two participants reserved overall judgement on AI as they felt they didn't fully understand "*its capabilities*" (P5), or what constitutes AI due to the wealth of tools /plugins that are available within DAWs (P9). P7 raised the wider need for AI literacy, highlighting the need to demonstrate how these tools can be used in a positive way.

4.2 Opinions on AI

Overarching views of AI were mixed. Participants noted that they were "*excited by it*" (P7), "*very positive*" (P10) and they thought it was "*quite good*" (P9). P14 stated that "*in terms of the creative sides of AI I think it's cool*". Not all participants were positive, P2 stated that they thought it was "*impressive, scary*" and P8 advised that they thought it was "*about accepting it*". This passivity and feeling of

inevitability was mirrored around concerns relating to the control and regulation of AI (see section 4.4). Three participants had used AI within their practice; the majority had not. P13 advised that they would not use AI as it did not match their musical "*aesthetic,*" mirrored in a similar response by P14 in that they would not use it for recording or live performance as they'd rather "*have it done by a person*." This viewpoint aligns with other participants in terms of perceived authenticity (see section 4.7).

Three participants expressed negative opinions, 2 of these related to the loss of skills due to the potential ease of music composition. P5 advised that this could lead to missed opportunities in learning especially as "*experimenting brings the joy*". P11 expressed concerns around devaluing musicians and diversity of skills, "*specific instruments have specific methods of playing and it would be sad to see them go*." Along a similar vein P14 surmised that AI in music may be the "*death of originality*" due to the lost nuance of human skills and experience.

4.3 Economic Concerns

Relating to RQ2, one main concern from participants was the economic impact AI may have on musicians, particularly the exploitation of small artists (P6), loss of jobs (P14, P6, P7) and overall shift in the economics of music (P1). P1 discussed how AI may "*yield alternative and different forms of economics*." Concerns were also raised around how the music industry may adopt AI to the detriment of artists: "*some people are just focused on the money so they would not care about the quality of the music*" (P11).

4.4 AI Ethics

A variety of concerns were raised concomitant to common ethical issues, including algorithmic explainability, bias, diversity and the requirement for greater regulation. Relating to explainability, AI was described by P2 as "*powerful systems that we don't really understand*", tying to issues around AI literacy. Bias and diversity were raised by 5 participants. P5 and P6 both raised concerns around the lack of diverse datasets used to train AI algorithms, particularly concerns relating to "*appropriation, the cultural sensitivities*" associated with specific genres and how AI may not be attuned to this sensitivity. P2 was worried that AI would reduce "*the diversity we have in music*" as a consequence of generic datasets. P6 was concerned not only with the impact of musicians within a geographical area but the ecology of that location due to how intrinsic music is to the cultural identity of a city. On a wider scale, P5 discussed the broader cultural shift in the value of music and how they feel AI may accelerate this trajectory of devaluing both music and musicianship "*in education and culturally*" (P5).

Concerns were raised around the balance of control between humans and AI "*so that you still feel relevant"* as a musician (P3). A HCAI approach with intrinsic stakeholder engagement may mitigate this concern if musicians see the benefit of AI to their craft. Concerns around regulation were raised, particularly how technology has progressed faster than "*humans have been able to put parameters around*" (P6). Control of music "*ownership*" (P5) and the lack of acknowledgment for artists whose work has been featured within GenAI training datasets and replication of their work (P6, P7) were also raised.

4.5 Historical Context of Music Technology

Six participants contextualised their perceptions of AI in music by discussing the broader context of music technology. Examples provided of technological developments which were initially perceived negatively included synthesizers (P1, P2), sampling (P14) and studio advancements including Pro Tools [40] (P4, P3). Respondents discussed how there has generally been "*scaremongering around new technology or new forms of music*" (P7) however, innovation has not always been the "*threat to authenticity that it's purported to be"* (P1) often leading to innovative forms of creativity.

4.6 Accessibility

AI was thought of positively in terms of increasing accessibility to music making with 7 participants highlighting their enthusiasm for this use case. P14 was particularly effusive around the potential for AI to "*democratise music making*", making it easier for someone to "*find their creative voice and express themselves"* a view echoed by P5, P6 and P9 who advised that AI "*opens your mind to new things*". These participants discussed the benefits of AI for young people to see themselves "*as musical humans*" (P6), allowing them to make an informed choice about the kind of music they might want to make and lowering the skills barrier to participation. Two participants (P2, P14) working in youth music discussed how technology such as GarageBand [41] had assisted people to "*create a tune in half an hour using the presets*" assisting them to express themselves creatively. P14 was positive about how they have already engaged with AI in their practice helping them to overcome deficits in their skills. They discussed using the piano player in Logic [42],

inputting chords they can't play on guitar and "*the AI piano player will generate a chord progression for me*" opening up a range of musical possibilities. This participant discussed how AI had inspired them to "*create something new and original, something that might have been out of reach to me a few years ago."* Although accessibility was identified as a positive benefit of GenAI, respondents P9, and P14 advised that it should be used as a tool to widen participation and should not be solely relied upon.

4.6.1 Efficiency

Seven participants discussed how they had integrated AI into their workflow, highlighting that it has "*helped them to do things*" (P4), reduced their workload (P9) and are happy for AI to "*make my life easier*" (P6). From the respondents' experiences the main motivating factor for AI use was as a time-saving tool, making tasks such as mixing easier. Four participants (P9, P13, P4, P3) discussed the use of AI in providing creative inspiration, for example, using ChatGPT [1] to assist with lyric writing (P4) or helping with melody generation (P4, P13).

4.7 Authenticity

Relating to RQ3, 13 out of 14 participants expressed concerns relating to the authenticity of music created with AI. P9 discussed the importance of imperfections within music advising that "*something that's created by AI would be too robotic to have that human feel.*" Music as an inherently human experience was raised by 4 participants, who discussed how important musicians' "*own lived experience*" was to the connection with the listener (P1). A lack of "*feel*" and "*nuance*" within AI-generated music was also raised restricting emotional connection (P13, P4, P14). P2 had stronger views in that they believe "*it's only humans who can create*". P2, P6 and P13 discussed the importance of collaboration with other musicians and how this is a core part of their process which they felt AI would not be able to replicate, advising "*having music in common is special*," cultivating the "*golden thread of connection*" (P13) between musicians and listener. Four participants highlighted the importance of a HCAI approach to music. P10 advised that human input is required to capture the essence of music and "*not everything should be 100% AI*", this was also discussed by P3 and P9 who felt the technology was acceptable for creative use if blended with human input. P6 discussed that they would not use AI for aspects of their composition process as it depended on "*where you put the value in the creative"* and they feel like the "*nuance is lost when you take it away from the human*." P5 anthropomorphised AI as a "*well-read human*" and aligned AI outputs to that of humans in that everything we create is a product of what we consume, drawing parallels to how GenAI algorithms work.

The impact AI may have on devaluing musicianship was discussed by 4 participants (P5, P3, P12, P14). This issue centred around concerns that people may become "*lazy*" (P5) and use AI-based tools to develop songs instead of gaining new skills and incorporating it into their practice. This is juxtaposed against discussions in section 4.6 around AI democratising music making.

4.8 Innovation to Creative Practice

Five participants (P1, P6, P9, P13, P14) advised that they thought AI could help them innovate within their practice. P1 discussed how they would be interested in combining human and AI-generated voices to explore potential "*friction*" between the two. P13 and P14 advised that they would be happy to use AI if it suggested different creative directions. P6 was intrigued by the idea of AI as a collaborator and how this would support musicians to collaborate differently. P14 already uses AI and discussed how it allowed them to explore ideas outside of their regular lexicon. P7 raised the idea that "*AI might create new forms of music*" as it allows musicians to diversify their practice.

4.9 AI as a Tool

The use of AI as a tool to assist with specific aspects of music composition was discussed by 9 participants (P1, P4, P9, P10, P13, P3, P12, P6, P2). P1 aligned this with other forms of music technology (see section 4.5), that new tools often "*present challenges but they seem to add a dimension to creative practice rather than subtracting*." P4 advised that they saw AI the same as "*any sound creation or manipulation tool*" in that it is something to create and experiment with. Three of the participants (P9, P10, P12) highlighted the importance of the use of "*elements*" of AI (P9), using it as a "*helpful suggester*" to which the musician can add their own personal input (P12). P6 and P2 highlighted the importance of AI literacy, advising that people need to understand how to use AI as a tool and "*not as a replacement*" (P6), assisting people to understand AI's limitations (P2).

5 Discussion

Determining perceptions of an AI musical collaborator is fundamental to understanding how this

technology may be designed and adopted within creative practice. This study endeavoured to initiate this conversation within a specific geographical location to determine views from a cohort of music enthusiasts. Although participant numbers were not as high as desired, the research methodology elicited a rich source of data in which to commence this form of inquiry. Although demographic information was collected, due to the number of respondents, the data was too limited to determine any differences of views in consideration of gender, age or musicianship.

5.1 RQ1: Perceptions of an AI Musical Collaborator

AI literacy appeared to be a large determinant of participants' views of AI, with respondents not feeling informed enough to make a definitive judgement. A lack of an accurate mental model of AI can also impact propensity for experimentation and user-self efficacy [34]. Therefore, transparency and intuitive interfaces may encourage more people to interact with such technology.

Views of AI in music were mixed. Passive or neutral opinions were determined by particular use cases. However, some participants were positive and keen to explore this further, highlighting that there are particular creative tasks musicians feel comfortable integrating AI [23]. Negative opinions related to concerns around diversity, economic impact, devaluing of musicianship and the 'quality' of music. Six of the participants contextualised their views of AI in music to other forms of music technology such as synthesizers, sampling and recording advances. Within this context P7 discussed how there were often negative views or "*scaremongering*" around these innovations, however these advances led to new forms of creativity. Accessibility was raised as a positive use case for AI in music regarding the potential democratisation of music making, lowering the skills entry barrier and widening participation.

Only 3 participants had specifically used AI within their practice, mainly to increase efficiency within the production process or to extend their current abilities. Four participants discussed how they may use it to provide inspiration for lyric and melody generation. These use cases align with existing research discussed in Section 2.2 in that musicians commonly use AI for production and as a starting point for composition. Seven participants discussed how the main motivating factor for AI use was efficiency or assistance, freeing musicians to focus on the larger creative tasks [43]. Antithetical to the concerns around loss of skills raised in section 4.7, were the points raised by 5 participants who advised that they thought AI may allow them to innovate within their creative practice. Examples included playing with the combination of human and AI voices to explore the juxtaposition (P1), to suggesting new creative directions through exploration of ideas outside of their usual style.

Use of AI as a tool was a recurring theme. Respondents aligned AI with the use of other music technologies such as sound creation or manipulation tools. However, this line of thinking came alongside the caveat that the musician should still have overall creative control, and AI should not be used as a replacement for human skills. Therefore a HCAI approach to a AI musical collaborator is core, with stakeholder engagement embedded throughout the design process, ensuring that musicians preferred methods of AI interaction fit their expectations [44].

5.2 RQ2: Concerns Regarding Integration of AI within Music

AI ethics was a central theme, core concerns such as explainability, bias and control were raised aligning with common AI concerns [45]. The cultural impact of AI in music was discussed, including the omission of cultural sensitivities inherent within specific genres. Discussion around AI advancing a wider cultural shift in the value of music in terms of economic concerns for both musicians and local economies and educational opportunities was raised. As Katz (2022) advises "in the relationship between music and technology, culture always matters" [46]. Therefore, awareness of the cultural entanglements between music, musicians and listeners [47] is key to ethical and human-centred integration of AI within music.

Authenticity and the inherently human activity of musicking was raised by all but one participant. The importance of human input and control was raised as a core requirement, aligning with existing literature [23], [33]. The personal aspect of composition was raised throughout, from the importance of nuance in styles of playing, to composition being led by musicians own lived experiences. This is tied to the core concerns around AI adoption of devaluing musicianship, loss of skills and lack of potential for experimentation. To address concerns, AI collaborative systems should ensure high user autonomy, with AI potentially acting as a "second person" to bounce ideas off [23], or as a muse,

generating ideas which users can incorporate within their creative vision.

5.3 RQ3: Perceptions Affected by AI in Music

Only 4 participants gave a clear yes/no response to whether the use of AI within music composition alters perceptions. Two participants (P2, P13) advised that it would give them a negative perception due to a lack of "substance" and authenticity (P13). The two 'no' respondents said that it would not change their perception as they viewed it as another creative tool. The additional 10 respondents had a more nuanced view, aligned with their previous responses and depended on particular use cases and the extent of AI integration. A HCAI approach may provide confidence that this technology can be embedded within creative practice, that AI can be shaped by musicians and their own creativity and sits within the broader technological scope as intrinsic to all musical activity [46]. Consumers may also have greater acceptance of AI within music if used as a creative tool similar to existing forms of music technology.

6 Limitations

Research limitations include potential participant bias due to the self-selecting aspects of the study. Participant numbers and geographical localisation also restrict some aspects of the result generalisability. However, hyper-localisation of the findings, particularly within a region consisting of a city with Music City status enabled context rich data which is of relevance to the AI music community.

7 Conclusions

This research provides insights into the views of musicians and music enthusiasts on the use of GenAI as a musical collaborator. Findings align with existing studies pertaining to AI use cases, particularly in assisting with production tasks and idea generation. The importance of AI literacy has been highlighted in this study, particularly how this may affect the uptake of AI. Aligning AI with existing music technology may assist musicians in accepting AI by providing a clearer mental model. Through stakeholder engagement, a HCAI approach should be embedded within the development of AI music tools to align with musicians existing practice and to allay the concerns raised by participants. Further work is needed to determine perceptions on a wider scale to map a clear approach to the development of HCAI music tools.

Acknowledgements

The authors would like to thank all research participants for their valuable insights.

References

[1] OpenAI, "Introducing ChatGPT." Accessed: Sep. 18, 2023. [Online]. Available: https://openai.com/blog/chatgpt

[2] D. Baidoo-Anu and L. Ansah, "Education in the Era of Generative Artificial Intelligence (AI): Understanding the Potential Benefits of ChatGPT in Promoting Teaching and Learning," *Journal of AI*, vol. 7, no. 1, pp. 52–62, 2023.

[3] Aiva Technologies, "AIVA: Your personal AI musical generation assistant." Accessed: Feb. 16, 2024. [Online]. Available: https://www.aiva.ai/

[4] Meta, "Generative AI for Audio." Accessed: Feb. 16, 2024. [Online]. Available: https://ai.meta.com/resources/models-and-libraries/audiocraft/

[5] OpenAI, "MuseNet." Accessed: Feb. 16, 2024. [Online]. Available: https://openai.com/research/musenet

[6] Suno Inc, "Suno." Accessed: Jun. 10, 2025. [Online]. Available: https://suno.com/home

[7] M. Riedl, "Human-Centered Artificial Intelligence and Machine Learning," *Human Behavior and Emerging Technologies*, vol. 1, no. 1, 2018, doi: https://doi.org/10.1002/hbe2.117.

[8] Oxford English Dictionary, "Musician." Accessed: Feb. 20, 2024. [Online]. Available: https://www.oed.com/search/dictionary/?scope=Entries&q=musician

[9] Oxford English Dictionary, "Enthusiast." Accessed: Feb. 20, 2024. [Online]. Available: https://www.oed.com/search/dictionary/?scope=Entries&q=enthusiast&tl=true

[10] C. Small, *Musicking*. Connecticut: Wesleyan University Press, 1998.

[11] K. Martineau, "What is generative AI?," IBM. Accessed: Feb. 23, 2024. [Online]. Available: https://research.ibm.com/blog/what-is-generative-AI

[12] I. J. Goodfellow *et al.*, "Generative Adversarial Networks," Jun. 10, 2014, *arXiv*: arXiv:1406.2661. doi: 10.48550/arXiv.1406.2661.

[13] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," Dec. 10, 2022, *arXiv*: arXiv:1312.6114. doi: 10.48550/arXiv.1312.6114.

[14] Z. Evans, J. Parker, C. Carr, Z. Zukowski, J. Taylor, and J. Pons, "Stable Audio Open," in *2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Hyderabad, India: IEEE, 2025.

[15] C.-Z. A. Huang *et al.*, "Music Transformer: Generating Music With Long-Term Structure," in *ICLR 2019*, New Orleans, 2019.

[16] Q. Huang *et al.*, "Noise2Music: Text-conditioned Music Generation with Diffusion Models," Mar. 06, 2023, *arXiv*: arXiv:2302.03917. doi: 10.48550/arXiv.2302.03917.

[17] Loudly, "AI Music For Your Creative Universe." Accessed: Feb. 16, 2024. [Online]. Available: https://www.loudly.com/

[18] Soundraw Inc, "Welcome to the AI Music Generation." Accessed: Feb. 28, 2024. [Online]. Available: https://soundraw.io/

[19] Google AI, "Magenta Studio." Accessed: Jan. 03, 2024. [Online]. Available: https://magenta.tensorflow.org/studio

[20] Orb Plugins, "Orb Producer Suite 3." Accessed: Jan. 03, 2024. [Online]. Available: https://www.orbplugins.com/orb-producer-suite/

[21] K.-B. Ooi *et al.*, "The Potential of Generative Artificial Intelligence Across Disciplines: Perspectives and Future Directions," *Journal of Computer Information Systems*, pp. 1–32, Oct. 2023, doi: 10.1080/08874417.2023.2261010.

[22] A. Holzapfel, B. L. Sturm, and M. Coeckelbergh, "Ethical Dimensions of Music Information Retrieval Technology," *Transactions of the International Society for Music Information Retrieval*, vol. 1, no. 1, pp. 44–55, Sep. 2018, doi: 10.5334/tismir.13.

[23] M. Newman, L. Morris, and J. H. Lee, "Human-AI Music Creation: Understanding the Perceptions and Experiences of Music Creators for Ethical and Productive Collaboration," in *Proceedings of the 24th ISMIR Conference*, Milan, Italy, 2023.

[24] P. Dahlstedt, "Musicking with Algorithms: Thoughts on Artificial Intelligence, Creativity, and Agency," in *Handbook of Artificial Intelligence for Music: Foundations, Advanced Approaches, and Developments for Creativity*, New York: Springer, 2021.

[25] Ditto Music, "60% of musicians are already using AI to make music," Music Industry Research. [Online]. Available: https://press.dittomusic.com/60-of-musicians-are-already-using-ai-to-make-music

[26] Pirate.com, "Over Half of Musicians Will Conceal AI Use." Accessed: Jan. 03, 2024. [Online]. Available: https://pirate.com/en/blog/over-half-of-musicians-will-conceal-ai-use/#:~:text=Notably%2C%2055%25%20of%20artists%20are,challenges%20surrounding%20AI%20in%20music.

[27] Institute for Human Centred AI, "The AI Index Report," Stanford University, 2023. Accessed: Jun. 14, 2023. [Online]. Available: https://aiindex.stanford.edu/report/

[28] C. Aragon, *Human-Centered Data Science*. London: MIT Press, 2022.

[29] B. Montgomery, "Music publishers sue Amazon-backed AI company over song lyrics," The Guardian. Accessed: Apr. 03, 2024. [Online]. Available: https://www.theguardian.com/technology/2023/oct/19/music-lawsuit-ai-song-lyrics-anthropic

[30] D. Tencer, "As Suno and Udio Admit Training AI With Unlicensed Music, Record Industry Says: 'There's Nothing Fair About Stealing An Artist's Life's Work,'" Music Business Worldwide. Accessed: Jun. 19, 2025. [Online]. Available: https://www.musicbusinessworldwide.com/

as-suno-and-udio-admit-training-ai-with-unlicensed-music-record-industry-says-theres-nothing-fair-about-stealing-an-artists-lifes-work/

[31] The Beatles, *Now and Then*. United Kingdom: Apple, 2023.

[32] H. McIntyre, "The Beatles Are Using AI To Release One Last Song - Why Aren't More Musicians Doing The Same?," Forbes. Accessed: May 03, 2024. [Online]. Available: https://www.forbes.com/sites/hughmcintyre/2023/10/09/the-beatles-are-using-ai-to-release-one-last-songwhy-arent-more-musicians-doing-the-same/?sh=5dca8e8929b2

[33] O. Ben-Tal, M. T. Harris, and B. L. T. Sturm, "How Music AI Is Useful: Engagements with Composers, Performers and Audiences," *Leonardo*, vol. 54, no. 5, pp. 510–516, Oct. 2021, doi: 10.1162/leon_a_01959.

[34] R. Louie, A. Coenen, C. Z. Huang, M. Terry, and C. J. Cai, "Novice-AI Music Co-Creation via AI-Steering Tools for Deep Generative Models," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, Honolulu HI USA: ACM, Apr. 2020, pp. 1–13. doi: 10.1145/3313831.3376739.

[35] The Glasshouse, "The Glasshouse: International Centre for Music." Accessed: Mar. 28, 2024. [Online]. Available: https://theglasshouseicm.org/?gad_source=1&gclid=Cj0KCQjwqpSwBhClARIsADlZ_TlK3ilOJzoxr1NLV-7ACeIfIGdxvnujZg0yvpe9JMpnh48mTcC0yNoaAjz8EALw_wcB

[36] The Fire Station, "The Fire Station: Sunderland's premier live music and performance venue." Accessed: Mar. 28, 2024. [Online]. Available: https://www.thefirestation.org.uk/

[37] P. Bilalova and A. Watson, "City to join international music network," BBC News. Accessed: Jun. 20, 2025. [Online]. Available: https://www.bbc.co.uk/news/articles/c4g370wjg3go

[38] K. Roulston and M. Choi, "Qualitative Interviews," in *The Sage Handbook of Qualitative Data Collection*, New York: SAGE Publications Ltd, 2018.

[39] V. Braun and V. Clarke, "Reflecting on reflexive thematic analysis," *Qualitative Research in Sport, Exercise and Health*, vol. 11, no. 4, pp. 589–597, Aug. 2019, doi: 10.1080/2159676X.2019.1628806.

[40] *Pro Tools*. (2025). Avid Technology Inc. [Online]. Available: https://www.avid.com/pro-tools

[41] *GarageBand*. (2025). Apple Inc. [Online]. Available: https://apps.apple.com/us/app/garageband/id408709785

[42] *Logic Pro*. (2025). Apple Inc. [Online]. Available: https://www.apple.com/uk/logic-pro/

[43] D. Buschek, L. Mecke, F. Lehmann, and H. Dang, "Nine Potential Pitfalls when Designing Human-AI Co-Creative Systems," *Joint Proceedings of the ACM IUI 2021 Workshop.*, 2021, doi: arXiv preprint arXiv:2104.00358.

[44] J. B. Smith and J. Freeman, "Effects of Visual Explanation on Perceived Creative Autonomy in an AI-Based Generative Music System," in *28th International Conference on Intelligent User Interfaces*, Sydney NSW Australia: ACM, Mar. 2023, pp. 25–28. doi: 10.1145/3581754.3584123.

[45] C. Huang, Z. Zhang, B. Mao, and X. Yao, "An Overview of Artificial Intelligence Ethics," *IEEE Trans. Artif. Intell.*, vol. 4, no. 4, pp. 799–819, Aug. 2023, doi: 10.1109/TAI.2022.3194503.

[46] M. Katz, *Music and Technology: A Very Short Introduction*. Oxford: Oxford University Press, 2022.

[47] P. Martin, *Sounds and Society: Themes in the Sociology of Music*. Manchester: Manchester University Press, 1995.

Audio Engineering Society

Conference Paper 10

Presented at the AES International Conference on Machine Learning and Artificial Intelligence for Audio 2025 September 8–10, London, UK

This paper was peer-reviewed as a complete manuscript for presentation at this conference. This paper is available in the AES E-Library (http://www.aes.org/e-lib), all rights reserved. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Journal of the Audio Engineering Society.

Multiple Loudspeaker Localization with Simultaneous Deconvolution

Sunil Bharitkar[1] and Adrian Celestinos[1]

[1] *DMS Audio Lab, Samsung Research America*

Correspondence should be addressed to Sunil Bharitkar (s.bharitkar@samsung.com)

ABSTRACT

Loudspeaker localization in a reverberant room is essential for ensuring high-quality immersive rendering with spatially distributed loudspeakers. Traditional approaches rely on beamforming or deep-learning-based feature extraction for a single source localization at a time. This paper applies simultaneous deconvolution (SD) towards multi-source localization where loudspeaker distances are first estimated from impulse responses deconvolved *simultaneously*, and the distances are then used towards estimating loudspeaker signal Direction-of-Arrival (DOA). We compare geometric angle estimators, supervised and unsupervised machine learning (ML) models, and traditional signal processing approaches involving cross-correlation and pseudo-spectrum. The Trapezoid estimator and supervised ML models are the top two techniques for use with SD for loudspeaker localization, factoring both synthetic and listening room test sets.

1 Introduction

Calibrating loudspeakers in a room is essential to deliver high-quality, immersive spatial audio for rendering next-generation immersive audio formats [1]. The calibration involves computing and compensating for the time difference of arrival (TDOA) between loudspeaker signals, sound pressure level matching between loudspeakers, frequency equalization, and *estimating the location of the loudspeakers* for use with panning. To perform source localization, traditional signal processing methods such as GCC-PHAT and MUSIC form the basis for the signal processing models [2]-[5]. Newer deep-learning models have been used for source localization and tracking but have high computational and memory footprints, preventing use in real-time processing in a DSP silicon. Several of these have been designed with limited sources active at any given time (usually one source at a time). The input features are spectrogram, features extracted from GCC, inter-channel features, binaural features, or even raw signals captured at microphones. The output is usually azimuth (obtained from classification or regression), with less emphasis on elevation. Distance estimation has not been properly investigated because it is a challenging problem. A survey paper by Grumiaux *et al.* (cf. Table 2 in [6]) gives an excellent coverage of deep-learning-based source localization.

An approach for estimating distances is through impulse responses (IRs) captured between loudspeakers and microphones in a room. Towards this, the state-of-the-art (SOTA) approach involves deconvolving the impulse response $h_{j,i}(n)$ at a microphone position j,

by exciting a loudspeaker i with an excitation stimulus ([7]), and then proceeding sequentially to the $i+1$-th loudspeaker. The disadvantages of sequential measurements become apparent when the number of loudspeakers and positions increases, as the time required to deconvolve room responses from all loudspeakers increases, which may be unacceptable in consumer (home-theater) environments. Bharitkar [8], [9] presents a technique for simultaneous deconvolution (SD), which involves exciting all loudspeakers at the same time and deconvolving responses using a single time-domain measurement. We incorporate SD into this work for loudspeaker localization. From the IRs deconvolved from SD, by exciting all sources simultaneously, we compute distances between speakers and microphones and then determine relative angles. We present a comparison of various machine learning techniques (supervised and unsupervised) with geometric estimators and classical signal processing solutions to estimate the angles of the speakers. The machine learning models include the Generalized Regression Neural Network (GRNN) [12], Kolmogorov-Arnold Neural Network (KAN) [13], and Multidimensional Scaling (MDS) [14]. Simultaneous Deconvolution, which provides a faster measurement than classical IR deconvolution [9], and the best-performing techniques involving trapezoid and ML-based estimators have low memory and compute footprint and can be implemented in consumer devices in a DSP.

The paper's organization is as follows: Section 2 summarizes SD for IR deconvolution and distance estimation from IRs. Section 3 outlines the various techniques used to calculate the speakers' angles using ML, geometric, and signal-processing localization models. Section 4 compares results from the various localization models, whereas sections 5 and 6 present discussions and conclusions for the paper, respectively.

2 Simultaneous Deconvolution for IR & Distance Estimation

Traditional deconvolution techniques in room acoustics involve single-channel-at-a-time IR estimation, which can result in a long measurement time for several loudspeakers. Newer approaches to address this limitation include the work presented by Majdak *et al.* [10] based on interleaving or partial overlap sweep stimuli. This section summarizes SD [8], [9] used in this work for the deconvolution of IRs with all speakers being active simultaneously during measurement.

2.1 Excitation Stimulus: Logarithmic Sine-sweep

In the case of an exponential sweep [11], assuming that ω_1 and ω_2 are the start and end frequencies (in rad/s), with a total duration of T_{log} seconds, the logarithmic sweep signal $x(t)$ is

$$x(t) = \sin(\frac{\omega_1 T_{\log}}{\log \frac{\omega_2}{\omega_1}}(e^{\frac{t}{T_{\log}}\log(\frac{\omega_2}{\omega_1})} - 1)) \tag{1}$$

whereas the discrete-time equivalent is $\mathbf{x}_1(n) = (x(n), x(n-1), x(n-2), \ldots, x(n-(P_{log}-1)))^T$, T represents the vector transpose and $P_{log} = T_{log} f_s$ in samples and f_s is sampling frequency.

2.2 Simultaneous Deconvolution

The measurement is a linear convolution sum between the ground-truth loudspeaker-room impulse response (IR) $\mathbf{h}_i$ and the stimuli $\mathbf{x}_i(n)$,

$$\mathbf{y}(n) = \sum_{i=1}^{N} \mathbf{x}_i(n) \circledast \mathbf{h}_i \tag{2}$$

The N-channel log-sweep stimulus for simultaneous deconvolution is constructed by circularly shifting the log-sweep for each channel/loudspeaker stimulus relative to the stimulus $\mathbf{x}_1(n)$ applied to the first channel/loudspeaker. The duration P_{log} and circular shift value M are determined by minimizing the mean-square error between reconstructed room response and original (ground-truth) responses using a dataset of room responses [8],[9].

$$\begin{aligned} \mathbf{x}_1(n) &= [x(n), x(n-1), \ldots, x(n-P_{log}+1)]^T; \\ \mathbf{x}_i(n) &= [x(< n-(i-1)M >_{P_{log}}), \ldots, \\ & \quad x(< n-(i-1)M-1 >_{P_{log}}), \ldots, \\ & \quad x(< n-(i-1)M-P_{log}+1) >_{P_{log}}]^T; \end{aligned} \tag{3}$$

with $i = (2, \ldots, N)$, $< m >_{P_{log}} = m$ modulo P_{log}, and $\mathbf{h}_i = [h_i(1), h_i(2), \ldots, h_i(K)]^T$ is a K-length impulse response. The loudspeaker-room responses are deconvolved using,

$$\hat{\mathbf{h}}_j = \mathscr{F}^{-1}\{\frac{S_{\mathbf{x}_j,\mathbf{y}}(e^{j\omega})}{S_{\mathbf{x}_j,\mathbf{x}_j}(e^{j\omega})}\} \tag{4}$$

where, $S_{\mathbf{x}_j,\mathbf{y}}(e^{j\omega})$ and $S_{\mathbf{x}_j,\mathbf{x}_j}(e^{j\omega})$ are cross- and auto-spectra, respectively. Multi-source SD has a significant measurement time advantage over classical single-channel-at-a-time measurement as it allows the estimation of IRs with all sources active at the same time during measurement.

For example, if (i) T_m is the total measurement time that includes stimulus duration for playback and the microphone recording time, (ii) T_t is the time it takes to switch between loudspeakers for single-channel measurements, and (iii) L_{avg} is the number of measurement averages for a high signal-to-noise ratio (SNR), then the conventional single-channel-at-a-time takes $\tau_{conv} = NL_{avg}T_m + (N-1)T_t$; whereas for SD it is $\tau_{SD} = L_{avg}T_m$. With $L_{avg} = 3$, $T_m = 5$ s, $T_t = 3$ (sec.), $N = 12$ for an ITU 7.1.4 [1], $\tau_{conv} = 180$ s whereas $\tau_{SD} = 15$ s.

2.3 Distance Estimation from IR

An onset detection technique for distance estimation comprises first normalizing the IR to unit amplitude and then applying a band-pass filter to isolate the higher-frequency components in the IR. A thresholding scheme detects the time sample k where the absolute value of the IR first exceeds 0.1, which leads to the estimated distance $\hat{d} = c * k/f_s$ (where the speed of sound is $c = 343$ and sampling frequency $f_s = 48$ kHz). Figure 1 shows zoomed-in examples of IRs deconvolved at four different ground truth distances d_i, between loudspeaker and microphone, and the onset values k_i. The errors $|e_i| = |d_i - \hat{d}_i|$ at each of the four positions are $(2.16, 2.1, 0.38, 10)$ cm, demonstrating that the onset detection technique can approximate distances with small error.

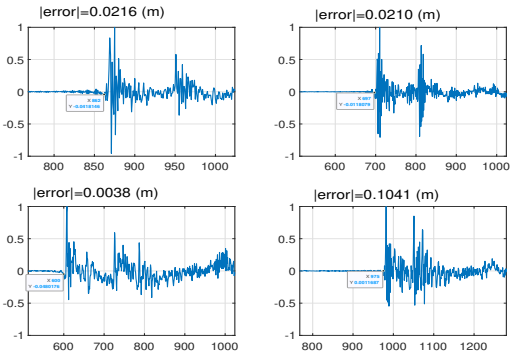


Fig. 1: Examples showing error $|e|$ between ground-truth distance (measured with a laser meter) and estimated from SD-IR onset at four different distances.

Fig. 2: Example of two discrete loudspeakers mounted relative to the TV used for testing data (each discrete loudspeaker has a microphone and the TV has dual microphones).

3 Loudspeaker Localization Models

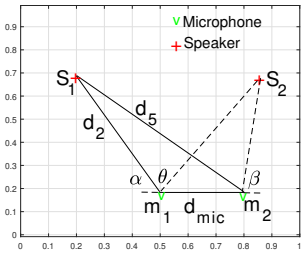
Pairwise distances between devices can be used to localize them, as shown in this paper. To derive practical localization models, we constrain the problem to identifying the elevation angles of two speakers, one on each side of a television (TV) as shown as an example in Fig. 2. This setup is typical in a consumer home theater as it involves two frontal discrete loudspeakers (e.g., left and right loudspeakers for stereo signals or the front loudspeakers of an ITU setup such as 5.1 or 7.1.4 system) mounted on each side but at arbitrary elevations relative to a television (TV) microphone array. As described below, various estimators or localization models are used in this paper to estimate the elevation angles of the loudspeaker by first using SD to determine distances between the loudspeakers.

3.1 Geometric Estimators

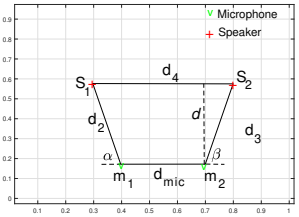
(i) Triangle: With reference to Fig. 3(a), the angle α can be computed from the distances (d_2, d_3) and known inter-microphone spacing d_{mic}, as

$$\theta = \cos^{-1}\left(-\frac{d_5^2 - (d_2^2 + d_{mic}^2)}{2d_2 d_{mic}}\right) \tag{5}$$

with $\hat{\alpha} = \pi - \theta$. An important constraint for $|\cos(\theta)| \leq 1$ is $d_5 \leq (d_2 + d_{mic})$; for the angles to be real and not imaginary. The results section will show that this



(a)



(b)

Fig. 3: Geometric representations for determining microphone-source angles (a) Triangle, (b) Quadrilateral

estimator results in imaginary solutions as distances estimated from certain IR measurements do not satisfy this constraint.

(ii) Trapezoid (Quadrilateral): With reference to Fig. 3(b), the angles $(\hat{\alpha},\hat{\beta})$ can be readily determined using

$$\begin{aligned} p &= d_2+d_3+d_4-d_{mic}; q = d_2+d_3+d_{mic}-d_4 \\ r &= d_2+d_4-d_3-d_{mic}; s = d_3+d_4-d_2-d_{mic} \\ d &= \sqrt{p\times q\times r\times s}/(4(d_4-d_{mic})^2) \qquad (6) \\ \hat{\alpha} &= \frac{\pi}{2}-\sin^{-1}(\sqrt{(d_2^2-d^2)/d_2^2}); \\ \hat{\beta} &= \frac{\pi}{2}-\sin^{-1}(\sqrt{(d_3^2-d^2)/d_3^2}) \end{aligned}$$

3.2 Supervised Models

(i) GRNN[12]: The Generalized Regression neural network (GRNN) is popular due to its "one-shot" training and its strong generalization ability. The GRNN is a form of radial basis function neural network whose approximation is based on linear combinations of nonlinear basis functions. Let $\underline{U}_i = [u_i(1),u_i(2),\ldots,u_i(m)]^T$ be an input vector and $\underline{V}_i = [v_i(1),v_i(2),\ldots,v_i(n)]^T$ be the output vector corresponding to the $i-$th training data $(i=1,\ldots,N)$. For an arbitrary input vector $\underline{U} = [u(1),u(2),\ldots,u(m)]^T$, the estimated output $\underline{\hat{V}}$ can be expressed as,

$$\underline{\hat{V}} = \frac{\sum_{i=1}^{N}\frac{\underline{V}_i}{\sigma^m}\exp(-\|\underline{U}-\underline{U}_i\|^2/2\sigma^2)}{\sum_{i=1}^{N}\frac{1}{\sigma^m}\exp(-\|\underline{U}-\underline{U}_i\|^2/2\sigma^2)} \qquad (7)$$

The input to the GRNN is distances (d_2,d_3,d_4,d_{mic}). The output is the two estimated angles $(\hat{\alpha},\hat{\beta})$ corresponding to the notation in Fig. 3. The training set for GRNN is derived using synthetic positions/distances (as input) and angles (computed using positions relative to TV microphones), as explained in the next section.

(ii) KAN [13]: The Kolmogorov-Arnold Networks (KANs) are an alternative to feedforward multilayer networks (MLP) leveraging the Kolmogorov-Arnold Theorem. The KAN is a two-layer neural network with activation functions on the edges rather than the nodes. KAN's key feature is the replacement of traditional weights with learnable activation functions. The inputs to the KAN are distances (d_2,d_3,d_4,d_{mic}), and the output is the two angles $(\hat{\alpha},\hat{\beta})$ corresponding to the notation in Fig. 3.

3.3 Unsupervised Models

MDS [14]: Multidimensional scaling (MDS) is a technique that maps distance data between pairs of objects into distances between points in a multidimensional space (usually two or three dimensions). The final visual representation includes coordinates of objects that satisfy the distance relationships between the objects. Towards this, the MDS uses as input $(d_2,d_3,d_4,d_{mic},d_{S_1,m_2},d_{S_2,m_1})$. The elevation angles can be determined once the coordinates are determined using MDS relative to the TV mics.

3.4 Classical Signal Processing

Neither of the classical techniques can cope with multiple active loudspeakers at the same time, but the methods can still be used with each TV microphone-captured signal $y_i(n) = x_j(n) \circledast h_{j,i}(n); (i=1,2)$.

(i) GCC-Nonlin [15]: The generalized cross-correlation with phase transform (GCC-PHAT) method [18] is a popular angular spectrum-based method for DOA estimation. Based on the assumption that the direct sound of one source predominates in each time-frequency bin,

the TDOA $\hat{\tau}$ is estimated from the the phase difference between a pair of microphone-captured signals. With $R_{x1,x2}(t,f)$ being the time-frequency based covariance matrix, the GCC-PHAT solution is,

$$\phi_{GCC}(t,f,\tau) = \Re\{\frac{R_{y_1,y_2}(t,f)e^{-2\pi f\tau}}{|R_{y_1,y_2}(t,f)|}\} \quad (8)$$

with $\hat{\tau} = \text{argmax}_{\tau}(\phi_{GCC}(t,f,\tau))$ and $\Re\{.\}$ denoting the real part. Loesch *et al.* (Eq. (4) in [16]) demonstrate improved results by emphasizing large values of $\phi_{GCC}(.)$ by first transforming Eq. (8) using a nonlinear function such as tanh(.). The angle of the loudspeaker using far-field assumptions is $\hat{\theta} = \cos^{-1}(d_{mic}/c\hat{\tau})$ with $d_{mic} << d_i; \forall i$ (as in this paper).

(ii) MUSIC [17]: Multiple signal classification (MUSIC) and variants is a subspace-based method that incorporates eigenvalue decomposition of a sample covariance matrix to project signals into a subspace while filtering out unwanted noise. The DOA is then computed by analyzing the MUSIC pseudo-spectra over a grid for peaks associated with candidate angles described by the projection of a steering vector $a(\theta)$ on the noise subspace E_n comprised of the eigenvectors of $M-D$ lowest eigenvalues of the covariance matrix for D sources

$$P_{MUSIC}(\theta) = 1/(a^H(\theta)E_nE_n^Ha(\theta))$$

4 Results

We create synthetic data using rectangular coordinates to derive distances between speakers and microphones, so as to train and test the various models. The results using this synthetic data represent an upper bound on performance since (i) onset detection from IRs measured in the real world is not exact given the differences between physical measurements and acoustical measurements (e.g., the ground-truth distances in real-world measurements are obtained with a laser from the physical center of the loudspeaker to microphones whereas the distance obtained from onset detection is from the loudspeaker transducer to the microphone), (ii) with the synthetic data the speakers are depicted on the same wall as the TV for elevation estimation, whereas during validation in the listening room, the loudspeakers were offset from the wall. While the speaker acoustical centers could be used to measure ground-truth physical distances, we used the physical center of the speaker for simplicity due to differences in acoustical centers for different loudspeakers (viz., some speakers could be front-firing and others may be side-firing).

4.1 Synthetic Data

Figure 4 depicts the random distribution of $N_1 = 1000$ data for the locations of the front loudspeakers on each side of the TV microphones. The (+) indicates the physical center of the speaker, whereas the microphones are depicted with (v). The ground-truth distances and synthetic elevation angles between speakers and microphones, $(\alpha,\beta)^T$, corresponding to angles shown in Fig. 3 are computed with trigonometry. The thin vertical space between the TV edge and the depicted center of the loudspeaker position ensures that the loudspeaker does not physically overlap with the TV. The training input matrix is $\Re^{1000\times 4}$ for the GRNN (with $\sigma = 0.08$) and KAN with the four distances described in Sec. III.B, whereas for the MDS model, all six distances are used as input as described in Sect. III.C. No effort was made to optimize the neural network complexity or parameters (e.g., σ in the GRNN, training set size, number of layers for KAN). For the synthetic case, the GRNN had 7000 parameters (weights)[2]. In contrast, the shallow single hidden layer KAN had 960 parameters. In the case of GCC-Nonlin and MUSIC, the signals at each of the TV microphones, y_i, are generated by convolving a male speech signal $x_j(n)$ located at source S_j with $h_{j,i}(n) = \delta(n - \lfloor d_i f_s/c \rfloor)$[3]. Figure 5 is the box-

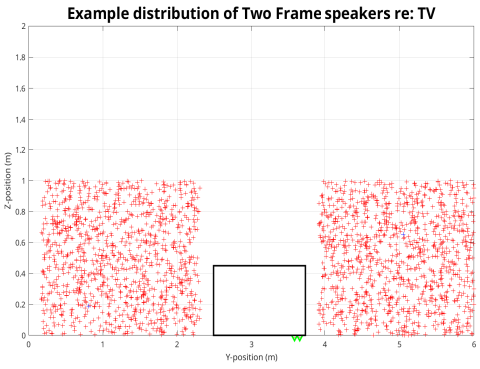


Fig. 4: Synthetic data showing speaker distribution (+) relative to dual-mics in TV (v).

[2]With 4-inputs and 2-outputs, and with a 1000 training set size, the number of parameters is $4\times 1000 + 2\times 1000 + 1000 = 7000$ (includes 1000 bias neurons)

[3]Both implementations and relevant speech signals, $x_j(n)$, were obtained at `https://gitlab.inria.fr/bass-db/bss_locate`

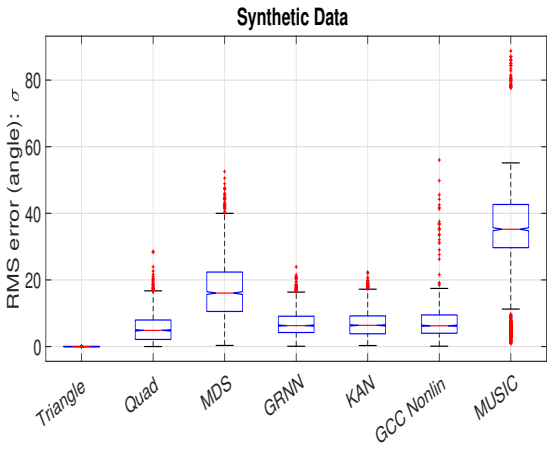


Fig. 5: Upper-bound of performance (RMS error) of localization models using test synthetic data.

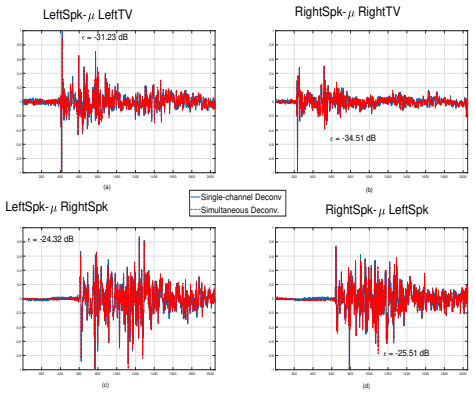


Fig. 6: Comparison between SD and single-channel deconvolved IRs for four loudspeakers.

plot of the RMS error $\sigma = \sqrt{\|(\alpha,\beta)^T - (\hat{\alpha},\hat{\beta})^T\|_2^2/2}$ on a synthetic test set with $N_2 = 1000$ showing optimal performance with the triangle estimator, as expected. In contrast, the trapezoid estimator, GRNN, KAN, and GCC-Nonlin give slightly higher errors with a median $\sigma \approx 4^\circ$.

4.2 Validation with Listening Room

Figure 2 shows an example setup for the listening room with a TV and two speakers. In one setup, each of the two speakers was mounted on the sidewalls orthogonal to the wall adjacent to the TV. Figure 6 shows the match between SOTA (one-channel-at-a-time) and SD (in red) for the two internal TV speakers, as well as left and right discrete speakers corresponding to Fig. **??** measured at the left and right TV microphones. The error in the match for the left discrete speaker to left TV mic is -31 dB, right discrete speaker to right TV mic is -34.5 dB, left discrete speaker to mic in right speaker is -24 dB and right discrete speaker to mic in left discrete speaker is -26 dB. For GCC and MUSIC, the male speech signal is convolved with IRs associated with the two discrete loudspeakers measured at the TV mics $(y_i(n) = x_j(n) \circledast h_{j,i}(n); (i = 1,2))$. The geometric estimators and MDS are not trainable models, and hence, the pairwise distances from onset detection are used directly to estimate the elevation angles. The GRNN and KAN models trained on synthetic data were used to predict angles using distances from onset detection. A latency compensation solution ensured the microphones on each of the discrete speakers (which were wireless) and the TV were operating on the same clock before measurements for use with SD. The ground truth (α,β) was obtained from distances confirmed with a laser meter, and the error σ in Table 1 is expressed in degrees. Negative angles for the left speaker to the left microphone are clockwise referenced from the horizontal. In contrast, anti-clockwise angles from the horizontal for the right speaker to the right microphone are expressed as positive. In all but one case, for the triangle estimator, at least one angle in the pair (α,β) was estimated on the imaginary axis. In this case, the corresponding imaginary angle was set to zero. The errors determined from this invalid estimation are therefore italicized in Table I.

5 Discussion

As seen in the listening room results in Table I, the triangle estimator gives high errors compared to results in Fig. 5 due to at least one imaginary angle found in five of the six measurements. The imaginary angles arise since the triangle estimator is sensitive to distances obtained from onset detection in IRs, as the condition $d_5 \leq (d_2 + d_{mic})$ is not satisfied. In contrast, the MDS approach performed well in the listening room, but its performance was not the best on the synthetic set. For MDS, the listening room distance data

Table 1: RMS error σ in degrees from ground truth (α, β)

[a: Triangle, b. Trapezoid, c. GRNN, d. MDS, e. KAN, f. GCC, g. MUSIC]

(α, β)	a	b	c	d	e	f	g
(-4,25)	48	6	5	6	6	18	11
(-13,9)	*11*	2.5	3	3	6	58	45
(-27,21)	*19*	6	6	4	11	51	63
(-12,41)	*30*	5	9	2	5	74	38
(-27,41)	*29*	4	10	6	8	39	33
(-23,14)	*10*	5	12	5	20	19	46
Avg.	24.5	5	7.5	4.5	9.3	43	39.3

is likely distributed similarly to the synthetic distance data at the MDS boxplot's lower whisker in Fig. 5. The GRNN, KAN, and Trapezoid (quadrilateral) estimators consistently performed well on both test sets. The GCC-Nonlin approach performed well in the synthetic set but had higher error in the listening room due to room reflections and reverberation.

6 Conclusions

This paper presents a multi-source loudspeaker localization approach using SD and employing various location estimators. The supervised models are assessed on synthetic test set (employing distances as input and angles as output) and listening room data. The distances in the listening room measurements are determined using onset detection using IRs deconvolved from a fast measurement technique (SD) with all sources active. The supervised models and the trapezoid estimator outperform the triangle estimator and the classical signal processing methods using the RMS error (σ) metric in both synthetic and real-world data. Follow-up research will expand the technique to include localizing more loudspeakers in three dimensions and optimizing the complexity of models (e.g., neural network model size using Bayesian optimization).

References

[1] ITU-R BS. 2051-3, *Advanced sound system for programme production* , Int. Telecom. Union (ITU), 2022.

[2] M. Brandstein and D. Ward (Eds.), *Microphone Arrays: Signal Processing Techniques and Applications*, Springer: Berlin, 2001.

[3] J. Benesty, J. Checn, and Y. Huang, *Microphone Array Signal Processing*, Springer: New York, 2008.

[4] M. Cobos, F. Antonacci, A. Alexandridis, A. Mouchtaris, and B. Lee, "A survey of sound source localization methods in wireless acoustic sensor networks," *Wireless Comm. Mobile Comput.*, 2017, pp. 1–24.

[5] S. Argentieri, P. Danes, and P. Souéres, "A survey on sound source localization in robotics: From binaural to array processing methods," *Comput. Speech Lang.*, 34(1), 2015, pp. 87–112.

[6] P-A. Grumiaux, S. Kitić, L. Girin, and A. Gu/'erin, "A survey of sound localization with deep learning methods," *J. Acoust. Soc. Amer.*, 152(1), July 2022, pp. 107–151.

[7] S. Cecchi, A. Carini and S. Spors, "Room Response Equalization—A Review," *Applied Sciences*, 8(16), 2018.

[8] S. Bharitkar, "Bayesian Optimization for Simultaneous Deconvolution of Room Impulse Responses," *Proc. 36th IEEE Wkshp. Sig. Proc. Syst.*, Rennes (France), Nov. 2022.

[9] S. Bharitkar, "A Comparison Between Sweep Signals for Simultaneous Deconvolution," *Proc. 156th AES Conv.*, Madrid (Spain), June 2024.

[10] P. Majdak, P. Balazs, and B. Laback, "Multiple Exponential Sweep Method for Fast Measurement of Head-related Transfer Functions," *J. Audio Eng. Soc.*, 55(7/8), July/Aug. 2007, pp. 623–637.

[11] A. Farina, "Simultaneous Measurement of Impulse Response and Distortion with a Swept-Sine Technique," *Proc. 108th Audio Eng. Soc. (AES) Conv.*, Paris (France), Feb. 2000.

[12] D. Specht, "A general regression neural network," *IEEE Trans. Neur. Netw.*, 2(6), pp. 568-576, 1991.

[13] Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljačić, T. Hou, and M. Tegmark, "KAN: Kolmogorov-Arnold Networks,"

arXiv:2404.19756 (accepted *Int. Conf. Learn. Rep.* ICLR 2025).

[14] T. Cox and M. Cox, *Multidimensional Scaling* (2nd Edition), CRC Press, 2001.

[15] C. Blandin, A. Ozerov, and E. Vincent, "Multi-source TDOA estimation in reverberant audio using angular spectra and clustering," *Sig. Proc.*, 92, pp. 1950–1960, 2012.

[16] B. Loesch and B. Yang, "Adaptive segmentation and separation of determined convolutive mixtures under dynamic conditions," in *Latent Variable Analysis and Signal Separation*, Springer (Berlin), 210, pp. 41–48.

[17] R. Schmidt, "Multiple emitter location and signal parameter estimation," *IEEE Trans. Antennas & Prop.*, 34 (3), (1986), pp. 276– 280.

[18] C. Knapp and G. Carter, "The generalized correlation method for estimation of time delay," *IEEE Trans. Acoust., Speech & Sig. Proc.*, ASSP-24(4), Aug. 1976, pp. 320–327.

[19] E. Claudio, R. Parisi, and G. Orlandi, "Multi-source localization in reverberant environments by ROOT-MUSIC and clustering," *Proc. ICASSP-00*, Istanbul (Turkey), June 2000.

Audio Engineering Society

Conference Paper 11

Presented at the AES International Conference on Artificial Intelligence and Machine Learning in Audio
2025 September 8–10, London, UK

This paper was peer-reviewed as a complete manuscript for presentation at this conference. This paper is available in the AES E-Library (http://www.aes.org/e-lib), all rights reserved. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Journal of the Audio Engineering Society.

Complex-valued physics-informed neural networks for sound field estimation

Vlad S. Paul, Nara Hahn, and Philip A. Nelson

Institute of Sound and Vibration Research, University of Southampton, United Kingdom

Correspondence should be addressed to Vlad S. Paul (v.s.paul@soton.ac.uk)

ABSTRACT

Sound field reconstruction from sparse measurements is a challenging problem in acoustics, with applications in room acoustics, spatial audio capture, and active noise control. In this work, a complex-valued physics-informed neural network (PINN) is proposed for reconstructing sound fields from data collected by an open-sphere microphone array at control points distributed on a 2D grid. Unlike conventional PINNs that use only spatial coordinates as input, our model additionally incorporates measured pressure values, enabling it to learn the mapping between observed fields and their spatial distributions. In order to investigate the impact of the physical constraints on the model's performance, the network is trained on sound fields generated from superposed single-frequency plane waves and its performance is compared to that of a complex-valued multilayer perceptron (MLP). Results show that the complex-valued PINN yields smoother training curves and extended regions of low reconstruction error, highlighting its enhanced spatial consistency. Future work will focus on investigating the model's generalization to unseen sound fields and varying control point grids, as well as optimizing the placement and number of training points.

1 Introduction

Sound field estimation involves reconstructing the spatial distribution of acoustical variables from a finite number of discrete measurements. This fundamental approach is used in applications such as room acoustics [1, 2], spatial sound capture [3] and active noise control [4]. Conventional methods often rely on physical models, such as finite element methods (FEM), which discretise the domain to numerically solve the wave equation. While effective, these methods become computationally expensive, especially at higher frequencies, due to the need for dense spatial sampling. Alternatively, analytical approaches like spherical harmonic (SH) expansions provide efficient representations by leveraging known wave solutions but might struggle in complex environments with irregular boundaries or unknown scattering effects.

Virtual sensing extends this concept by estimating the sound field at unmeasured locations, known as "virtual sensors". This approach is particularly useful when direct measurements are impractical or invasive. By leveraging data from available sensors and incorporating physical principles, virtual sensing aims to provide accurate estimations of the sound field in regions of interest.

In recent years, machine learning techniques have been used for sound field estimation applications to enhance accuracy and efficiency [5, 6]. In these data-driven approaches, the network's parameters are adjusted using a training dataset to estimate the sound fields, without any prior knowledge of physical properties. Physics-informed neural networks (PINNs) are a notable development in this area, since these models incorporate the governing physical laws, such as those leading to the wave equation, directly into the loss function of the neural network. This integration ensures that the network's predictions adhere to known physical behaviors, even when trained on limited data [7]. PINNs have recently gained attention in the field of sound field reconstruction. In [8], the authors proposed the use of PINNs to address the issue of spherical Bessel function nulls in open sphere microphone arrays (OSMAs). The model is trained from measurements on an OSMA to estimate the sound field on a sphere of a different radius. This approach leverages the fact that nulls in spherical Bessel functions vary with radius. Using the PINN enables the retrieval of sound field coefficients that would otherwise be difficult to obtain directly from the OSMA measurements. In [9], a PINN-based approach is proposed for recovering acoustic fields, particularly focusing on speech signals. Comparisons with various frequency-domain and time-domain reconstruction benchmark techniques revealed that the proposed PINN method achieved superior accuracy across various measurement setups.

Traditional neural networks are typically real-valued and may struggle with the phase and magnitude relationships present in complex data. Combining PINNs with complex-valued neural networks (CVNNs) offers a promising approach for sound field estimation applications and has been used very recently in [10] for acoustic holography. CVNNs process complex numbers directly, allowing them to inherently represent the phase relationships crucial for accurate sound field representation. The benefits of CVNNs have been investigated in various works such as [11, 12, 13] and it has been shown that in some cases, using complex numbers during training increases the performance of the model compared to using just real data.

This work will investigate the use of complex-valued PINNs for a sound field reconstruction task. Individual sound fields, each generated from superposed single-frequency plane waves, are measured using a microphone array placed on the surface of an open sphere. The task is to estimate the pressure at some control points on the horizontal plane inside the open sphere. Unlike conventional PINNs, which typically take only spatial (and possibly temporal) coordinates as inputs, the proposed model incorporates the measured sound fields into the training data as an additional condition. By including both microphone pressure measurements and spatial coordinates as inputs, the network can infer the relationship between multiple sound field measurements and their corresponding spatial distributions. This design is inspired by the work of [14], who introduced Conditional PINNs. These models incorporate "tag vectors" as conditional inputs to represent a family of physical scenarios.

Such an approach has promising implications for a range of audio applications. For instance, in Active Noise Control (ANC) systems, accurately reconstructing various sound fields at virtual sensor locations is essential for generating the anti-noise signals needed to cancel unwanted sounds. Moreover, this method can enhance immersive audio capturing and reproduction, where high-fidelity spatial reconstruction enables 3D sound experiences for virtual reality (VR) and augmented reality (AR) environments.

2 Problem Formulation

2.1 Preliminaries

Assume a three-dimensional space $\mathbb{R}^3$, where a spatial coordinate vector that represents a point in space is defined as $\mathbf{r} = (x, y, z)$. The acoustic pressure field $p(\mathbf{r}, k)$ is governed by the Helmholtz equation, which describes the propagation of time-harmonic waves. For a given frequency f and corresponding wavenumber $k = 2\pi f/c$, where c is the speed of sound, the Helmholtz equation is expressed as

$$\nabla^2 p(\mathbf{r}, k) + k^2 p(\mathbf{r}, k) = 0, \quad \mathbf{r} \in \Omega \tag{1}$$

where ∇^2 denotes the Laplacian with respect to $\mathbf{r}$ and where Ω is the spatial domain of interest. Consider a microphone located at position $\mathbf{r}_\mathrm{m}$ and a plane wave propagating in the direction specified by the unit vector $\hat{\mathbf{r}}_\mathrm{s}$. The pressure at $\mathbf{r}_\mathrm{m}$ for a single-frequency plane wave is given by

$$p(\mathbf{r}_\mathrm{m}, k) = A e^{-j(k\hat{\mathbf{r}}_\mathrm{s} \cdot \mathbf{r}_\mathrm{m} + \alpha)}, \tag{2}$$

where A is the amplitude of the plane wave. Here, $\hat{\mathbf{r}}_\mathrm{s} \cdot \mathbf{r}_\mathrm{m}$ denotes the dot product between the propagation direction and the microphone position, effectively projecting the position onto the direction of the incident wave. Note that $\hat{\mathbf{r}}_\mathrm{s}$ is normalised to unit length, ensuring it represents the direction of wave propagation. The phase term α can be used to add a random offset to simulate slight differences in the arrival times of the plane wave at different microphones in the array.

2.2 Data model

Consider a simulated microphone array positioned on an open sphere of radius $R_\mathrm{mic} = 0.4$ m in free-field conditions. The microphone array contains 64 sensors positioned at the same angular positions (azimuth and colatitude) as those of an Eigenmike em64 [15], so that an even coverage of the open sphere is ensured. The sensors capture the sound field of superposed plane waves, constituting a spatial sampling of the complex spectrum. Assuming the positions of the 64 sensors are given by $\{\mathbf{r}_{\mathrm{m}_1}, \mathbf{r}_{\mathrm{m}_2}, \ldots, \mathbf{r}_{\mathrm{m}_{64}}\}$, the corresponding sensor measurements form a complex vector $\mathbf{p}_\mathrm{m} = \{p(\mathbf{r}_{\mathrm{m}_1}, k), p(\mathbf{r}_{\mathrm{m}_2}, k), \ldots p(\mathbf{r}_{\mathrm{m}_{64}}, k)\} \in \mathbb{C}$ based on Eq. (2). A set of control points are defined inside the open sphere

$$\Omega = \{\mathbf{r}_{\mathrm{c}_1}, \mathbf{r}_{\mathrm{c}_2}, \ldots, \mathbf{r}_{\mathrm{c}_N}\} \subset \mathbb{R}^3, \tag{3}$$

where the task is to estimate the sound field based on the measurements $\mathbf{p}_\mathrm{m}$.

A complex-valued PINN is implemented to learn the unknown pressure field $p(\mathbf{r}_\mathrm{c}, k)$ at any control point from the sensor measurements $\mathbf{p}_\mathrm{m}$. Unlike a standard PINN, which might take the spatial coordinates $\mathbf{r}_\mathrm{c}$ as input, the proposed network also incorporates the microphone measurements associated with a given sound field. Specifically, for each generated sound field s, a vector of sensor data is defined as

$$\mathbf{p}_\mathrm{m}^{(s)} = \left[p^{(s)}(\mathbf{r}_\mathrm{m1}, k),\, p^{(s)}(\mathbf{r}_\mathrm{m2}, k), \ldots,\, p^{(s)}(\mathbf{r}_\mathrm{m64}, k)\right] \tag{4}$$

which captures the pressures recorded at the 64 microphone locations. Each entry reflects how the sound field computed from superpositions of plane waves interacts with the array geometry. The network is therefore designed to learn a mapping

$$\mathscr{N} : \left(\mathbf{r}_\mathrm{c}, \mathbf{p}_\mathrm{m}^{(s)}\right) \mapsto p^{(s)}(\mathbf{r}_\mathrm{c}, k), \tag{5}$$

where $\mathbf{r}_\mathrm{c} = (x_\mathrm{c}, y_\mathrm{c}, z_\mathrm{c}) \in \mathbb{R}^3$ is the location of the control point and $\mathscr{N}$ denotes the neural network model, viewed here as a nonlinear function that takes both the spatial coordinates and the microphone measurements as inputs and outputs the predicted pressure.

Including the microphone pressure vectors $\mathbf{p}_\mathrm{m}^{(s)}$ as part of the network input offers several important benefits. First, each sound field s generates a unique set of pressure measurements across the microphone array, capturing the spatial variations in phase and amplitude that depend on the direction of arrival of the individual plane waves and the constructive/destructive interferences. By conditioning the network on $\mathbf{p}_\mathrm{m}^{(s)}$, the model learns which sound field is being observed and can adapt its prediction of the pressure field accordingly. In contrast, if only the coordinate (x, y, z) were used, the network would be forced to learn a single field solution, failing to account for the differences in measurement patterns that arise from different superpositions of sound fields. Secondly, the pressure measurements naturally encapsulate valuable physical information relating to the interaction of the incident wave with the array, such as phase shifts and amplitude variations, which is essential for an accurate reconstruction of the sound field. Finally, this approach allows us to train a single network that can generalize across multiple sound fields. Thus, the inclusion of microphone pressures as part of the input not only preserves crucial physical context but also promotes efficiency and flexibility in handling sound fields consisting of superpositions of plane waves.

2.3 MSE and PDE control points

In the approach presented here, the interior control points Ω are split into two complementary sets (Ω_MSE, Ω_PDE) to make use of both direct supervision from known pressure values and physical constraints. The subset Ω_MSE contains only a small number Q of control points, where ground-truth pressure measurements are assumed to be available. At these locations, the mean squared error (MSE) is computed between the complex PINN's predicted pressure $\hat{p}^{(s)}(\mathbf{r}_\mathrm{c}, k)$ and the true pressure $p^{(s)}(\mathbf{r}_\mathrm{c}, k)$. This yields the loss term

$$\mathscr{L}_\mathrm{MSE} = \frac{1}{Q} \sum_{\mathbf{r}_{\mathrm{c}_q} \in Q} \left\| \hat{p}^{(s)}(\mathbf{r}_{\mathrm{c}_q}, k) - p^{(s)}\left(\mathbf{r}_{\mathrm{c}_q}, k\right) \right\|_2^2, \tag{6}$$

which compares the network solution to actual data at a number Q of critical points. Meanwhile, the larger set

Ω_{PDE} enforces the Helmholtz equation by minimizing the PDE residual adapted from Eq. (1) at L control points leading to the loss

$$\mathscr{L}_{\text{PDE}} = \frac{1}{L} \sum_{\mathbf{r}_{c_l} \in L} \left\| \frac{\nabla^2 \hat{p}(\mathbf{r}_{c_l}, k) + k^2 \hat{p}(\mathbf{r}_{c_l}, k)}{k^2} \right\|_2^2. \quad (7)$$

Note that the regular Helmholtz equation in Eq. (1) is normalized by k^2 and used as PDE loss. This operation results in keeping the Helmholtz residual on a comparable scale to the data loss, but it also diminishes the direct influence of the frequency on the training process. Especially at higher frequencies, the term k^2 grows quickly, resulting in an inflation of $\mathscr{L}_{\text{PDE}}$ relative to $\mathscr{L}_{\text{MSE}}$. Splitting the control points this way ensures that the network learns to match known pressures where they are available (through Ω_{MSE}) while also satisfying the fundamental wave equation over the entire domain (through Ω_{PDE}). The network is then trained using a weighted combination of the two losses, given by

$$\mathscr{L}_{\text{tot}} = \mathscr{L}_{\text{MSE}} + \alpha_{\text{PDE}} \mathscr{L}_{\text{PDE}}, \quad (8)$$

where α_{PDE} is a factor that weights the importance of the PDE loss.

2.4 Data generation

To form the training set, single frequency plane waves are first simulated arriving only in the horizontal plane, each having a random phase. All sound fields are simulated under free-field conditions. Fifty directions of arrival (DoAs) are defined around the open spherical microphone array of radius $R_{\text{mic}} = 0.4$ m, uniformly distributed in azimuth. From these 50 candidate plane waves, 5 distinct DoAs are randomly picked and their pressure fields are superposed, creating one combined sound field. By repeating this random selection process 10 times, 10 distinct superpositions are generated, each comprising 5 plane waves from different azimuthal directions. This superposition approach reflects more realistic acoustical scenarios, where multiple waves arrive simultaneously from various directions. This application is based on previous work [16], where a simple complex-valued MLP has been trained to estimate the sound field of single-frequency plane waves on a sphere of control points. Figure 1 shows an example of the plane waves grouped to create the superposed sound fields.

Due to the size of the open sphere and the number of microphones, the array has a known spatial aliasing frequency that can be calculated from the maximum spherical harmonic order of the array $N = 7$, the radius of the sphere $R_{\text{mic}} = 0.4$ m and the speed of sound $c = 343$ m/s as

$$f_{\text{alias}} = \frac{cN}{2\pi R_{\text{mic}}} \approx 955 \text{ Hz}. \quad (9)$$

In order to investigate the performance of the network at different frequencies, three different datasets are generated with plane wave frequencies $f_{\text{low}} = f_{\text{alias}}/2$, f_{alias} and $f_{\text{high}} = 2f_{\text{alias}}$. By generating these three different datasets, a variety of aliasing behaviors are captured and the proposed method can be investigated across different wave numbers. For each frequency, an analogous set of plane wave superpositions are generated, preserving the same DoA configurations and plane wave phases, while varying the wave number k.

The control points, at which the pressure field is to be reconstructed, are placed on the horizontal plane within a smaller sphere of radius 0.1 m. By limiting the control points to a two-dimensional cross-section, the spatial sampling is simplified while still capturing the critical features of the pressure field, making it sufficient for validating the reconstruction accuracy. To avoid spatial aliasing at these control points the spacing between neighboring points is set to $\frac{\lambda}{8}$, where λ is the wavelength corresponding to the highest simulated frequency (f_{high}). This ensures that the separation of control points is sufficiently fine to capture the pressure variations accurately. The resulting 61 control points are split into 5 MSE points (Ω_{MSE}), where target pressures are assumed to be known. These points correspond to the four diagonal (extreme) positions plus the center of the grid. The remaining 56 points (Ω_{PDE}) enforce the Helmholtz equation through the PDE loss. Thus, a small number of directly supervised points are combined with a larger set of physics-driven points to ensure the final pressure field solution is both data-consistent and physically accurate. Figure 2 illustrates the split between MSE and PDE control points.

For validation, a different set of control points is generated on the same horizontal plane but with a finer spacing of $\frac{\lambda}{16}$, resulting in 241 points. The

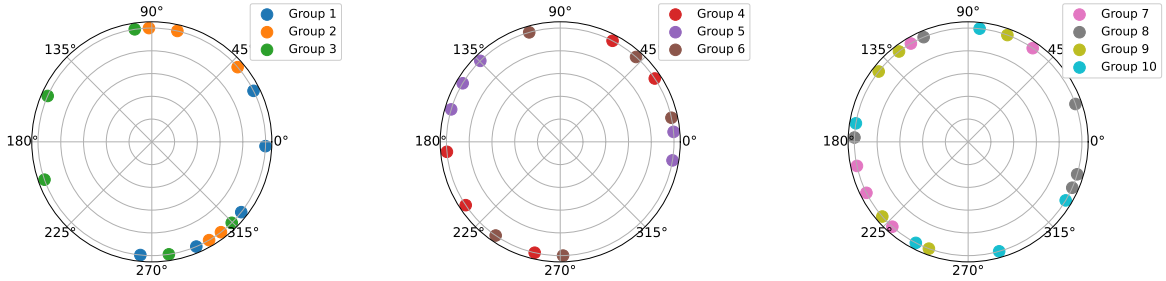


Fig. 1: Plane wave directions and their grouping into superposed sound fields

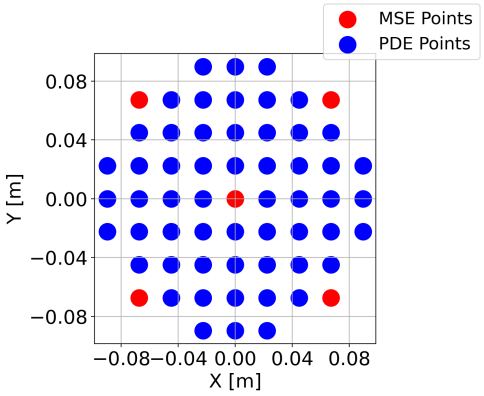


Fig. 2: Control points split into MSE and PDE points.

same superposed sound fields are kept during the validation process, but the networks performance is evaluated on this denser grid. This approach tests whether the model's predictions generalize well to new, more finely sampled positions inside the sphere.

Figure 3 shows an example of the sound field computed from the superposition of 5 plane waves and recorded by the microphone array on the open sphere, together with the sound field at the control points for training and validation.

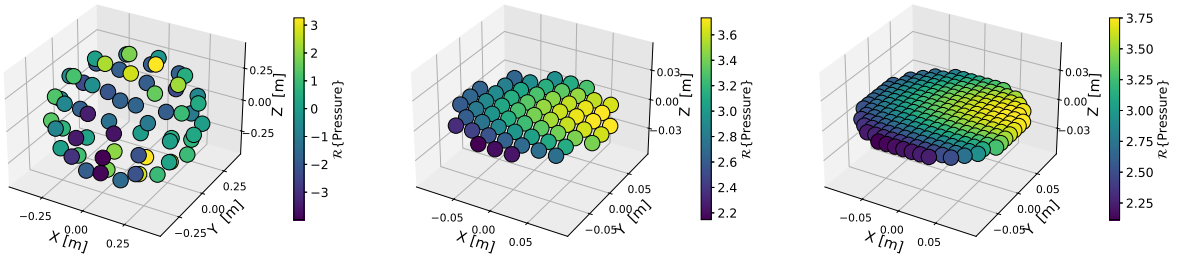
2.5 Network parameters

For each superposed sound field s, the complex pressure $\mathbf{p}_{\mathrm{m}}^{(s)}$ is computed at the 64 microphones. During training, these 64 complex-valued measurements are concatenated with the (x, y, z) coordinates of a selected control point $\mathbf{r}_{\mathrm{c}}$, also treated as a complex input by adding a zero imaginary part. Hence, each input vector is of length 67 (64 for the microphones plus 3 for the coordinates). The network then outputs a single complex value, corresponding to the predicted pressure at that control point $\mathbf{r}_{\mathrm{c}}$.

The proposed network architecture is a fully complex-valued multilayer-perceptron implemented using the complexPyTorch library [17]. Two hidden layers, each with 200 complex neurons were used and the activation function in each of them was the complex cardioid function [18] to effectively capture the phase-amplitude relationships in the data. The output layer contained a single complex neuron without an activation function to estimate the complex pressure at the control point.

For the MSE points (Ω_{MSE}), the $\mathscr{L}_{\mathrm{MSE}}$ was used during training, while for the PDE points (Ω_{PDE}), the normalised $\mathscr{L}_{\mathrm{PDE}}$ was used. Keeping a comparable scale between the two losses reduces the need for manual tuning of the loss weight λ_{PDE}, which is a well-known challenge when training PINNs [19]. An imbalance can lead to training difficulties, where one loss term may dominate, causing the network to prioritize either the data fit or the physical constraints inadequately.

During training, an initial learning rate of 0.01 was used and the model trained for 5000 epochs using a batch size of 64. A learning rate scheduler halved the rate every 1000 epochs, improving the convergence stability. The network weights were updated using the Adam optimizer. It is important to note that the same network architecture was trained individually from scratch for the three different datasets. All network models were trained on a MacBook Pro equipped with a 16-core CPU, 16-core Neural Engine and 36 GB of unified memory. Training each model took



(a) Open sphere **(b)** Training control points **(c)** Validation control points

Fig. 3: Example of sound field computed from a superposition of 5 plane waves at $f_{\text{low}} = 478$ Hz, corresponding to Group 2 (orange) in Figure 1.

Table 1: Comparison between the two network models in terms of averaged error expressed in dB.

	$f_{\text{low}} = 478$ Hz		$f_{\text{alias}} = 955$ Hz		$f_{\text{high}} = 1911$ Hz	
Control points	Train	Val	Train	Val	Train	Val
complexPINN	-14.6	-14.9	-5.1	-5.4	5.2	5.0
complexMLP	-9.9	-10.1	-2.9	-3.2	6.4	6.2

approximately 1 hour and 30 minutes. Data simulation and model training were performed using Python 3.11 and PyTorch 2.2.

3 Results

The objective is to evaluate the impact of incorporating physical laws into neural network training by comparing complex PINNs with standard complex neural networks that do not embed such physics-based constraints. To achieve this, two types of models are trained. The complexPINN models are trained based on the loss function presented in Eq. (8), with a weight $\alpha_{\text{PDE}} = 1$. The second type of models have $\alpha_{\text{PDE}} = 0$, effectively removing the physics constraint from the training process. Thus, the network learns only from the data-driven loss, which is based on the 5 control points Ω_{MSE} previously defined.

Table 1 shows the averaged error between the network predictions and the targets at both training and validation control points for the three different datasets corresponding to the different plane wave frequencies. The error is computed using Eq. (6). The error is first averaged over the control points for one sound field s and then averaged again over all 10 sound fields. The averaged errors from Table 1 show a couple of interesting observations for the models trained in this work. First, as expected, the sound field reconstruction accuracy at the control points is higher if the frequency is lower than the aliasing frequency. This is not surprising, since below the aliasing frequency, the spatial structure of the sound field can be extracted accurately from the microphone signals, while around and above f_{alias} the distance between the microphones is larger than the wavelength, which leads to an increase in sound field estimation error.

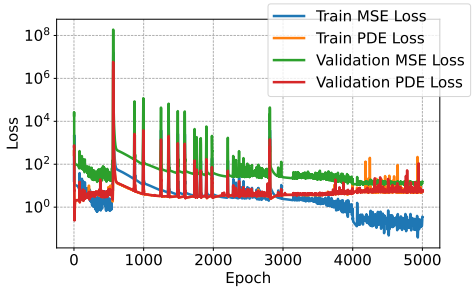
Incorporating the physical laws into the network model shows an improved sound field reconstruction. The complexPINN outperforms the complexMLP models at all investigated frequencies, for both training and validation control points. Interestingly, the reconstruction error at the validation control points is slightly lower than that at the training points. This may be attributed to several factors. Although both sets are drawn from the same grid, the training set is a sparser sampling (61 points) compared to the denser validation sampling (241 points). While this is a rather unusual way of splitting the training and validation control points, this was done intentionally to test the model's ability to generalize to a denser, unseen grid. This sparser selection may disproportionately include points in regions where the model's prediction error is higher, making the average error appear larger. In contrast, the denser validation set provides a more comprehensive sampling of the pressure field, so the averaging process smooths out localized high-error regions and results in a lower overall error.

Figure 4 shows the training and validation losses as

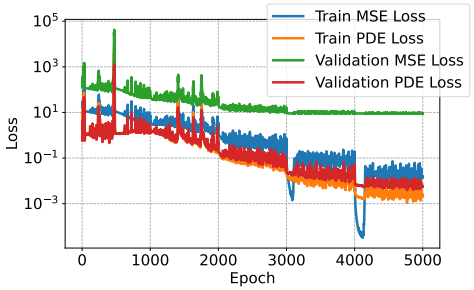
training progresses for both complexPINN and complexMLP models. The case shown here involves sound fields computed from plane waves at $f_{\text{alias}} = 955$ Hz. The plots display the following: the difference between predictions and targets at Ω_{MSE} (Train MSE Loss), the Helmholtz residual for the training control points Ω_{PDE} (Train PDE Loss), the difference between predictions and targets at validation control points (Validation MSE Loss) and Helmholtz residual at the validation points (Validation PDE Loss). First, it can be seen that adding the Helmholtz residual as regularization parameter in the model results in smoother training curves. By forcing the model to satisfy the Helmholtz equation during training, the loss landscape becomes less erratic, leading to more stable gradients and a more gradual reduction in the loss over epochs. This additional constraint effectively reduces the degrees of freedom of the solution.

Not surprisingly, if $\alpha_{\text{PDE}} = 0$, the model does not learn to satisfy the Helmholtz equation and therefore the PDE losses at training and validation control points (orange and red curves) remain constant during training. When $\alpha_{\text{PDE}} = 1$, the complexPINN model is learning to satisfy the Helmholtz equation as evidenced by the decreasing PDE losses over time. Interestingly, the Helmholtz residual at the training points behaves similarly to that at the validation points, even though the model does not see these control points during training. This can be explained by the fact that the validation control points are sampled from the same grid as the training points, just using a finer grid resolution. Because the solutions to the Helmholtz equation change gradually over small distances, once the model learns to satisfy the equation at one point, it is likely to achieve similar accuracy in neighboring regions. Toward the end of training, the PDE loss on the training points becomes slightly lower than on the validation points, which is expected because the model directly optimizes the physics constraint at the training locations. This result emphasises the model's ability to generalize the learned physics to unseen regions, even though it cannot perfectly satisfy the PDE at every point.

An interesting observation in Figure 4 is that, even though the complexPINN model significantly reduces the Helmholtz residual compared to the complexMLP, the overall error between predictions and targets at validation control points shows only a small improvement. This is not necessarily because the model prioritizes fitting the Ω_{PDE} data, but rather, it arises from the underdetermined nature of the Helmholtz equation in the absence of boundary conditions. With a 1:1 weighting between the data loss and the PDE loss, both are considered equally. However, since the Helmholtz equation admits an infinite family of solutions that locally satisfy the PDE, the network's single-neuron output for pressure may converge to a physically valid solution that minimizes the PDE residual without necessarily matching the desired target pressure. A possible solution to improve the performance would be to reduce α_{PDE} so that the model is encouraged to match the target pressure values. However, this comes with a trade-off, because the model might find a solution that is less physically consistent. Another approach would be to incorporate additional constraints during training.



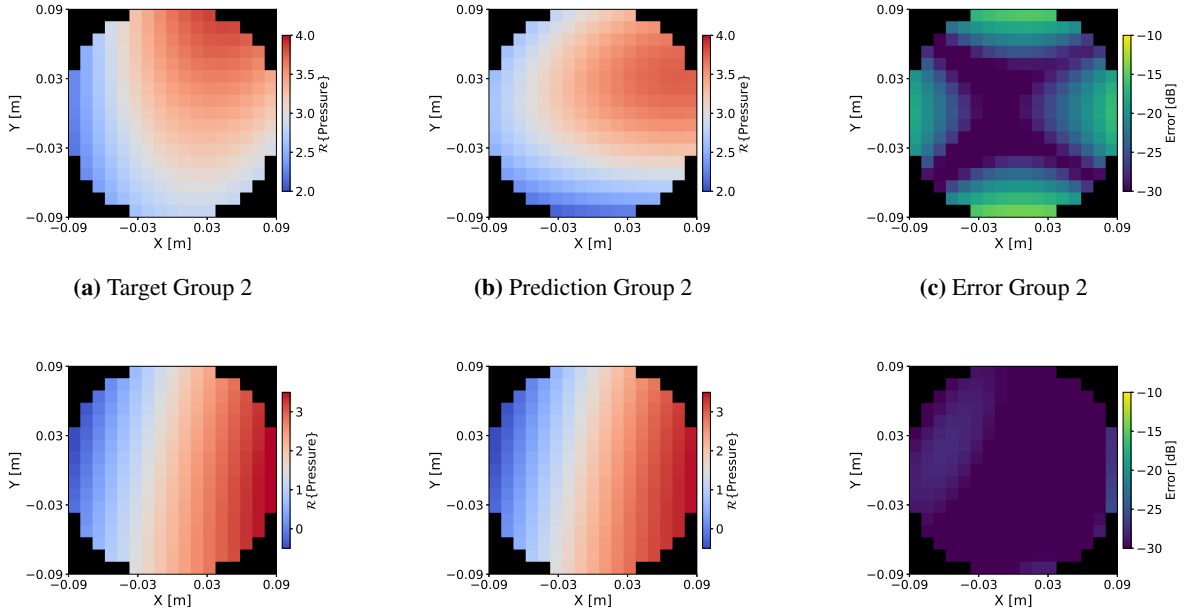
(a) complexMLP



(b) complexPINN

Fig. 4: Illustration of losses as training progresses for the complexMLP and complexPINN models for the dataset with plane waves at $f_{\text{alias}} = 955$ Hz.

It is also interesting to investigate how the network models perform for each individual sound field. Figure 7 shows the averaged error over training or

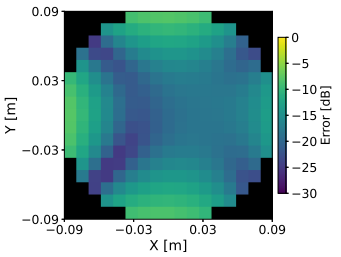


(a) Target Group 2 **(b)** Prediction Group 2 **(c)** Error Group 2

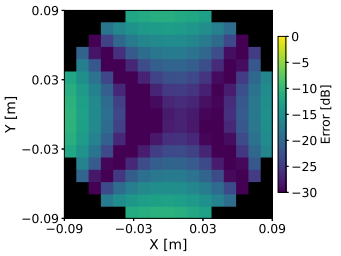
(d) Target Group 5 **(e)** Prediction Group 5 **(f)** Error Group 5

Fig. 5: Illustration of target, prediction and sound field error for two different groups of superposed single-frequency plane waves at $f_{\text{low}} = 478$ Hz. The plots are shown for the complexPINN model with $\alpha_{\text{PDE}} = 1$.

validation points for each of the 10 sound fields computed from the superposition of plane waves at frequency f_{low}. It can be observed that the performance of the network models varies across different superposed sound fields. This variability is not necessarily surprising and can be explained by the complexity of each sound field. When multiple plane waves are superposed, their relative directions and phase relationships produce interference patterns that can vary significantly in their complexity. As a result, the difficulty of accurately reconstructing the pressure field depends on the specific configuration of the underlying plane waves. In fields where the interference patterns are more complex, the network faces a greater challenge, leading to higher reconstruction errors. For example, the complex PINN achieves the lowest reconstruction error for the 5-th sound field, which corresponds to Group 5 in Figure 1. The superposed sound field in Group 5 is computed from adding the plane waves corresponding to the purple circles. Two of them are close to each other at 0° while the other three are also very close to each other between 135° and 180°. The superposition of these plane waves together with their assigned random phases creates a sound field that is less complex, compared for example to the sound field generated by the plane waves from Group 2. This can be seen in Figure 5, where the target and the estimated sound fields are shown for the validation control points as 2D plots together with the error distributed across the control points. The performance is shown for the complexPINN model with $\alpha_{\text{PDE}} = 1$. It can be observed that if the generated sound field is more complex due to the different plane wave directions and different phases (Group 2), the network achieves a lower reconstruction accuracy. Interestingly, in the error plot for Group 2, a pattern emerges: the reconstruction error is smaller along the two diagonals of the horizontal circle. This pattern likely corresponds to the locations of the Ω_{MSE} shown in Figure 2, where the network is directly supervised. In the complexPINN model, the Helmholtz equation helps propagate this low-error information to neighboring points, effectively broadening the region of accurate predictions. In contrast, the standard MLP, which relies solely on point-wise MSE, only achieves a narrow band of low error around these training points. This is shown in Figure 6 as a comparison of error plots between complexMLP and complexPINN for the



(a) complexMLP



(b) complexPINN

Fig. 6: Comparison of error plots between targets and estimations at the validation control points for the complexMLP and complexPINN models for the sound field generated in Group 10 with plane waves at $f_{\text{alias}} = 955$ Hz.

soundfield generated in Group 10 with plane waves at f_{alias}.

4 Conclusion

The work presented here demonstrates that incorporating physics-based constraints via complex-valued PINNs enhances the physical consistency of sound field reconstructions and improves the model's ability to estimate the sound field at control points, where it is assumed that no training data is available. By integrating measured pressure data as an additional input, the network can learn to estimate different sound fields and the reconstruction performance depends on the complexity of the sound field. Future work will focus on evaluating the network's ability to generalize to entirely new sound fields that were not included in the training set, as well as its performance when predicting pressure at control points sampled on grids that differ from those used during training. Additionally, the influence of the number and spatial distribution of the MSE training points on the model's accuracy could be investigated

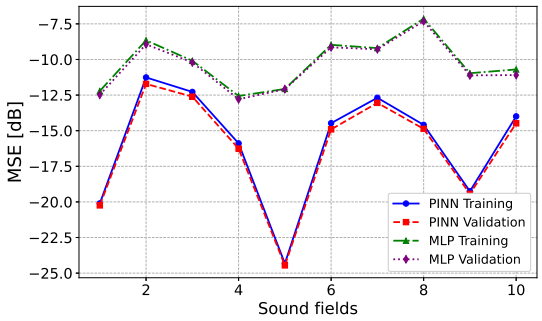


Fig. 7: Averaged reconstruction error across the different sound fields used during training for plane waves at frequency $f_{\text{low}} = 478$ Hz.

further. These studies will help refine the approach and further enhance its robustness for practical sound field reconstruction applications.

References

[1] Verburg, S. A. and Fernandez-Grande, E., "Reconstruction of the sound field in a room using compressive sensing," *The Journal of the Acoustical Society of America*, 143(6), pp. 3770–3779, 2018, ISSN 0001-4966, doi:10.1121/1.5042247.

[2] Fernandez-Grande, E., "Sound field reconstruction using a spherical microphone array)," *The Journal of the Acoustical Society of America*, 139(3), pp. 1168–1178, 2016, ISSN 0001-4966, doi:10.1121/1.4943545.

[3] Zotter, F. and Frank, M., *Ambisonics: A Practical 3D Audio Theory for Recording, Studio Production, Sound Reinforcement, and Virtual Reality*, Springer Topics in Signal Processing, Springer Cham, 1 edition, 2019, doi:10.1007/978-3-030-17207-7.

[4] Nelson, P. A., "Active control of acoustic fields and the reproduction of sound," *Journal of Sound and Vibration*, 177(4), pp. 447–477, 1994.

[5] Miotello, F., Comanducci, L., Pezzoli, M., Bernardini, A., Antonacci, F., and Sarti, A., "Reconstruction of Sound Field Through Diffusion Models," in *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1476–1480, 2024, doi:10.1109/ICASSP48485.2024.10446761.

[6] Fernandez-Grande, E., Karakonstantis, X., Caviedes-Nozal, D., and Gerstoft, P., "Generative models for sound field reconstruction," *The Journal of the Acoustical Society of America*, 153(2), pp. 1179–1190, 2023, ISSN 0001-4966, doi:10.1121/10.0016896.

[7] Koyama, S., Ribeiro, J. G. C., Nakamura, T., Ueno, N., and Pezzoli, M., "Physics-Informed Machine Learning for Sound Field Estimation: Fundamentals, state of the art, and challenges," *IEEE Signal Processing Magazine*, 41(6), pp. 60–71, 2024, doi:10.1109/MSP.2024.3465896.

[8] Ma, F., Abhayapala, T. D., and Samarasinghe, P. N., "Circumvent spherical Bessel function nulls for open sphere microphone arrays with physics informed neural network," in *10th Convention of the European Acoustics Association (Forum Acusticum 2023)*, Turin, Italy, 2023, doi:10.61782/fa.2023.0101.

[9] Olivieri, M., Karakonstantis, X., Pezzoli, M., Antonacci, F., Sarti, A., and Fernandez-Grande, E., "Physics-informed neural network for volumetric sound field reconstruction of speech signals," *EURASIP Journal on Audio, Speech, and Music Processing*, 2024(1), p. 42, 2024.

[10] Luan, X., Olivieri, M., Pezzoli, M., Antonacci, F., and Sarti, A., "Complex-Valued Physics-Informed Neural Network for Near-Field Acoustic Holography," in *2024 32nd European Signal Processing Conference (EUSIPCO)*, pp. 126–130, IEEE, 2024.

[11] Paul, V. S. and Nelson, P. A., "Efficient design of complex-valued neural networks with application to the classification of transient acoustic signals," *The Journal of the Acoustical Society of America*, 156(2), pp. 1099–1110, 2024, ISSN 0001-4966, doi:10.1121/10.0028230.

[12] Hirose, A., "Complex-valued neural networks: The merits and their origins," in *Proceedins of the International Joint Conference on Neural Networks*, pp. 1237–1244, 2009, doi:10.1109/IJCNN.2009.5178754.

[13] Scardapane, S., Van Vaerenbergh, S., Hussain, A., and Uncini, A., "Complex-Valued Neural Networks With Nonparametric Activation Functions," *IEEE Transactions on Emerging Topics in Computational Intelligence*, 4(2), pp. 140–150, 2020, ISSN 2471-285X, doi:10.1109/TETCI.2018.2872600.

[14] Kovacs, A., Exl, L., Kornell, A., Fischbacher, J., Hovorka, M., Gusenbauer, M., Breth, L., Oezelt, H., Yano, M., Sakuma, N., Kinoshita, A., Shoji, T., Kato, A., and Schrefl, T., "Conditional physics informed neural networks," *Communications in Nonlinear Science and Numerical Simulation*, 104, p. 106041, 2022, ISSN 1007-5704, doi:https://doi.org/10.1016/j.cnsns.2021.106041.

[15] mh acoustics, *em64 Eigenmike release notes*, 25 Summit Ave Summit, NJ 07901, 2023.

[16] Paul, V. S., Hahn, N., and Nelson, P. A., "Sound Field Analysis Using a Complex-Valued Neural Network," in *Proceedings of the Institute of Acoustics, Vol. 46, Pt. 3*, 2024.

[17] Matthès, M. W., Bromberg, Y., de Rosny, J., and Popoff, S. M., "Learning and Avoiding Disorder in Multimode Fibers," *Phys. Rev. X*, 11, p. 021060, 2021, doi:10.1103/PhysRevX.11.021060.

[18] Virtue, P., Yu, S. X., and Lustig, M., "Better than real: Complex-valued neural nets for MRI fingerprinting," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, pp. 3953–3957, 2017, doi:10.1109/ICIP.2017.8297024.

[19] Hua, J., Li, Y., Liu, C., Wan, P., and Liu, X., "Physics-informed neural networks with weighted losses by uncertainty evaluation for accurate and stable prediction of manufacturing systems," *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

Audio Engineering Society

Conference Paper 12

Presented at the AES International Conference on Artificial Intelligence and Machine Learning in Audio
2025 September 8–10, London, UK

This paper was peer-reviewed as a complete manuscript for presentation at this conference. This paper is available in the AES E-Library (http://www.aes.org/e-lib), all rights reserved. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Journal of the Audio Engineering Society.

Compressing Neural Network Models of Audio Distortion Effects Using Knowledge Distillation Techniques

Riccardo Simionato[1] and Aleksander Tidemann[1]

[1] *University of Oslo, Department of Musicology*

Correspondence should be addressed to Aleksander Tidemann (aleksander.tidemann@imv.uio.no)

ABSTRACT

Neural networks have proven to be effective for modeling analog audio effects using a black-box approach. However, few can guarantee lightweight solutions suitable for real-time environments where the models must run concurrently on consumer-grade equipment. This paper explores knowledge distillation techniques for compressing recurrent neural network models for audio distortion effects, aiming to produce computationally efficient solutions with high accuracy that maintain compact model size. In particular, we consider an audio-to-audio LSTM architecture for regression tasks where small networks are trained to mimic the internal representations of larger networks, known as feature-based knowledge distillation. The evaluation was conducted on three different audio distortion effect datasets with experiments on both parametric and non-parametric data. The results show that distilled models are more accurate than non-distilled models with equal parameter size, especially for models that exhibit higher error rates. Furthermore, we observe that smaller complexity gaps between student and teacher models yield greater improvements in non-parametric cases.

1 Introduction

Through their intricate physical circuits and non-linear characteristics, analog audio effects have a unique influence on sound. Attempting to capture and digitally emulate analog audio effects is a challenging task in the field of Virtual Analog (VA) modeling [1]. VA modeling approaches are often categorized as "white-box" or "black-box", where white-box modeling [2, 3] concerns the simulations of the circuitry starting from their mathematical description, and black-box approaches [4, 5] focus primarily on analyzing the target system's measurements in order to establish mapping relationships between inputs and outputs. Using neural networks for black-box modeling of audio effects has been fairly successful. In particular, Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) have shown significant promise for emulating various analog effects, such as tube amplification [6, 7], distortion [8, 9], optical dynamic range compression [10, 11], and other time-based effects [12, 13].

However, neural networks can become quite large and require significant operations per sample. This can hinder the real-time capabilities of models that strive for efficiency. In these cases, building lighter and smaller networks can be desirable. However, reducing the number of computational operations in a network is not trivial and is usually detrimental to model accuracy.

Many model compression techniques can be employed

to reduce the size of neural networks [14], including pruning, quantization, low-rank factorization, and knowledge distillation. The pruning technique focuses on diminishing the number of parameters in a neural network by identifying and removing redundant parameters that contribute less during training. This technique has also been used for VA applications [15], where authors were able to compress recurrent networks while maintaining significant performance. Quantization compresses the network by lowering the number of bits required to represent each weight. Low-rank factorization helps identify redundant parameters by utilizing matrix and tensor decomposition, which entails decomposing larger matrices into smaller ones. Lastly, knowledge distillation involves training a small network to mimic the complex representations of a larger and more complex model. Once the larger network achieves good generalization and performance on unseen data, its knowledge can be transferred to the smaller networks through distillation. This training process can maintain model performance and reduce model complexity to a significant degree [16, 17].

Knowledge distillation for audio has been primarily explored for classification tasks. Data-free distillation for audio classification was proposed in [18], using synthetic data to generate transfer sets to train student networks without the need for the original dataset. Experiments to increase the classification accuracy of student networks through increased temporal dependencies have also been examined in [19], where attention weights are extracted from transformer-based teacher models and aligned with student models using KL divergence loss. Collaborative learning frameworks with multiple audio feature representations have also been successful in improving the accuracy of networks for classification tasks [20].

However, there are only a few cases where knowledge distillation is explored for audio-related regression tasks [21] and no such study, to our knowledge, in the field of VA modeling of analog audio effects. This paper presents a preliminary study into the effects of knowledge distillation on neural networks for black-box modeling of audio distortion effects. In particular, we explore to what degree feature-based knowledge distillation can reduce the size of said networks while maintaining comparable performance, enhancing the accuracy of smaller networks. Three datasets of audio distortion effects were used for evaluation, including the Electro Harmonix Big Muff Pedal and Blackstar HT-1 amp previously explored in [9, 15], as well as parametric and non-parametric versions of the analog modeled overdrive plugin DrDrive [1] [22]. Results compare the performance of distilled models, which are trained to mimic the behavior of the larger model, against non-distilled equivalent models, which are traditionally trained, quantitatively and through MUSHRA-style listening tests.

The rest of the paper is organized as follows. Section 2 details the knowledge distillation techniques chosen. In Section 3, the neural architectures designed for black-box VA modeling are presented. Section 4 explains the setup of our experiments to determine the effect of knowledge distillation on the accuracy and size of our distortion models. Section 5 presents the experiment results, while Section 6 concludes the paper.

All the source code for this paper, along with audio examples and trained models, is available online [2].

2 Knowledge Distillation For Neural Networks

Knowledge distillation is a technique used to compress complex and large "teacher" networks into smaller "student" networks [17, 16]. Knowledge distillation and other model compression techniques offer ways to minimize the computational expenses of working with neural networks, as well as to optimize models for deployment and gain real-time performance advantages.

There exist three primary types of knowledge distillation for neural networks, each with a unique approach to the distillation process: response-based, feature-based, and relation-based distillation. In our study, we use feature-based knowledge distillation, a suitable and popular type of Knowledge Distillation for regression tasks that explores compression of neural audio distortion effects [23, 24].

2.1 Feature-based Distillation

The process of feature-based knowledge distillation is characterized by the transfer of internal representations, or features, of a teacher network to a smaller student network. By retaining these high-level feature extraction capabilities, feature-based distillation allows student models to capture more nuanced representations

[1] https://www.audiority.com/shop/dr-drive/
[2] https://github.com/aleksati/kd-audio-fx

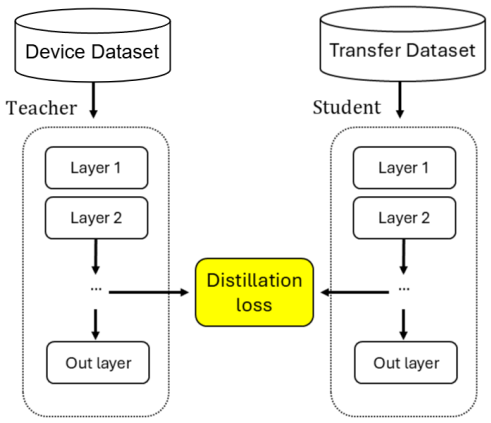


Fig. 1: Feature-based distillation. During training, the feature representations from the last LSTM layers of both the teacher and student models are aligned using a designated distillation loss function.

of the input data, leading to improved generalization and performance.

With feature-based distillation, specific layers of the teacher model are identified as feature extractors that capture important intermediary feature representations of the input data. The intermediate layers of a student network are then aligned to those of the teacher. During training, the student model learns to produce feature maps similar to the teacher model, inheriting its structure and hierarchical understanding of the data. The loss function is defined to minimize the difference between the feature representations of the teacher and student models at these corresponding layers, as illustrated in Figure 1. Traditionally, this process employs the Kullback–Leibler (KL) divergence to measure the difference between the probability distributions derived from the feature representation vectors [16]. However, alternative loss functions can also be utilized [25]. Additionally, this approach can be combined with additional learning to match the output predictions.

3 Neural Network Architecture

Our proposed distillation architecture utilizes recurrent LSTM neural networks, a well-researched and effective framework for building neural models of distortion and other audio effects [7, 11, 12].

Further, we consider two architectural variations: the depth of the networks and their size. The first teacher model, denoted as `H`, consists of 7 LSTM layers configured in an hourglass shape, with 8, 16, 32, 64, 32, 16, and 8 units, as illustrated in Figure 2. This design was determined through preliminary experiments, aiming to establish a versatile teacher model that consistently outperforms typically trained student models across all datasets. A second teacher model, denoted as `S`, consists of 2 LSTM layers, with 64 and 8 units, respectively.

The student models consist of 2 LSTM layers, where the first layer can be configured to be 8, 16, 32, or 64 units, and the final layer is fixed at 8 units to match the final layer of the teacher model. These models will be referred to `S-64-8`, `S-32-8`, `S-16-8`, `S-8-8`. An additional student model, referred to as `S-8`, is configured with only a single 8-unit LSTM layer.

In all models, the input and output layers consist of a linear fully connected layer with a single unit, each processing 1 sample of audio per iteration during inference. Additionally, a skip connection is employed, linking the input directly to the output layer. This design helps integrate raw input features into the final predictions, providing an alternative path for the gradient and improving model convergence.

The conditioning layer is designed to influence the model by incorporating the user-controllable 'Drive' knob parameter, which introduces a gradual increase in distortion, as found in the DrDrive plugin. The conditioning process utilizes Feature-wise Linear Modulation (FiLM) and the Gated Linear Unit (GLU) [26, 22]. This layer employs conditioning information to calculate coefficients that modulate the internal representation of the input signal. The output is then passed to the GLU, functioning as a logical gate. The conditioning layer is positioned before the final 8-unit LSTM layer. Consequently, the alignment of feature representations occurs after the conditioning information has already modulated the signal.

4 Experiment Design

In this section, we provide a detailed overview of the dataset and the training strategy employed.

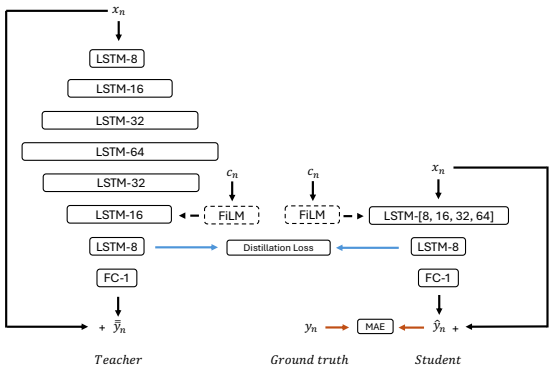


Fig. 2: Teacher `H` in the hourglass shape (left) and student (right) architectures. The teacher model features 7 LSTM layers, while the student model has only 2 LSTM layers. A conditioning layer is introduced for the parametric case, and it is positioned before the final LSTM layer in both the teacher and student models.

4.1 Dataset

The modeled devices include the Blackstar HT-1 vacuum tube amplifier (HT-1), Electro-Harmonix Big Muff (Big Muff) guitar pedal, and the analog-modeled overdrive plugin DrDrive. The HT-1 is a single-watt dual-channel 8" tube amplifier with classic valve harmonics that offers natural overdrives through gain controls and its "Infinite Shape Feature" (ISF). The Big Muff is a famous distortion guitar pedal effect that produces heavily distorted sounds through its dual cascaded diode clipping stages. The pedal features sustain and volume (pre- and post-gain) control, tone adjustment, and an on/off switch. Finally, DrDrive is a free VST plugin from Audiority[3] that offers an analog simulation of an overdrive pedal. Not counting the extended functionalities, DrDrive features four parameter controls for drive, brightness, attack, and level (gain).

Our distillation architecture was evaluated on four datasets, one for each of the HT-1 and Big Muff devices, and two for DrDrive. The HT-1 and Big Muff datasets were identical to those in [15, 9], with the same technical and training configurations. For DrDrive, both parametric and non-parametric sets were considered.

The DrDrive datasets consist of mono audio files with a 10-minute duration, sampled at 48 Khz, where the effect was added to a variety of musical sources ranging from frequency sweeps and white noise with various amplitudes, to physical recordings of musical instruments, such as bass, drums, piano, pads and sections from various pop/rock songs. In preparation for training, 10% of the datasets (1 min) were allocated for the test and validation sets each. In the case of the parametric dataset, the full range of the four VST control parameters was first scaled to the range $[0, 1]$. Then, three different values of the Drive amount were considered as the parametric conditioning (0, 0.5, and 1). For each conditioning instance, fixed values were assigned to the three other VST parameters, attack (0), level (0.3), and brightness (0.5). The datasets were collected using dedicated software for generating datasets for audio-related ML tasks, namely the DGMD presented in [27].

[3] `https://www.audiority.com/shop/dr-drive/`

4.2 Training Strategy

Training is conducted for up to $1,000$ epochs, with an early stopping mechanism in place to halt training if there is no reduction in validation loss for 30 consecutive epochs. We utilize the Adam optimizer, starting with an initial learning rate of 3×10^{-4}. This learning rate is halved if no improvements are seen after 5 epochs. The batch size is set to 8 examples, with each example composed of $2,048$ input audio samples. These configurations are used within the framework of Truncated Backpropagation Through Time (TBPTT). The same strategy is applied in training both the student and teacher models. Test losses and evaluation metrics are computed using the model's weights, which minimize the validation loss throughout the training epochs.

4.3 Loss Function

The teacher and student models included in the experiments are trained using the Mean Absolute Error (MAE) as a loss function. The distillation employed a combination of two loss functions. The feature representations from the last LSTM layers are aligned using the KL divergence of the probability distribution retrieved from the feature representation vectors:

$$\frac{e^{z_i}}{\sum_j e^{z_j}} \tag{1}$$

where z_i represents a single element in the feature vector, and the denominator is the sum of the exponentials

of all elements in the feature vector. This loss function is denoted as L_D. The feature representations from the last teacher's LSTM layers are pre-computed to be used as targets during the learning process.

The second loss function L_{MAE} is based on MAE, which is computed from the model's output and the ground truth. These two loss functions are combined into a single loss value, with each component weighted by factors that change over time:

$$L = \alpha(e) \cdot L_D + \beta(e) \cdot L_{MAE} \tag{2}$$

where e represents the epoch number. Specifically, α initial values is 1. and β is 0. α decrease of 10% each epoch, while β is computed as $1 - \alpha$. This weighting strategy enables dynamic optimization, balancing the emphasis primarily on accurately mimicking the teacher's feature representations and subsequently achieving precise output predictions.

4.4 Evaluation Metrics

To assess the accuracy of the model, we also computed the error-to-signal-ratio (ESR):

$$\frac{\sum_{n=0}^{N-1} |y[n] - \hat{y}[n]|^2}{\sum_{n=0}^{N-1} |y[n]|^2} \tag{3}$$

where y and $\hat{y}$ are the target and prediction, respectively, and the multi-resolution STFT error, which is defined as:

$$\begin{aligned} &\frac{1}{N}\sum_{m} \left|\left| \frac{|STFT_m(y)| - |STFT_m(\hat{y})|}{|STFT_m(y)|} \right|\right|_1 \\ &+ \frac{1}{N}\sum_{m} ||\log(|STFT_m(y)|) - \log(|STFT_m(\hat{y})|)||_1 \end{aligned} \tag{4}$$

where $m = [256, 512, 1024]$.

5 Results

Table 1 presents the test loss results of the hourglass-type `H` teacher models, while Table 2 displays the performance of both distilled and non-distilled student models, `S-64-8`, `S-32-8`, `S-16-8`, `S-8-8`. Overall, these results highlight the positive impact of distillation, with teacher models achieving the highest accuracies.

We observe that student models typically exhibit improved performance with larger hidden layer sizes, meaning that the effectiveness of distillation is proportional to the size of the student models, as can be expected. Only the parametric DrDrive case, which incorporates the user-controllable 'Drive' knob as a parameter to condition the audio processing, shows a different trend, performing better with a lower size. The parametric results show that distillation works when aligning the feature after conditioning, a task that introduces additional complexity. Also, the non-parametric set of DrDrive and Big Muff exhibit negative improvements at 8 units, but quickly recover at 16 and beyond, following an upward trend.

The HT-1 dataset emerges as the most challenging case, as it produces the largest errors when compared to other datasets. On the other hand, HT-1 demonstrates the most significant improvements through distillation. In contrast, when non-distilled students are already performing well, the benefits of distillation are less pronounced.

Table 2 also reports the number of parameters and related floating-point operations per sample (FLOP). Compared to teacher models, the FLOP count shows that student models require significantly fewer parameters and operations per sample, representing a big improvement in memory requirements and efficiency.

Figure 3 takes a closer look at the performance advantages of distilled versus non-distilled models with respect to the size of their hidden layers. STFT and MAE loss show relatively stable performance across all models, following an upward trend. In contrast, ESR results exhibit greater variability, with both extreme negative and positive values. This greater variance could be because ESR is a normalized function, and the distillation improves more when the signal's amplitude is larger than low. Lastly, HT-1 and parametric DrDrive datasets show larger improvements at lower sizes, an opposite trend compared to the other datasets.

The results further pinpoint that the `H` teacher model is not always effective for the smallest students. Table 3 presents the results of our experiments using the teacher `S` model, which exhibits a smaller complexity gap compared to the students. This scenario shows improvements even in the smallest student models. In particular, both DrDrive and Big Muff demonstrated advantages from distillation, as well as HT-1, although to a slightly lesser extent. Similar to the `H` teacher scenario, the performance improvements of students

Table 1: Test performance and loss of `H` teacher models with related number of parameters and floating-point operations per sample (FLOPs).

Device dataset	Total params	FLOPs	MSE	MAE	ESR	STFT
HT-1	49385	98768	$4.20 \cdot 10^{-3}$	$3.92 \cdot 10^{-2}$	4.11	0.15
Big Muff	-	-	$4.00 \cdot 10^{-4}$	$5.30 \cdot 10^{-3}$	1.48	0.24
DrDrive (un-cond)	-	-	$3.00 \cdot 10^{-4}$	$1.08 \cdot 10^{-2}$	0.63	0.10
DrDrive (cond)	49993	98898	$6.71 \cdot 10^{-5}$	$4.63 \cdot 10^{-3}$	0.15	0.05

Table 2: Test performance and loss of student models (`S-64-8`, `S-32-8`, `S-16-8`, `S-8-8`) trained on `H` teacher model, with related number of parameters and floating-point operations per sample (FLOPs). Instances with the best performance are highlighted in bold. Variations in the hidden size of the network's LSTM layer are presented row-by-row.

Device dataset	Student	Total params	FLOP count	MSE non-distilled	MSE distilled	MAE non-distilled	MAE distilled	ESR non-distilled	ESR distilled	STFT non-distilled	STFT distilled
HT-1	`S-8-8`	873	1744	$4.91 \cdot 10^{-2}$	$4.97 \cdot 10^{-2}$	$1.44 \cdot 10^{-1}$	$1.34 \cdot 10^{-1}$	41.97	24.92	0.56	0.59
	`S-16-8`	1961	3856	$2.72 \cdot 10^{-2}$	$1.34 \cdot 10^{-2}$	$1.07 \cdot 10^{-1}$	$7.78 \cdot 10^{-2}$	23.72	11.17	0.42	0.30
	`S-32-8`	5673	11152	$1.17 \cdot 10^{-2}$	$1.00 \cdot 10^{-2}$	$6.65 \cdot 10^{-2}$	$6.34 \cdot 10^{-2}$	12.51	8.96	0.26	0.24
	`S-64-8`	19241	38032	$9.47 \cdot 10^{-3}$	$\mathbf{8.69 \cdot 10^{-3}}$	$5.99 \cdot 10^{-2}$	$\mathbf{5.85 \cdot 10^{-2}}$	8.60	**6.73**	0.24	**0.23**
Big Muff	`S-8-8`	873	1744	$1.03 \cdot 10^{-3}$	$9.75 \cdot 10^{-4}$	$9.88 \cdot 10^{-3}$	$1.38 \cdot 10^{-2}$	5.41	6.00	0.47	0.49
	`S-16-8`	1961	3856	$1.10 \cdot 10^{-3}$	$8.83 \cdot 10^{-4}$	$1.11 \cdot 10^{-2}$	$1.35 \cdot 10^{-2}$	4.90	5.32	0.60	0.48
	`S-32-8`	5673	11152	$1.06 \cdot 10^{-3}$	$8.86 \cdot 10^{-4}$	$1.01 \cdot 10^{-2}$	$9.31 \cdot 10^{-3}$	4.92	2.69	0.51	0.47
	`S-64-8`	19241	38032	$5.50 \cdot 10^{-4}$	$\mathbf{4.39 \cdot 10^{-4}}$	$6.47 \cdot 10^{-3}$	$\mathbf{5.70 \cdot 10^{-3}}$	1.82	**1.24**	0.32	**0.27**
DrDrive (un-cond)	`S-8-8`	873	1744	$2.97 \cdot 10^{-3}$	$2.59 \cdot 10^{-3}$	$3.38 \cdot 10^{-2}$	$3.08 \cdot 10^{-2}$	5.92	6.19	0.29	0.23
	`S-16-8`	1961	3856	$1.20 \cdot 10^{-3}$	$1.00 \cdot 10^{-3}$	$1.97 \cdot 10^{-2}$	$1.84 \cdot 10^{-2}$	1.97	1.87	0.17	0.15
	`S-32-8`	5673	11152	$1.08 \cdot 10^{-3}$	$7.00 \cdot 10^{-4}$	$1.91 \cdot 10^{-2}$	$1.59 \cdot 10^{-2}$	2.00	1.35	0.16	0.13
	`S-64-8`	19241	38032	$6.81 \cdot 10^{-4}$	$\mathbf{4.53 \cdot 10^{-4}}$	$1.49 \cdot 10^{-2}$	$\mathbf{1.23 \cdot 10^{-2}}$	1.16	**0.90**	0.13	**0.10**
DrDrive (cond)	`S-8-8`	1049	1810	$3.63 \cdot 10^{-3}$	$3.30 \cdot 10^{-3}$	$3.31 \cdot 10^{-2}$	$3.15 \cdot 10^{-2}$	7.74	**3.68**	0.30	0.26
	`S-16-8`	2569	3986	$2.67 \cdot 10^{-3}$	$\mathbf{2.36 \cdot 10^{-3}}$	$2.88 \cdot 10^{-2}$	$\mathbf{2.78 \cdot 10^{-2}}$	5.13	3.69	0.24	**0.22**
	`S-32-8`	7913	11410	$3.70 \cdot 10^{-3}$	$3.57 \cdot 10^{-3}$	$3.27 \cdot 10^{-2}$	$3.21 \cdot 10^{-2}$	7.11	4.22	0.30	0.26
	`S-64-8`	27817	38546	$4.42 \cdot 10^{-3}$	$3.84 \cdot 10^{-3}$	$3.46 \cdot 10^{-2}$	$3.29 \cdot 10^{-2}$	8.02	4.60	0.30	0.25

Table 3: Test performance and loss of student models (`S-32-8`, `S-16-8`, `S-8-8`) trained on `S` teacher model, with the related number of parameters and floating-point operations per sample (FLOPs). Instances with the best performance are highlighted in bold. Variations in the hidden size of the network's LSTM layer are presented row-by-row.

Device dataset	Student	Total params	FLOP count	MSE non-distilled	MSE distilled	MAE non-distilled	MAE distilled	ESR non-distilled	ESR distilled	STFT non-distilled	STFT distilled
HT-1	`S-8-8`	873	1744	$4.91 \cdot 10^{-2}$	$4.10 \cdot 10^{-2}$	$1.44 \cdot 10^{-1}$	$1.36 \cdot 10^{-1}$	41.97	30.65	0.54	0.49
	`S-16-8`	1961	3856	$2.72 \cdot 10^{-2}$	$2.48 \cdot 10^{-2}$	$1.07 \cdot 10^{-1}$	$1.04 \cdot 10^{-1}$	23.72	20.10	0.42	0.40
	`S-32-8`	5673	11152	$\mathbf{1.17 \cdot 10^{-2}}$	$1.72 \cdot 10^{-2}$	$\mathbf{6.65 \cdot 10^{-2}}$	$8.35 \cdot 10^{-2}$	**12.51**	13.46	**0.26**	0.33
Big Muff	`S-8-8`	873	1744	$1.00 \cdot 10^{-3}$	$1.00 \cdot 10^{-3}$	$1.00 \cdot 10^{-2}$	$1.00 \cdot 10^{-2}$	5.41	3.83	0.47	0.45
	`S-16-8`	1961	3856	$1.10 \cdot 10^{-3}$	$9.99 \cdot 10^{-4}$	$1.11 \cdot 10^{-2}$	$9.98 \cdot 10^{-3}$	4.90	3.86	0.60	0.45
	`S-32-8`	5673	11152	$5.50 \cdot 10^{-4}$	$\mathbf{5.19 \cdot 10^{-4}}$	$1.01 \cdot 10^{-2}$	$\mathbf{6.48 \cdot 10^{-3}}$	4.92	**1.85**	0.51	**0.32**
DrDrive (un-cond)	`S-8-8`	873	1744	$2.40 \cdot 10^{-3}$	$2.23 \cdot 10^{-3}$	$2.98 \cdot 10^{-2}$	$2.81 \cdot 10^{-2}$	5.92	5.12	0.23	0.23
	`S-16-8`	1961	3856	$1.20 \cdot 10^{-3}$	$\mathbf{5.45 \cdot 10^{-4}}$	$1.97 \cdot 10^{-2}$	$\mathbf{1.34 \cdot 10^{-2}}$	1.97	0.96	0.17	**0.11**
	`S-32-8`	5673	11152	$1.08 \cdot 10^{-3}$	$5.76 \cdot 10^{-4}$	$1.91 \cdot 10^{-2}$	$\mathbf{1.34 \cdot 10^{-2}}$	2.00	**0.93**	0.16	**0.11**
DrDrive (cond)	`S-8-8`	1049	1810	$3.63 \cdot 10^{-3}$	$3.12 \cdot 10^{-3}$	$3.31 \cdot 10^{-2}$	$3.09 \cdot 10^{-2}$	7.74	3.50	0.30	0.26
	`S-16-8`	2569	3986	$2.67 \cdot 10^{-3}$	$3.02 \cdot 10^{-3}$	$2.88 \cdot 10^{-2}$	$3.05 \cdot 10^{-2}$	5.13	3.42	**0.24**	0.25
	`S-32-8`	7913	11410	$3.70 \cdot 10^{-3}$	$\mathbf{9.60 \cdot 10^{-4}}$	$3.27 \cdot 10^{-2}$	$\mathbf{2.11 \cdot 10^{-2}}$	7.11	**1.45**	0.30	0.27

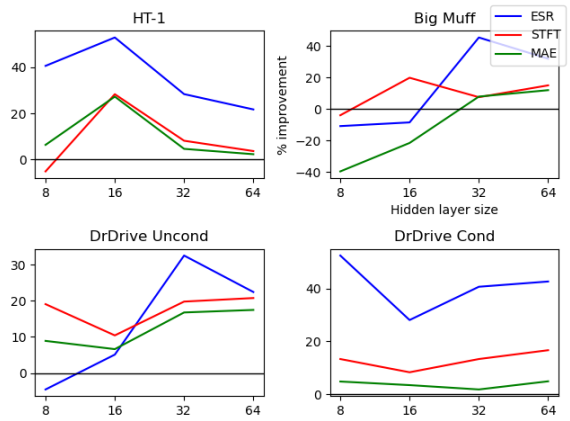


Fig. 3: Plots detailing the improvements, in percent (%), of using distilled models over non-distilled models for different hidden layer sizes, with the `H` teacher. Comparisons are made using ESR, STFT, and MAE loss metrics.

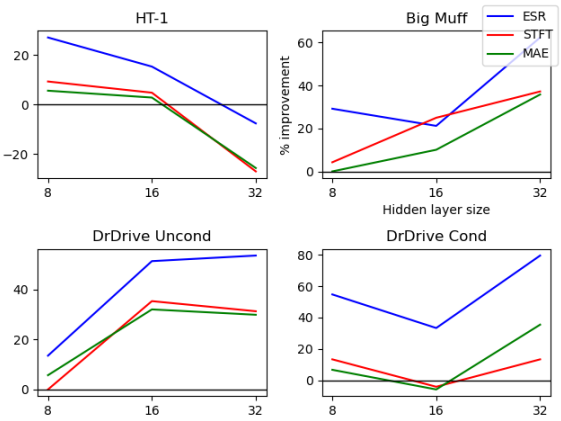


Fig. 4: Plots detailing the improvements, in percent (%), of using distilled models over non-distilled models for different hidden layer sizes, with the `S` teacher. Comparisons are made using ESR, STFT, and MAE loss metrics.

distilled from a teacher `S` model tend to be more significant as their size increases. This trend is illustrated in Figure 4, which compares the performance of distilled and non-distilled models in relation to the size of their hidden layers. The improvements in STFT and MAE metrics are nearly identical, while the ESR results closely mirror these trends. Notably, except for the HT-1 dataset, the performance gains observed with distilled models are greater than those achieved in the `H` teacher case. Moreover, these improvements tend to increase as the model sizes grow.

Finally, Table 4 presents the results for a distilled student model with a single LSTM layer (`S-8`) trained on both the `H` and `S` teachers. These cases demonstrate that a smaller complexity gap between teachers and students typically results in enhanced performance, showing that distillation is effective in the majority of cases. However, the HT-1 dataset does not show significant improvements. We also observe that the parametric DrDrive student performs best when trained by the `H` teacher model.

5.1 Listening tests

A MUSHRA-style listening test was conducted to investigate the perceptual quality of distilled versus non-distilled models [28]. Audio examples from all three device datasets were included, along with parametric examples of DrDrive. However, only audio from `S-64-8` student models trained on the `H` teacher model was considered, as these models achieved the lowest test loss on average. In addition, we included audio examples from `S-8-8` student models trained on the `S` teacher model.

The listening test included two sets of audio examples per device model, comprising both distilled and non-distilled examples. When considering both the `S-64-8` and `S-8-8` cases, this amounted to sixteen conditions total for one listening trial. In each trial condition, five audio examples and one reference were presented to participants, consisting of the audio distortion from the device effects applied to either guitar, bass, or drum signals. Participants were asked to rate the perceptual quality of each audio example against the reference on a scale from 0 to 100. The reference audio acted as the ground truth and was therefore sourced directly from the device dataset. The remaining five audio examples included a hidden reference, low and mid-range anchors, where reference duplicates are passed through low-pass filters at 3.5KHz and 7KHz, respectively, as well as the outputs of the distilled and non-distilled models.

Of the thirteen people who participated in the listening tests, three were removed in the post-screening as they

Table 4: Test performance and loss of student models (`S-8`) trained on `H` and `S` teacher models, with the related number of parameters and floating-point operations per sample (FLOPs) are reported. Instances with the best performance are highlighted in bold.

Device dataset	Teacher	Total params	FLOP count	MSE non-distilled	MSE distilled	MAE non-distilled	MAE distilled	ESR non-distilled	ESR distilled	STFT non-distilled	STFT distilled
HT-1	S	329	656	$5.03 \cdot 10^{-2}$	$\mathbf{5.47 \cdot 10^{-2}}$	$1.50 \cdot 10^{-1}$	$\mathbf{1.57 \cdot 10^{-1}}$	39.19	**30.07**	0.61	0.61
-	H	-	-	-	$5.58 \cdot 10^{-2}$	-	$1.60 \cdot 10^{-1}$	-	30.99	-	0.61
Big Muff	S	-	-	$1.00 \cdot 10^{-3}$	$\mathbf{9.87 \cdot 10^{-4}}$	$9.83 \cdot 10^{-3}$	$\mathbf{9.83 \cdot 10^{-3}}$	5.57	4.12	0.51	**0.48**
-	H	-	-	-	$1.00 \cdot 10^{-3}$	-	$9.97 \cdot 10^{-2}$	-	**4.02**	-	0.51
DrDrive (un-cond)	S	-	-	$3.98 \cdot 10^{-3}$	$\mathbf{1.28 \cdot 10^{-3}}$	$3.97 \cdot 10^{-2}$	$\mathbf{1.83 \cdot 10^{-2}}$	6.97	**1.07**	0.33	**0.20**
-	H	-	-	-	$3.07 \cdot 10^{-3}$	-	$3.45 \cdot 10^{-1}$	-	7.51	-	0.26
DrDrive (cond)	S	337	666	$1.60 \cdot 10^{-3}$	$1.26 \cdot 10^{-3}$	$3.60 \cdot 10^{-2}$	$2.41 \cdot 10^{-2}$	3.75	3.41	0.32	0.28
-	H	-	-	-	$\mathbf{1.23 \cdot 10^{-3}}$	-	$\mathbf{2.39 \cdot 10^{-2}}$	-	**3.25**	-	**0.27**

scored the hidden reference below 80 in more than 15% of trial conditions. The trials were conducted using a pair of Beyerdynamic DT 770 Pro headphones.

As presented in Figure 5, results show high overall scores for distilled and non-distilled examples. However, the perceptual improvements of distillation are not pronounced, as non-distilled models already start with high accuracy. This finding was further substantiated by p-values gathered from statistical T-tests comparing listening test scores from distilled and non-distilled models. On all devices, the p-value scores were greater than 0.05, meaning it's still inconclusive whether there are definitive perceptual advantages of distilled models over non-distilled models. On the other hand, we observe a slight preference for distilled models in most cases, especially in the `S-64-8` students. Non-distilled models also show greater variability, suggesting slightly less consistency in their perceived quality over their distilled counterparts.

6 Conclusion

The paper explores knowledge distillation techniques to reduce the size of large neural network models and enhance the accuracy of smaller ones in the context of emulating audio distortion effects. In particular, the study examines feature-based knowledge distillation for audio-to-audio modeling on both non-parametric and parametric datasets, using an LSTM architecture to train smaller student models that replicate the internal representations of larger teacher networks. Our results indicate that the process of distillation can be used to optimize training for VA audio models, as we notice improved model accuracy when comparing distilled and non-distilled networks of equal parameter size. Notably, the distillation improvements are more pronounced as student models increase in size. The results also highlight the known challenges of using feature-based distillation techniques, demonstrating that it is more difficult for models to learn from teachers that present particularly more complexity. Finally, listening tests provided additional qualitative data on the perceptual effects of distillation for audio effects modeling. Although not statistically conclusive, the listening test results show positive tendencies toward using distillation to improve the perceptual quality of audio effect models.

For future work, more exhaustive architectural modifications should be considered. Additionally, designing more efficient loss functions could help refine the distance metrics between the probability distributions of teachers and students. Finally, employing a multi-teacher ensemble approach could be advantageous in mitigating potential complexity gaps between models.

References

[1] Pekonen, J. and Välimäki, V., "The brief history of virtual analog synthesis," in *Proc. 6th Forum Acusticum. Aalborg, Denmark: European Acoustics Association*, 2011.

[2] Rest, M., Parker, J. D., and Werner, K. J., "WDF Modeling of a Korg MS-50 Based Non-linear Diode Bridge VCF," in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, 2017.

[3] Werner, K. J., Bernardini, A., Smith, J. O., and Sarti, A., "Modeling Circuits With Arbitrary

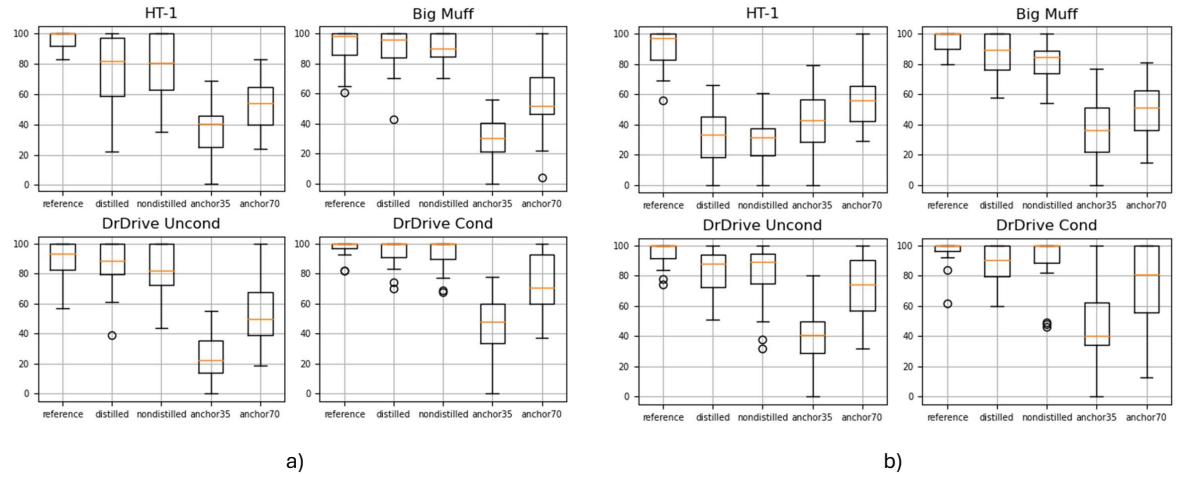


Fig. 5: Box plot showing the rating results of the listening tests for `S-64-8` students trained on `H` teacher (a) and `S-8-8` students trained on `S` teacher (b), for HT-1 (top-left), Big Muff (top-right), non-parametric DrDrive (bottom-left), and parametric DrDrive (bottom-right). The orange horizontal lines indicate the median value, and the outliers are displayed as circles.

Topologies and Active Linear Multiports Using Wave Digital Filters," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 65(12), 2018, doi:10.1109/TCSI.2018.2837912.

[4] Eichas, F. and Zölzer, U., "Black-box modeling of distortion circuits with block-oriented models," in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, 2016.

[5] Holmes, B. and Van Walstijn, M., "Physical model parameter optimisation for calibrated emulation of the Dallas Rangemaster Treble Booster guitar pedal," in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, 2016.

[6] Damskägg, E.-P., Juvela, L., Thuillier, E., and Välimäki, V., "Deep learning for tube amplifier emulation," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019.

[7] Covert, J. and Livingston, D., "A vacuum-tube guitar amplifier model using a recurrent neural network," in *Proceedings of IEEE Southeastcon*, 2013.

[8] Damskagg, E.-P., Juvela, L., and Valimaki, V., "Real-Time Modeling of Audio Distortion Circuits with Deep Learning," in *Proceedings of the Sound and Music Computing conference (SMC)*, 2019.

[9] Wright, A., Damskägg, E.-P., and Välimäki, V., "Real-Time Black-Box Modelling With Recurrent Neural Networks," in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, 2019.

[10] Steinmetz, C. J. and Reiss, J. D., "Efficient neural networks for real-time modeling of analog dynamic range compression," in *152nd Convention of the Audio Engineering Society (AES)*, 2022.

[11] Simionato, R. and Fasciani, S., "Modeling Time-Variant Responses of Optical Compressors with Selective State Space Models," *Journal of Audio Engineering Society*, 73(3), pp. 144–165, 2025.

[12] Wright, A. and Välimäki, V., "Neural Modeling of Phaser and Flanging Effects," *Journal of the Audio Engineering Society*, 69(7/8), 2021, doi: 10.17743/jaes.2021.0029.

[13] Mikkonen, O., Wright, A., Moliner, E., and Välimäki, V., "Neural modeling of magnetic tape

recorders," *International Conference on Digital Audio Effects (DAFx)*, 2023.

[14] Marinó, G. C., Petrini, A., Malchiodi, D., and Frasca, M., "Deep neural networks compression: A comparative survey and choice recommendations," *Neurocomputing*, 520, 2023.

[15] Südholt, D., Wright, A., Erkut, C., and Välimäki, V., "Pruning Deep Neural Network Models of Guitar Distortion Effects," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 31, 2023, doi:10.1109/TASLP.2022.3223257.

[16] Hinton, G., Vinyals, O., and Dean, J., "Distilling the Knowledge in a Neural Network," in *NIPS Deep Learning and Representation Learning Workshop*, 2015.

[17] Gou, J., Yu, B., Maybank, S. J., and Tao, D., "Knowledge Distillation: A Survey," *International Journal of Computer Vision*, 129(6), 2021, doi: 10.1007/s11263-021-01453-z.

[18] Kang, Z., He, Y., Wang, J., Peng, J., Qu, X., and Xiao, J., "Feature-Rich Audio Model Inversion for Data-Free Knowledge Distillation Towards General Sound Classification," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023.

[19] Choi, K., Kersner, M., Morton, J., and Chang, B., "Temporal Knowledge Distillation for on-device Audio Classification," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022.

[20] Gao, L., Xu, K., Wang, H., and Peng, Y., "Multi-representation knowledge distillation for audio classification," *Multimedia Tools and Applications*, 81(4), 2022, doi:10.1007/s11042-021-11610-8.

[21] Gholami, B., El-Khamy, M., and Song, K., "Knowledge Distillation for Tiny Speech Enhancement with Latent Feature Augmentation," in *Proceedings of Interspeech*, 2024.

[22] Simionato, R. and Fasciani, S., "Conditioning Methods for Neural Audio Effects," in *Proceedings of the Sound and Music Computing conference (SMC)*, 2024.

[23] Passalis, N. and Tefas, A., "Learning deep representations with probabilistic knowledge transfer," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[24] Xu, Q., Wu, K., Wu, M., Mao, K., Li, X., and Chen, Z., "Reinforced Knowledge Distillation for Time Series Regression," *IEEE Transactions on Artificial Intelligence*, 2023.

[25] Park, W., Kim, D., Lu, Y., and Cho, M., "Relational Knowledge Distillation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[26] Perez, E., Strub, F., De Vries, H., Dumoulin, V., and Courville, A., "Film: Visual reasoning with a general conditioning layer," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[27] Fasciani, S., Simionato, R., and Tidemann, A., "A Universal Tool for Generating Datasets from Audio Effects," in *Proceedings of the Sound and Music Computing conference (SMC)*, 2024.

[28] Schoeffler, M., Bartoschek, S., Stöter, F.-R., Roess, M., Westphal, S., Edler, B., and Herre, J., "webMUSHRA—A comprehensive framework for web-based listening tests," *Journal of Open Research Software*, 6(1), 2018.

Audio Engineering Society
Conference Paper 13
Presented at the AES International Conference on Machine Learning and Artificial Intelligence for Audio
2025 September 8-10, London, UK

This paper was peer-reviewed as a complete manuscript for presentation at this conference. This paper is available in the AES E-Library (http://www.aes.org/e-lib), all rights reserved. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Journal of the Audio Engineering Society.

A Scalable AI Architecture for Audio and Multimodal Analysis on Mobile Devices: A Case of Environmental Monitoring

Marina Eirini Stamatiadou[1], Athanasia Mpesmerti[1], Nikolaos Vryzas[1], Lazaros Vrysis[1], Charalampos Dimoulas[1]

[1] *Multidisciplinary Media & Mediated Communication Research Group (M3C), Aristotle University of Thessaloniki, 54124 Thessaloniki, Greece*

ABSTRACT

The increasing need for real-time environmental monitoring has made Artificial Intelligence (AI) and Machine Learning (ML) models for audio classification and multimodal sensing essential for detecting and analyzing pollution-related sounds. In cases where mobile devices are used for capturing and processing audiovisual content, such models offer significant potential for research, storytelling, and public awareness. This study proposes a scalable and modular architecture that enables direct and flexible access to AI models for multimodal and audio processing. A CNN-LSTM hybrid model is trained for real-time environmental sound classification and deployed as a service. Building on prior 1D CNN-based approaches and incorporating temporal dependencies, the new model achieves an AUC of 0.91, demonstrating improved accuracy and generalization. The system leverages a REST API and Docker-based containerization, allowing deployment of independent AI services and supporting mobile and IoT use cases. The architecture accommodates both pre-trained and custom models, accessible from any device via a unified interface, confirming that a hybrid CNN-LSTM topology can support effective real-time sound classification, within a modular, containerized framework.

Keywords: Environmental Audio, CNN-LSTM, REST API, Multimodal, Microservices, Real-time AI

1. Introduction

1.1. Background and motivation

The rapid advancement of technology and the proliferation of Artificial Intelligence (AI) and Machine Learning (ML) models have opened the way to numerous applications that process audio, from speech recognition and music generation, to audio-event detection and environmental monitoring. In the context of environmental awareness, sound classification plays a crucial role in detecting sources of pollution, such as industrial noise, traffic congestion, and construction activities [1], [2]. Environmental sound monitoring (ESM) plays a crucial role in modern cities, where air and noise pollution affect public health and overall quality of life, as well as natural ecosystems, and is becoming a growing concern. Noise pollution exacerbates stress, anxiety, depression [3], hypertension, and a higher risk of cardiovascular diseases [4], [5], particularly in urban areas, where prolonged exposure disrupts sleep and learning, and is linked to cognitive impairments, especially for children near highways or construction zones [6]. Additionally, ESM contributes to indirect air pollution detection by analysing urban soundscapes and identifying acoustic footprints associated with major sources of air pollutants.

Although significant progress has been noted in audio processing algorithms, there are cases in which computational requirements, limited mobility or restricted public accessibility, prevent their

applicability and exploitation in real-world scenarios. The ubiquity of smartphones and other mobile devices presents an unprecedented opportunity for ESM. These devices offer significant advantages over traditional fixed monitoring stations, including mobility, computational capabilities, and widespread availability. Their integrated sensors (i.e., microphone, camera, GPS) enable comprehensive data collection, while on-device AI processing facilitates immediate noise classification. The contextual richness of mobile data collection integrates noise measurements with location data, temporal patterns, and environmental conditions, providing multidimensional and multi-modal insights. Additionally, mobile devices allow dual functionality, in both offline and online modes, a fact that makes them particularly valuable for continuous environmental monitoring across diverse settings—from densely populated urban centers to remote natural habitats.

Lately, several smart city implementations [7], [8], [9], exploit mobile-based noise monitoring solutions and address the need for extensive coverage through distributed citizen participation, rather than relying on limited fixed sensor networks. Their capability to provide (near) real-time analysis allow for immediate pollution detection and pollution levels exceedance, enabling prompt intervention measures. From an economic perspective, smartphones are a remarkably cost-effective solution compared to traditional monitoring systems that require stationary audio sensors, significantly reducing the infrastructure investment by leveraging existing personal devices. Another important aspect is that mobile platforms excel at dynamic tracking, offering the unique capability to monitor transient pollution sources such as temporary construction sites, public events, or changing traffic patterns that fixed sensors might miss entirely. The crowdsourcing dimension of mobile-based environmental monitoring transforms passive citizens into active environmental stewards. By engaging the public through accessible and user-friendly applications [10][11], this approach generates substantial datasets for comprehensive urban noise mapping while simultaneously raising environmental awareness. Sound patterns captured during environmental monitoring serve as valuable proxies for detecting air quality issues. The distinctive acoustic signatures are not only correlated with noise pollution but also indicate the presence of common air pollutants. This relationship enables environmental scientists to use audio data to identify potential air quality hotspots, effectively extending a single microphone's capabilities beyond noise measurement to provide indicators of broader environmental conditions.

Our research addresses the critical gap between advanced AI capabilities and practical field deployment, by developing a lightweight, accessible architecture that democratizes access to sophisticated environmental sound analysis. It builds upon previous work in environmental sound classification, where a one-dimensional Convolution Neural Network (1D CNN) classification model was employed to distinguish between audio footprints that **are** or **are not** related with air pollution [11], [12] (R/NR classes respectively). While the 1D CNN approach demonstrated promising classification performance, we extended our methodology by integrating a Long Short-Term Memory (LSTM) layer, enabling the system to capture temporal dependencies effectively and model sequential patterns in environmental audio data. The proposed classification model is transformed into an independently deployable microservice and leverages REST API technology to become accessible as a web service, enabling seamless integration and real-time environmental monitoring from any internet-connected mobile device. This approach eliminates traditional barriers to AI deployment by abstracting the technical complexity behind standardized API calls, allowing non-technical users to benefit from advanced audio processing capabilities without specialized knowledge. By creating this bridge between sophisticated ML models and everyday environmental monitors, we aim to significantly expand participation in pollution detection efforts.

1.2. Paper structure

The rest of the paper is structured as follows: in section 2 previous related work is presented, in order to give a comprehensive background in the underlying research fields. Section 3 outlines the methodology that has been followed for building and evaluating the overall system. Section 4 highlights the experimental process and the results that came out of it. Finally, section 5 discusses the conclusions of the research and opens the way for future work and further research in the field.

2. Related work

2.1 AI in Environmental Sound Monitoring

Urban noise classification has gained high attention in the last decades due to the increasing demand for accurate environmental monitoring within the context of pollution analysis and mitigation strategies.

Previous research has explored the use of Deep Learning models, especially CNNs and Recurrent Neural Networks (RNNs), for environmental sound classification [13]-[15]. Recent advancements in IoT-based urban noise monitoring exploit the integration of embedded systems like Raspberry Pi with deep learning models, leveraging data normalization, augmentation, and 2D CNN architectures to achieve high-accuracy real-time sound classification [13]. Additionally, Audio Entailment [17] enhances complex audio scene understanding, enabling multimodal fusion of audiovisual data with other environmental inputs for more accurate pollution reasoning.

In parallel, studies have demonstrated that hybrid deep learning models, such as CNN-LSTM architectures, achieve superior sound classification by combining feature extraction with temporal modelling [18]. Edge devices, like Jetson Nano, have been tested in wireless sensor networks, integrating MFCC and CNN-LSTM models to achieve low-latency, high-accuracy classification, addressing stability and real-time performance limitations in existing systems [19]. Meanwhile, an optimized Bayesian ensemble-based 1D-CNN has been tested for environmental sound classification, achieving state-of-the-art accuracy on raw waveforms with low computational cost [20]. Latest research leverages Sequential CNNs and VGG19 CNNs, which have demonstrated high accuracy in processing and analysing urban soundscapes, while adding an explainability layer using techniques such as Grad-CAM to enhance model interpretability, transparency, trust, and applicability in real-world noise monitoring applications [21], [22].

2.2 Microservices and REST APIs for AI Applications

A microservice-based architecture decomposes applications into independent services, each being responsible for a specific functionality and database. This software design approach allows modular development and easier maintenance [23], enhancing resilience by isolating faults within individual services. While microservices were initially deployed in virtual machines (VMs), they have evolved with containerization, which optimizes resource use and improves deployment efficiency [24]. Similar to common service-based architectures they also offer flexibility, enabling development in different programming languages and environments depending on specific needs and/or prior knowledge [25]. RESTful API-based communication is a scalable and robust approach for modern applications that facilitates easy integration with client applications, making machine learning and AI models more accessible to users, through a process known as *Restification*. This allows developers to train a model and expose it via HTTP requests (GET, POST, PUT, DELETE), enabling clients to send data and receive predictions.

In the context of environmental sound classification, this approach aligns with advancements in music information retrieval, where microservices have been successfully used to detect and analyze audio patterns, as seen in systems like MusicTalk [26]. A key advantage of deploying audio classification models as RESTful microservices is their ability to function independently while interacting with other services for multimodal environmental analysis. Studies in musical instrument recognition have shown how microservice-based architectures enable the integration of independent detectors while ensuring compatibility across datasets and applications. Similarly, in environmental sound monitoring, microservices allow multiple models—such as noise classification, air pollution inference from soundscapes, and event detection—to be dynamically linked, ensuring flexible and efficient system expansion [11], [12], [27], [28].

Building upon recent advances in both environmental sound classification and service-oriented architectures, our research synthesizes key insights from both domains. Real-time AI applications require efficient model architectures and optimized deployment strategies. Prior approaches have explored the use of edge computing and lightweight AI models, but these solutions often trade accuracy for speed. Our approach balances both by leveraging a CNN-LSTM hybrid architecture. As demonstrated in [29], encapsulating machine learning models as REST microservices removes significant implementation barriers while enhancing scalability. Similarly, the success of microservice architectures in music information retrieval systems like MusicTalk [26] provides a proven framework for audio pattern recognition which we have adapted for environmental audio classification and pollution monitoring. This convergence enables non-technical users to leverage sophisticated audio classification capabilities through standardized API calls, while maintaining the flexibility to incorporate new models as environmental science advances.

3. Methodology

The proposed system consists of a 1D-CNN-LSTM hybrid model to leverage both spatial and temporal

features in audio signals, in order to identify sources of air pollution. The developed model is encapsulated and containerized, deployed on the cloud, offered via an API as a REST microservice.

3.1 Dataset

The initial UrANP dataset, which contains audio clips related and not related to air pollution, was constructed using samples from the benchmark datasets UrbanSound8k and ESC-50 [27], [28]. These clips were manually selected and categorized into two distinct classes: *pollution-related (R)* and *non-pollution-related (NR)* sounds. The final dataset was formed by concatenating the selected clips into two separate waveform files, one for each class, using the Audacity software. This format allows the direct feeding of raw waveform input into the 1D CNN network that will be presented next. To further enhance the dataset, and analyze the model's performance across different dataset sizes, additional audio segments sourced from YouTube were incorporated into both categories, with durations of 1 hour, 30 minutes, 12 minutes, and 6 minutes, resulting in four training variations with total dataset durations of 10, 9, 8:24, and 8:12 hours.

3.2 Model training and optimization

The optimization of a classification model can range from modifications in the dataset to a complete restructuring of the training architecture. Built on the foundations of previous research [2], [11], [30] and further extended to optimize its accuracy, the 1D-CNN model is trained to classify environmental sounds in two distinct categories: related and not-related to air pollution sources (PR/NPR). Significant improvements in the output results of classifiers have been observed with the use of k-fold cross-validation [31], and the inclusion of data shuffling to enhance model generalization [32]. An alternative to k-fold cross-validation is described in the Scikit-learn library, which introduces the "ShuffleSplit" function. Unlike k-fold cross-validation, ShuffleSplit randomly selects training and validation subsets in each iteration, providing greater variability in data partitioning. Following a similar approach but within the context of k-fold cross-validation, we implement data shuffling before each partitioning step, ensuring a more balanced distribution of training samples across different folds, following the logic of stratified ShuffleSplit, as shown in Figure 1. Our optimization process includes the steps of hyperparameter configuration, k-fold cross validation and data shuffling.

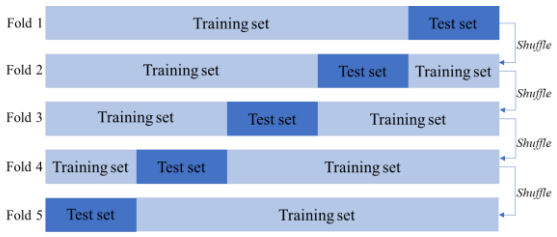


Figure 1: Data shuffling in 5-fold cross validation

3.2.1 1D-CNN classifier optimization

The optimization of the 1D-CNN classifier was attempted through deliberate manual hyperparameter tuning rather than automated methods, as the baseline model already showed promising accuracy. This decision was taken in order to ensure a balance between model performance and computational efficiency. Key parameters (i.e., CNN filters, kernel size, dropout rates) were selected and adjusted differently, analyzing each modification's effect on validation accuracy and potential overfitting. The hyperparameter optimization process confirmed that the initial configuration is the optimal one. The details are shown in Table 1.

Layer (type)	**Output shape**
Input Layer	(22050, 1)
Conv1D (F=32, CS=8)	(22043, 32)
MaxPooling1D (KS=8)	(2755, 32)
Dropout (10%)	(2755, 32)
Conv1D (F=64, KS=8)	(2748, 64)
MaxPooling1D (KS=8)	(343, 64)
Dropout (10%)	(343, 64)
Conv1D (F=96, KS=8)	(336, 96)
MaxPooling1D (KS=8)	(42, 96)
Dropout (10%)	(42, 96)
Conv1D (F=128, KS=8)	(35, 128)
MaxPooling1D KS=8)	(4, 128)
Dropout (10%)	(4, 128)
GlobalAveragePooling1D	(128,)
Dense (N=16, Fu:ReLU)	(16,)
Dropout (10%)	(16,)
Dense (N=2, Fu:Softmax)	(2,)

KS: Kernel Size, F: Filters, Fu: Function, N: Neurons
Table 1: 1D-CNN model architecture and hyperparameter configuration

Further enhancement of the model's generalization capabilities included k-fold cross-validation with additional fold configurations of k = 5 and k= 7, to

assess the optimal balance between accuracy's consistency and variance reduction. Best results were achieved for k=5 folds.

3.2.2 LSTM layer

To further optimize the model, a Long Short-Term Memory (LSTM) layer was added after the CNN, by removing the Global Average Pooling layer, as it reduces data dimensions and would prevent the LSTM from processing sequential dependencies. In a second training approach both Dropout and Max Pooling layers were removed, allowing the LSTM to be applied directly after the final Conv1D layer, preserving 35 timesteps. This approach retains more temporal information compared to maintaining Dropout and Max Pooling, which would have reduced the sequence length to 4. The final model architecture is presented in Table 2.

Layer (type)	**Output shape**
Input Layer	(22050, 1)
Conv1D (F=32, CS=8)	(22043, 32)
MaxPooling1D (KS=8)	(2755, 32)
Dropout (10%)	(2755, 32)
Conv1D (F=64, KS=8)	(2748, 64)
MaxPooling1D (KS=8)	(343, 64)
Dropout (10%)	(343, 64)
Conv1D (F=96, KS=8)	(336, 96)
MaxPooling1D (KS=8)	(42, 96)
Dropout (10%)	(42, 96)
Conv1D (F=128, KS=8)	(35, 128)
LSTM (64 units)	(64,)
Dropout (10%)	(64,)
Dense (N=16, Fu:ReLU)	(16,)
Dense (N=2, Fu:Softmax)	(2,)

KS: Kernel Size, F: Filters, Fu: Function, N: Neurons
Table 2: The hybrid CNN-LSTM model architecture

3.3 Restification

To enable real-time interaction with the trained classification model, a REST API was developed, using Flask, deploying the model as a service. The process includes Flask installation along with its dependencies, server configuration, routes and endpoints definition for the HTTP requests. Upon receiving an audio file, the system stores it in a temporary directory, allowing for immediate access during processing and ensuring file deletion after the prediction is completed to maintain system cleanliness.

1. For real-time handling, audio input is segmented into fixed-duration waveform chunks of 1 second with 0.5 seconds overlap, prior to inference. These buffered segments are forwarded to the classification model. This approach simulates real-time streaming and enables low-latency predictions even from non-continuous audio sources. The trained model is invoked via the "/predict" endpoint to classify the incoming audio as PR (pollution-related) or NPR (non-pollution-related), and the outcome is returned as a structured JSON response. In case of failure, the system includes robust error-handling routines to ensure reliability.

2. Finally, the service is configured to automatically launch and remain active, allowing users to send audio files for classification via a simple HTTP request. This architecture supports remote access, enabling real-time predictions from any connected device without requiring local model deployment, facilitating seamless integration into mobile and web-based applications.

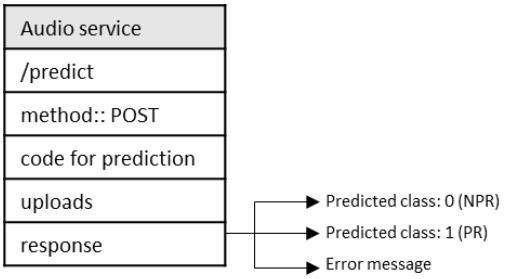


Figure 3: Audio classification model as a service

3.4 Integrated architecture with complementary services

Following the successful deployment of the hybrid audio classification model as a RESTful service, the next step expands the system to integrate additional models. For the needs of the current research, a topic modeling service (processing text) and an image event identification service (identifying pollution sources) were incorporated. To achieve seamless interoperability while preserving the modularity of each service, Docker containerization was employed, ensuring efficient management, scalability, and standardized execution across diverse computing environments. The first step in the containerization process is the selection of a base image, the definition of the workspace, copy and installation of the dependencies, and integration of the necessary model

files and application code. The ports are exposed to enable service and external communication.

In the final stage of the micro-service architecture implementation the three services were deployed on a shared cloud server, (Google Cloud), to achieve centralized management and secure access to each service through unique URLs that are automatically generated upon deployment. Cloud hosting ensures that these services remain accessible for real-time utilization of the models from the main application as well as any external interface.

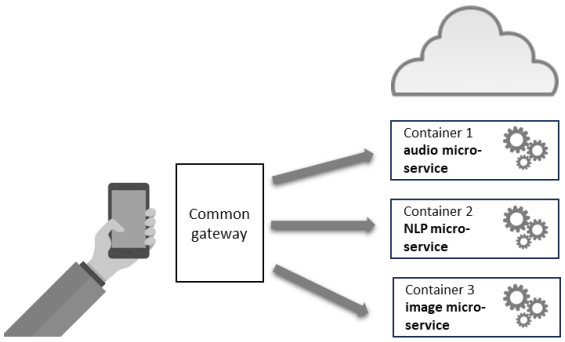


Figure 4: The proposed architecture

4. Experimental Results

The evaluation of the first model's training was conducted using precision, recall, F1-score, and accuracy, measured across each fold of k-fold cross-validation. These metrics quantify model performance, with precision assessing the accuracy of positive predictions, recall measuring the detection of actual positives, F1-score balancing both, and accuracy indicating overall correctness. The average accuracy across folds was reported, alongside the final fold's metrics for a representative performance snapshot. In cases of significant performance improvement, ROC-AUC analysis was performed using an independent dataset, visualizing model discrimination ability through True/False positive rate (TPR/FPR) calculations, with AUC measuring classification effectiveness.

Class	Precision	Recall	F1-score
0 (NPR)	0.77	0.75	0.76
1(PR)	0.76	0.77	0.77
Mean accuracy	**78.5 ±1.5%**		

Table 3: Baseline model evaluation

4.1 Model evaluation with manual hyperparameter configuration

Changes in hyperparameters and the overall architecture of the initial convolutional network resulted in significantly lower performance across evaluation metrics. Consequently, the original structure was retained, with subsequent optimization efforts focusing on specific aspects of the model to improve evaluation metrics.

4.2 Model evaluation with k-fold cross validation (k=5 and k=7)

In 5-fold cross-validation, performance slightly declined, with an average accuracy of 78.2% and 71.3% in the final fold, while variability increased (±4.5% deviation), indicating reduced model stability. In 7-fold cross-validation, accuracy further dropped to 76.4% overall and 70.1% in the final fold, with an even higher deviation (±6.1%), suggesting increased instability. These results indicate that increasing the number of folds did not enhance accuracy but rather introduced greater variability, confirming that 3-fold cross-validation was the most suitable choice for training the baseline model.

4.3 Model evaluation with shuffling

Using random shuffling (via numpy.random.shuffle function) for each fold significantly improved model performance, increasing accuracy from 78.5% to 92.5%. This enhancement indicates that random data redistribution strengthened the model's ability to generalize to new data, reducing overfitting and increasing sample diversity per fold. The large dataset size further supported generalization, allowing the model to learn more representative patterns and avoid dependence on specific data structures.

The results of data shuffling combined with 5-fold and 7-fold cross-validation showed that model accuracy remained consistently high at 93%, indicating only a slight improvement over 3-fold cross-validation. The stability of these results across different fold numbers suggests that the model successfully generalized due to the large dataset size and random data redistribution, which increased sample diversity in each fold.

However, although evaluation metrics showed a noticeable improvement, the AUC score of 0.87 (Figure 5) suggests that while the model fits the training data well, its ability to generalize remains uncertain. Using ROC-AUC analysis with an independent dataset, only a slight improvement was observed (AUC increased from 0.86 to 0.87),

indicating the higher reliability of ROC-AUC for performance evaluation, and the potential for further model optimization to enhance generalization.

	k=3		k=5		k=7	
Class	0	1	0	1	0	1
Precision	0.91	0.95	0.90	0.93	0.90	0.93
Recall	0.96	0.91	0.93	0.90	0.95	0.90
F1-Score	0.93	0.93	0.92	0.92	0.94	0.92
Mean Accuracy	92.5±0.5%		93±0.7%		93±0.7%	

Table 5: Model evaluation after data shuffling for k-fold cross validation, for k=3, 5, 7

4.4 Hybrid (CNN-LSTM) Model evaluation

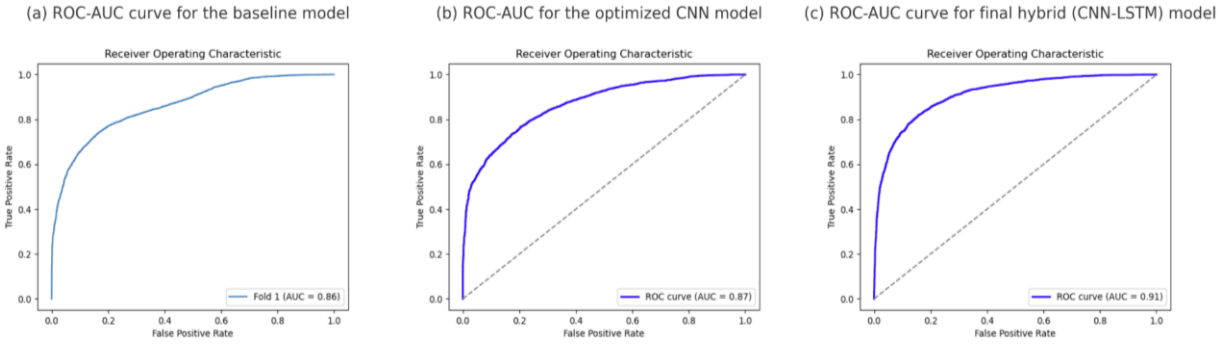
The final evaluation of the two LSTM-integrated training approaches showed consistently high model performance. In the first case (case a), where LSTM was placed immediately after the CNN, the model achieved 94.5% accuracy with 0.5% variability. In the second case (case b), where LSTM was added after the layer providing 35 timesteps, accuracy slightly improved to 95.3% with 0.3% variability. The results indicate that removing the Global Average Pooling layer allowed the LSTM to better utilize temporal information from the 35 timesteps, leading to enhanced performance. To ensure generalization and prevent overfitting, ROC curve was generated, by reserving 15% of the dataset, solely for validation.

The initial evaluation, conducted on a validation dataset derived from the training set, yielded an AUC score of 0.98, indicating potential overfitting. Although this dataset was not used during training, its small size and shared characteristics with the training data may have influenced the results, suggesting that the model memorized patterns rather than generalizing effectively.

To ensure a more objective assessment, a separate independent test-set was introduced. This dataset was sourced externally, processed with Audacity, and structured to include both pollution-related and non-pollution-related events, representing 20% of the total training data size. Using this dataset, the final ROC curve analysis produced an AUC score of 0.91, confirming that the model maintains high classification performance while demonstrating the ability to generalize to unseen data (Figure 6). The final ROC curve also indicates a high True Positive Rate (TPR) at low False Positive Rate (FPR) levels, verifying that the model achieves accurate predictions with minimal classification errors. These results validate the model's robustness and applicability in real-world environmental sound classification tasks.

	a		b	
Class	0	1	0	1
Precision	0.94	0.95	0.93	0.96
Recall	0.95	0.94	0.96	0.94
F1-Score	0.94	0.95	0.95	0.95
Mean Accuracy	**94.5 ± 0.5%**		**95.3 ± 0.3%**	

Table 6: Hybrid Model evaluation (a) LSTM layer right after CNN output (b) LSTM layer after last convolutional layer of the CNN (35 timesteps)



4.5 Micro-service architecture evaluation

The architecture's modular design allows for the incorporation of additional services, expanding its capabilities beyond audio classification. The integration of models for NLP and image event detection enables multi-modal data processing and analysis. This scalability enhances the system's applicability in environmental urban monitoring, allowing real-time analysis of diverse data sources while preserving the modularity and interoperability of the microservices framework.

To evaluate the proposed microservice-based architecture a prototype application was designed and developed, enabling real-time interaction with the deployed models. A web-based interface was

provided, allowing users to upload audio files and receive classification results via the integrated RESTful microservices. The assessment focused on system usability and accessibility, ensuring that the architecture effectively supported real-time environmental sound classification while maintaining flexibility for future extensions.

To measure the usability of the system, a small-scale System Usability Scale (SUS) test was performed, with a sample of 10 people with diverse technical backgrounds (journalists, engineers, communication experts, audio experts) [33]. The obtained average SUS score of 81,88 is well above the average usability threshold of 68 and indicates a positive user experience, confirming that the architecture

provides efficient, accessible, and intuitive interaction with the classification models. It is noteworthy to say that one of the key positive aspects noted was the ability to access the service directly via mobile devices, reinforcing its practicality in real-world scenarios. These results validate the potential of a microservices-based approach in AI-driven environmental monitoring, demonstrating its ability to provide scalable and modular processing capabilities while maintaining a user-friendly interface.

Additionally, the proposed approach has been tested in real-world conditions during three pilot deployments of the SOCIO-BEE project, which aims to engage citizens in environmental monitoring. In these pilots, participants used mobile devices to capture urban soundscapes in areas of interest. They followed the suggested process of uploading audio samples, receiving classification results, and contextualizing the audio with semantic descriptions. These semantic annotations were combined with real-time air quality sensor measurements, enabling a richer interpretation of environmental conditions. This use case highlights the potential of the proposed architecture to support citizen science, pollution awareness, and data enrichment in sustainable urban planning initiatives.

5. Discussion

The selected architecture for deployment is a CNN-LSTM model, where the LSTM layer is placed immediately after the final convolutional layer. This configuration allows the LSTM to fully leverage temporal dependencies extracted by the CNN while preserving the maximum timesteps available from the last convolutional layer. Additionally, data shuffling was applied per fold, increasing training sample diversity and further improving model generalization. The final evaluation was conducted using a 5-fold cross-validation strategy. A comparison between the ROC-AUC curves of the baseline model and the final optimized model clearly demonstrates the superiority of the latter. The increase in AUC from 0.86 to 0.91 validates the improvements in both architecture and training methodology, confirming enhanced classification performance and reliability on independent data. This makes the final model more suitable for real-world applications.

Beyond model performance, the study highlights the importance of architectural design in AI-driven environmental monitoring systems. The integration of RESTful microservices and Docker-based containerization allowed each model—audio classification, NLP, and image event identification—to operate independently while remaining accessible through a unified system. This modular approach ensures scalability and adaptability, making it possible to integrate additional models for multi-modal data processing, without significant architectural modifications.

While the core CNN-LSTM model architecture constitutes an extension of prior work, the primary contribution of this study lies in the development of an end-to-end, audio-focused AI infrastructure. The system enables real-time environmental sound classification through a modular, REST-based architecture that supports deployment as lightweight services, accessible via mobile or web interfaces. This allows for direct inference from raw audio waveforms, without requiring pre-segmented inputs, and facilitates integration with multi-modal data sources such as images or text metadata. The combination of raw waveform ingestion, cloud-based scalability, and mobile compatibility makes the proposed architecture particularly suitable for audio-centric applications in environmental monitoring. Future advancements could focus on container orchestration (e.g., use of Kubernetes for automated service scaling) and task distribution mechanisms (e.g., Celery task queue for asynchronous processing), to optimize resource allocation and improve responsiveness in real-time applications, addressing the growing demands of AI-driven audiovisual environmental monitoring.

References

[1] M. E. Stamatiadou, V. Nikolaos, V. Lazaros, and C. Dimoulas, "Identifying Sources of Air Pollution from Their Audio Footprints," in *Hellenic Institute of Acoustics*, Thessaloniki, Oct. 2022.

[2] A. Emvoliadis, N. Vryzas, M.-E. Stamatiadou, L. Vrysis, and C. Dimoulas, "Multimodal Environmental Sensing Using AI & IoT Solutions: A Cognitive Sound Analysis Perspective," *Sensors*, vol. 24, no. 9, p. 2755, 2024.

[3] J. Hegewald *et al.*, "Traffic noise and mental health: a systematic review and meta-analysis," *Int. J. Environ. Res. Public. Health*, vol. 17, no. 17, p. 6175, 2020.

[4] T. Münzel, M. Sørensen, and A. Daiber, "Transportation noise pollution and cardiovascular disease," *Nat. Rev. Cardiol.*, vol. 18, no. 9, pp. 619–636, 2021.

[5] T. Münzel, T. Gori, W. Babisch, and M. Basner, "Cardiovascular effects of environmental noise exposure," *Eur. Heart J.*, vol. 35, no. 13, pp. 829–836, 2014.

[6] J. Banks and D. Fink, "Noise as a Public Health Hazard," *The Hearing Journal*, vol. 75, no. 5. LWW, p. 6, 2022.

[7] A. Longo, M. Zappatore, M. Bochicchio, and S. B. Navathe, "Crowd-sourced data collection for urban monitoring via mobile sensors," *ACM Trans. Internet Technol. TOIT*, vol. 18, no. 1, pp. 1–21, 2017.

[8] B. Peng, K. I.-K. Wang, and W. H. Abdulla, "An Integrated Hierarchical Wireless Acoustic Sensor Network and Optimized Deep Learning Model for Scalable Urban Sound and Environmental Monitoring," *Appl. Sci.*, vol. 15, no. 4, p. 2196, 2025.

[9] B. İşler, "Urban Sound Recognition in Smart Cities Using an IoT–Fog Computing Framework and Deep Learning Models: A Performance Comparison," *Appl. Sci.*, vol. 15, no. 3, p. 1201, 2025.

[10] M. E. Stamatiadou, I. Thoidis, N. Vryzas, L. Vrysis, and C. Dimoulas, "Semantic crowdsourcing of soundscapes heritage: a mojo model for data-driven storytelling," *Sustainability*, vol. 13, no. 5, p. 2714, 2021.

[11] M. E. Stamatiadou, N. Vryzas, L. Vrysis, T. Saridou, and C. Dimoulas, "A citizen science approach to support joint air quality and noise monitoring in urban areas," in *Audio Engineering Society Convention 152*, 2022.

[12] M. E. Stamatiadou, N. Vryzas, L. Vrysis, and C. Dimoulas, "Beemate the game: A hunting treasure serious game for raising awareness and audience engagement in air pollution monitoring," in *2023 8th International Conference on Smart and Sustainable Technologies (SpliTech)*, IEEE, 2023, pp. 1–4.

[13] Demir, F., Abdullah, D. A., & Sengur, A. (2020). A new deep CNN model for environmental sound classification. *IEEE Access*, *8*, 66529-66537.

[14] K. Zaman, M. Sah, C. Direkoglu, and M. Unoki, "A survey of audio classification using deep learning," *IEEE Access*, vol. 11, pp. 106620–106649, 2023.

[15] Z. Mushtaq, S. F. Su, and Q. V. Tran, "Spectral images based environmental sound classification using CNN with meaningful data augmentation," *Appl. Acoust.*, vol. 172, p. 107581, 2021.

[16] S. K. Shah, Z. Tariq, and Y. Lee, "Iot based urban noise monitoring in deep learning using historical reports," in *2019 IEEE International Conference on Big Data (Big Data)*, IEEE, 2019, pp. 4179–4184.

[17] S. Deshmukh *et al.*, "Audio Entailment: Assessing Deductive Reasoning for Audio Understanding." 2024. [Online]. Available: https://arxiv.org/abs/2407.18062

[18] O. O. Abayomi-Alli, R. Damaševičius, A. Qazi, M. Adedoyin-Olowe, and S. Misra, "Data Augmentation and Deep Learning Methods in Sound Classification: A Systematic Review," *Electronics*, vol. 11, no. 22, 2022, doi: 10.3390/electronics11223795.

[19] B. Peng, W. H. Abdulla, I. Kevin, and K. Wang, "Urban noise monitoring using edge computing with CNN-LSTM on Jetson Nano," in *2023 Asia Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, IEEE, 2023, pp. 2244–2250.

[20] M. G. Ragab, S. J. Abdulkadir, N. Aziz, H. Alhussian, A. Bala, and A. Alqushaibi, "An ensemble one dimensional convolutional neural network with Bayesian optimization for

environmental sound classification," *Appl. Sci.*, vol. 11, no. 10, p. 4660, 2021.

[21] M. Agarwal, K. S. Gill, S. Chattopadhyay, and M. Singh, "Classification of Urban Sound using Sequential Convolutional Neural Network (CNN) Model and its Visualisation," in *2024 IEEE International Conference on Information Technology, Electronics and Intelligent Communication Systems (ICITEICS)*, IEEE, 2024, pp. 1–5.

[22] M. Agarwal, K. S. Gill, P. Aggarwal, R. S. Rawat, and G. Sunil, "Urban Sound Classification using VGG19 Convolutional Neural Network (CNN) Model and its Visualisation," in *2024 4th International Conference on Innovative Practices in Technology and Management (ICIPTM)*, IEEE, 2024, pp. 1–5.

[23] J. Lewis and M. Fowler, "Microservices: a definition of this new architectural term," *MartinFowler Com*, vol. 25, no. 14–26, p. 12, 2014.

[24] D. Merkel and others, "Docker: lightweight linux containers for consistent development and deployment," *Linux J*, vol. 239, no. 2, p. 2, 2014.

[25] E. Wolff, *Microservices: flexible software architecture*. Addison-Wesley Professional, 2016.

[26] Y.-B. Lin, C.-C. Cheng, and S.-C. Chiu, "MusicTalk: A Microservice Approach for Musical Instrument Recognition," *IEEE Open J. Comput. Soc.*, vol. 5, no. 01, pp. 612–623, Jan. 2024, doi: 10.1109/OJCS.2024.3476416.

[27] K. J. Piczak, "ESC: Dataset for environmental sound classification," in *Proc. 23rd ACM Int. Conf. Multimedia*, Brisbane, Australia, Oct. 2015, pp. 1015–1018.

[28] J. Salamon, C. Jacoby, and J. P. Bello, "A dataset and taxonomy for urban sound research," in *Proc. 22nd ACM Int. Conf. Multimedia*, Orlando, FL, USA, Nov. 2014, pp. 1041–1044.

[29] M.-O. Pahl and M. Loipfinger, "Machine learning as a reusable microservice," in *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, 2018, pp. 1–7. doi: 10.1109/NOMS.2018.8406165.

[30] L. Vrysis, I. Thoidis, C. Dimoulas, and G. Papanikolaou, "Experimenting with 1D CNN architectures for generic audio classification," in *Proc. Audio Eng. Soc. (AES) Conv. 148*, May 2020.

[31] S. Widodo, H. Brawijaya, and S. Samudi, "Stratified K-fold cross validation optimization on machine learning for prediction," *Sink. J. Dan Penelit. Tek. Inform.*, vol. 6, no. 4, pp. 2407–2414, 2022.

[32] J. Brownlee, "A gentle introduction to k-fold cross-validation," *Mach. Learn. Mastery*, vol. 2019, 2018.

[33] J. Brooke and others, "SUS-A quick and dirty usability scale," *Usability Eval. Ind.*, vol. 189, no. 194, pp. 4–7, 1996.

Audio Engineering Society

Conference Paper 14

Presented at the AES International Conference on
Machine Learning and Artificial Intelligence for Audio
2025 September 8-10, London, UK

This paper was peer-reviewed as a complete manuscript for presentation at this conference. This paper is available in the AES E-Library (http://www.aes.org/e-lib), all rights reserved. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Journal of the Audio Engineering Society.

Challenges in predicting the lyric intelligibility of musical segments for older individuals with hearing loss

ABSTRACT

Difficulty in understanding lyrics is a major barrier to enjoying music for people with hearing loss. Novel production techniques using machine learning could aid lyric understanding, but objective metrics for training are needed, and these must be informed by the experiences of the target population. Currently, there are no data on the lyric-recall ability of older individuals with hearing loss. Thirteen older participants with mostly mild-sloping hearing loss listened to and recalled 100 vocal segments of popular music that varied in genre, duration and word count. In each trial, participants heard a randomly chosen segment twice with a 5-s interstimulus interval. Reproduction was over headphones at 65 dB(A) using individualised frequency-dependent nonlinear gain to compensate for hearing loss. The proportion of words recalled correctly varied greatly across samples, from 0-100%, and varied as a function of genre, similar to past studies with different populations. Individual intelligibility across segments was correlated with age; segment intelligibility across individuals, however, was not correlated with word count or rate. An objective metric for lyric intelligibility for listeners with hearing loss was developed based on Whisper automatic speech recognition model and individualised hearing-loss simulation. At best, this metric was only able to account for 56% of the variance in listening test results and often underpredicting intelligibility. Improving sung-lyric intelligibility is a different challenge from spoken-speech enhancement for those with hearing loss. Not only does the enhancement need to be considered within the overall enjoyment of the music, but the variation in results – as seen in the current study – reflects variations in genre, orchestration and vocal quality in sung music.

1 Introduction

The joy of music listening can be a key factor for quality of life throughout the lifespan, including older age [1]. For many listeners, lyrics are a key feature in music, with research demonstrating the importance of these for affective experiences [2]-[6]. Critically, people with hearing loss report difficulties with understanding lyrics [7]-[8], introducing a major barrier to the enjoyment of music.

There are numerous factors affecting lyric intelligibility. In a survey of music listeners, Fine and Ginsborg [9] found singer-related (e.g., enunciation, vibrato), music-related (e.g., accompaniment level/ density, mixing), word-related (e.g., formulaic words), listener-related (e.g., attention, hearing loss), and environmental (e.g., audio-visual, reverberation, reproduction) factors can all affect how well the lyrics are understood. While several studies have looked at the differences between sung and spoken intelligibility, only two studies have looked at the intelligibility of sung lyrics in their original context. Condit-Schultz and Huron [10] asked thirty university music and speech-hearing students (ages not given), who passed a normal-hearing screening questionnaire, to repeat back the lyrics of recorded vocal music segments of differing genres. The results showed a strong genre-dependent effect, with their classical genre excerpts being the least intelligible. In a secondary experiment, a subset of segments was repeated; performance (correct recall) increased on second and slightly on third presentation, but not for additional repeats. Self-reported genre familiarity was not correlated with performance. A later study

by Ibrahim et al. [11] included 17 Singaporean participants aged 21-41, who were mostly university students, reported no hearing problems and "spoke some English as a second language" (p. 2). Those participants heard each of 100 recorded segments twice. That study [11] was intended to automatically classify the intelligibility of vocal music for potential use in the teaching of English, hence there was no analysis of the behavioural results, which were considered ground truth to evaluate their classifier.

One of the aims of the Cadenza project [12] is to run machine-learning challenges to optimise technology to improve lyric understanding without sacrificing overall enjoyment of the music. To achieve this optimisation, developers need lyric-intelligibility metrics that correlate with the perception of the target population. Currently, there are no data on the ability of older individuals with hearing loss to hear and recall lyrics.

In the current study, we aimed to create more inclusive future objective metrics by applying the same behavioural methodology of Ibrahim et al. to an older population with hearing loss. To ensure that the results were not simply a case of audibility, the stimuli were amplified to compensate for each listener's hearing loss. We then assessed how well a state-of-art machine-learning metric predicts lyric intelligibility for this population, and what is needed to create future lyric-intelligibility metrics that can better predict across listeners.

2 Methods

2.1 Participants

Thirteen adults, aged 58-79 (median 73) years, were recruited from the participant pool of Hearing Sciences – Scottish Section who were all native speakers of British English. Left and right pure-tone thresholds were measured using the modified Hughson-Westlake procedure [13] and are shown in Figure 1. Better-ear four-frequency pure-tone averages (BE4FA), a standard metric of objective hearing ability, were calculated from the left-right minimum average of thresholds at 0.5, 1, 2 and 4 kHz. BE4FAs ranged from 17.5-41.25 (median 25) dB HL. Age and BE4FA were not significantly correlated (Pearson product-moment correlation r = 0.02). All participants provided written informed consent prior to taking part and were paid for their participation.

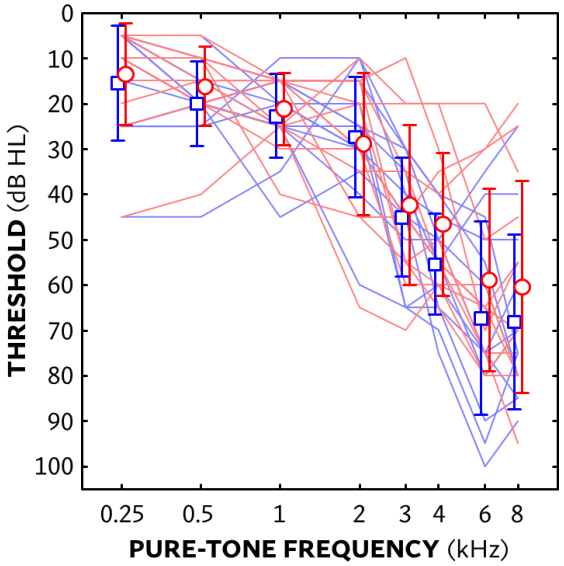


Figure 1. Pure-tone thresholds as a function of frequency for all participants left (blue lines) and right (red lines) ears as well as means (squares and circles for left and right ears, respectively) and standard deviations (error bars) at each frequency.

2.2 Stimuli

Stimuli consisted of 94 segments of stereo vocal music – 2 segments each from 47 pieces of music – that varied in genre, duration and word count. These segments, along with six other segments that were excluded as they were assessed as unpleasant in piloting, were used as they were made available from a previous lyric intelligibility study [11]. The broadly very defined genres were classical (20 segments, comprised of single-voice, mostly light opera excerpts), folk (20), jazz (20), pop (14) and rhythm & blues (20). Segment durations were 1.65-16 s (median 6 s); some segments began and/or ended abruptly mid phrase. The number of scored words ranged from 3-19 (median 8), and word rates ranged from 0.31-3.64 words/s (median 1.5 words/s). Fidelity varied across segments despite all being originally from commercial recordings, and some segments began or ended abruptly mid phrase. Despite these shortcomings, the corpus was available and allowed direct comparison with a previous dataset.

Segments were all RMS normalised (without and frequency weighting) and processed using a simulation of hearing-aid non-linear amplification. Each channel (L/R) of each segment was first split into octave bands using cascaded Butterworth filters with centre frequencies of 250, 500, 1000, 2000, 4000

and 8000 Hz. Gains for each participants' ears were based on an A-weighted 65 dB presentation level using an implementation of the CAMFIT nonlinear prescription [14]-[15]. These individualised frequency-dependent gains were then applied to the filtered-band L/R signals. Compression ratios for each participants' ears were determined by the difference in prescribed outputs at 55- and 85-dB input [16] with an upper ratio limit of 6:1 imposed to avoid extreme values. Compression attack and release times (11-14 and 80-100 ms, respectively, across centre frequencies) were based on average values measured using IEC 60118 procedures in conventional nonlinear hearing aids [17]. Individualised compression was then applied to the filtered-band L/R signals. The filtered, amplified and compressed stimuli were then summed. Stimuli were presented through an external soundcard (RME Babyface) to circumaural headphones (AKG702) in a sound-dampened booth. Participant responses were monitored and recorded using a lapel microphone.

2.3 Procedure

In each trial, participants heard a segment that was presented twice with a 5-s interstimulus interval over headphones at an A-weighted level of 65 dB plus individualised frequency-dependent gain (CAMFIT-nonlinear). Participants could begin recalling lyrics after the first presentation. Segments were presented in a randomised order for each participant. Participant responses were recorded as well as marked in real time, then checked again after the experiment by the authors for marking accuracy. Lyrics for each segment were initially provided by [11] and then validated by the authors.

2.4 Prediction

Each music segment used in the study was processed independently for the left and right channels for each participant. First, gain and compression were applied as described in the Methods above. The amplified signals were then passed through a Python implementation of a gammatone-filterbank hearing-loss model [18]-[19] that simulates decreased audibility, dynamic range, and temporal and frequency resolution using solely the participants' pure-tone thresholds as input. Each processed signal was then transcribed using an automatic speech recognition system trained on a large multilingual dataset in various contexts including background music (Whisper; [20]). The Whisper model was calibrated to produce deterministic results (i.e., avoid hallucinations) by setting its temperature parameter to zero.

3. Results

3.1 Behavioural results

Performance was measured as the proportion of words recalled correctly [P(C)]. Performance varied greatly across music segments, from 0-100%, and varied as a function of genre, as shown in Figure 2. There was a significant effect of genre on individual mean performance [repeated-measure ANOVA $F_{(4,24.36)} = 142.73$; $p << 0.001$; $\eta^2 = 0.92$]. In comparison to previous studies [10]-[11], the overall P(C) was significantly lower than either of the previous studies: 0.65 vs. 0.72 in both studies [Student's $t_{(12)} = 3.07$; $p = 0.0098$].

By genre, the current mean results were significantly less than both previous studies for the pop and R&B genres [$t_{(12)} = 3.15$-6.19; all $p_{adj} < 0.01$], and also significantly less than [11] for the classical genre, and [10] for the jazz genre (n.b., which was near perfect performance). The mean P(C) for the jazz genre was significantly greater than it was in [11] using the same segments, but this was partially due to a particular stimulus (a segment of Nat King Cole singing "Rambling Rose") for which the previous study's participants recalled zero correct words, whereas the current participants on average were very accurate [P(C) = 0.94]. Considering that previous study [11] used the same stimuli with a different population, the results were reasonably well correlated ($r = 0.72$; Spearman's $\rho = 0.70$; $p < 0.001$), although there were several more segments like the one noted above with large discrepancies (e.g., 12 segments with interstudy differences > 0.33), and a majority of segments (70%) were more intelligible [ΔP(C) > 0] in the previous study. Individual mean results in the current study were significantly correlated with age [$r = 0.83$; $p < 0.001$ ($\rho = 0.73$; $p = 0.0045$)] but not BE4FA ($r = -0.48$; $p = 0.099$), although this may be due to the small n (i.e., underpowered).

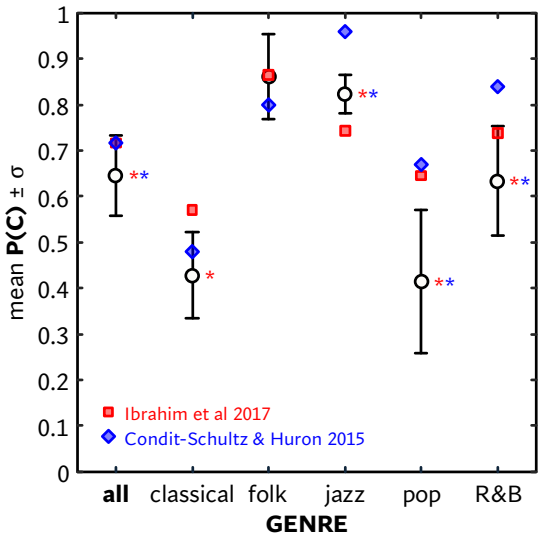


Figure 2. Mean proportion correct [P(C)] of lyrics recognised as a function of genre as compared to results for the same samples from Ibrahim et al. [11] (red squares) and different samples categorized into the same genres from Condit-Schultz and Huron [10] (blue diamonds). Error bars show ±1 standard deviation. Asterisks indicate significant differences from past studies. Note that the "pop" genre mean is based on 14 samples, all other genre means based on 20 samples.

Performance [P(C)] was generally not correlated with segment attributes: duration, word count nor word rate across all segments. However, as shown in Figure 3, P(C) was correlated with word rate within the pop genre. As highlighted by the filled symbols in Figure 3, there were some large P(C) differences between excerpts of the same song that had similar attributes. For example, there was a 59% difference in mean performance on the two "classical"-genre segments from Aaron Copland's The Tender Land (filled red circles). Both segments feature the same singer, but there are audible music-related differences: the highly intelligible segment has a sustained string background, whereas the much less intelligible segment has a more pronounced accompaniment that changes with each syllable, masking the consonants. Another example is the disparity between two "jazz"-genre segments of "This Masquerade Is Over" sung by Marvin Gaye (filled green downward triangles). The disparity in performance can be attributed to multiple factors: singer-related changes in enunciation, as well as word-related differences in between these samples that are both singer-related changes in enunciation and word-related changes in prevalence ("and so is love my darling" vs. "learn to laugh like Pagliacci with tears" for the more and less intelligible segments, respectively).

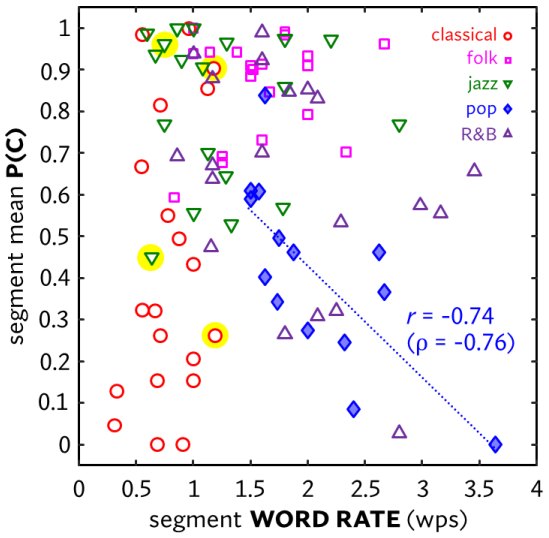


Figure 3. Mean proportion correct as a function of word rate for each segment with symbols indicating genre. Dotted (blue) line indicates linear regression for the pop genre (half-filled blue diamonds). The large differences between segments from the same recording described in the text are highlighted with yellow circles.

3.2 Prediction results

From the Whisper transcription, the number of correctly recognised words was computed without applying penalties for substitutions, insertions or deletions. The better score (proportion of correctly transcribed words) from the left and right channels was then used as the predicted performance for that participant. The use of the better-ear score emulated better-ear listening, well demonstrated in speech intelligibility but unknown for lyric intelligibility, as well as optimised the correlation with the listening test results.

Model results are shown in Figure 4. Predicted mean scores for each segment were well correlated with actual scores, both across and within genres except for the pop genre, for which intelligibility decreased with increasing word rate. The model, however, underpredicted performance for the vast majority (88%) of the segments; the mean P(C) difference between actual and predicted mean segment scores was +0.27. By genre, the underprediction was greatest for the "jazz" genre (mean difference = +0.35) and least for the "classical" genre (mean difference = +0.19). Actual-predicted score differences were not correlated with segment

attributes (i.e., duration, word count or rate; r^2 = 0.0061-0.015). By individual participant, predicted scores, averaged across segments, were not correlated with actual scores (r^2 = 0.007). Actual-predicted differences were negatively correlated with BE4FA (ρ = -0.60; p = 0.032), indicating that the prediction was generally worse for those with milder losses.

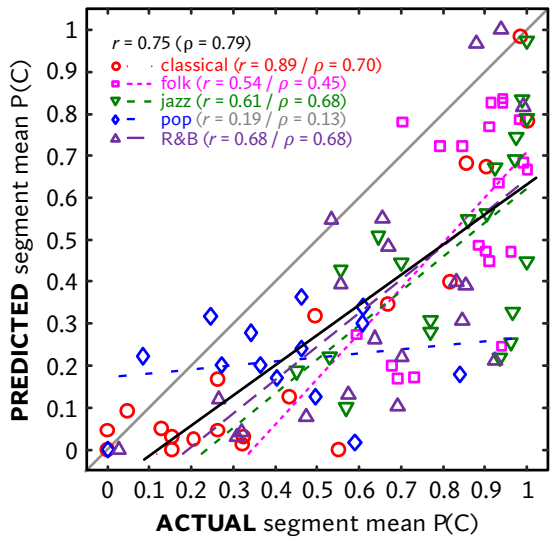


Figure 4. Predicted segment mean proportions correct [P(C)] as a function of actual segment mean P(C) for each segment. Genres are indicated by different symbols and colours. Linear regressions are shown for all genres; the solid black line showing the regression for all segments. The axes diagonal (grey line) is provided to better visualise the disparities from perfect prediction.

To better understand the potential sources of misprediction, we repeated the Whisper prediction (1) with only the individual hearing-loss simulation (i.e., without the nonlinear gain stage), and (2) without the hearing-loss simulation (i.e., without any processing). Overall results for these predictions along with the behavioural and fully processed predictions are shown in Table 1. Removing the gain stage, which should have partially compensated for reduced audibility introduced in the following hearing-loss simulation stage, only decreased the predicted mean performance by 1%. Removing the hearing-loss simulation processing – that is, the Whisper prediction for a listener without hearing loss – predicted poorer performance than what was achieved by the 13 individuals with hearing loss in the current study.

Table 1. Mean results showing changes in mean proportion correct P(C) and Pearson product-moment correlation coefficients (r^2) between actual and predicted mean scores for each segment.

data	mean P(C)	r^2
actual	0.65	--
predicted (gain + HL)	0.38	0.56
predicted (HL only)	0.37	0.64
predicted (no processing)	0.60	0.52

4 Discussion

Predicting the ability to recall lyrics for those with hearing loss poses myriad difficulties. In the current study, there was an absence of any general relationship between segment features and lyric intelligibility. This disparity is not limited to listeners with hearing loss; [10] only found a correlation between performance and log-duration of their excerpts; no other features correlated with performance in their study nor in [11]. Curiously, the one genre – "pop" – that showed any relationship to attributes (segment word rate; Figure 3) in the current study was also the only genre for which the Whisper-based prediction was poor. In [11] and a follow-up study [21], broad categories of (actual) intelligibility scores were used to inform and initially assess classifiers based on acoustic attributes. With the current results, such broad categories would be problematic due to the scale of underprediction [e.g., 38% of the segments had an actual P(C) > 0.5 and predicted P(C) < 0.5 whereas only 31% had an actual and predicted P(C) < 0.5]. In machine-learning endeavours, we want to include rather than exclude – to generalise our predictive capacity for real-world benefit. The current study, however, highlights the inherent difficulty in prediction when numerous factors (i.e., singer-, music-, word- and environment-related [9]) are involved. Homogeneity in sampling, however, is a challenge unto itself. Focusing on a single genre may not be enough to disentangle these factors, as here lyric intelligibility varied within song as well as within genre, and many features are shared across genres [22].

Another potential factor in this underprediction may be the hearing loss simulator, especially considering that gain compensation was applied to the signals to partially account for loss in audibility. One potential explanation is that Whisper ASR is unlikely to have been trained on audio degraded with a hearing loss simulator, and consequently some of the

underprediction could be an issue of out-of-domain generalisation. Another potential explanation is that Whisper's predictions may not consider the reduced sensation level due to hearing loss; that is, it normalises the input. This normalisation would then at least partially explain the lack of change due to applying nonlinear gain, but then the difference between predicted results with and without the HL simulator (see Table 1) would only be the result of Yet another consideration is that current hearing-loss models only use pure-tone thresholds as parameters, hence can only be expected to accurately simulate elevated thresholds, not supra-threshold abilities [23]. To better understand the role of the hearing-loss simulator in lyric intelligibility prediction, future work could present hearing-loss simulator processed stimuli to young normal-hearing listeners to compare performance to those with hearing loss. Another important facet for prediction is taking into consideration the age of the listener, although the negative correlation between age and performance may just be a result of the difficulty of the task (e.g., the short-term memory resources involved in having to recall sung words from a musical segment presented, albeit twice, out of context).

The current study was limited by having a small number of participants that were all older with varying but generally milder forms of hearing loss. There were no age-matched controls per se; however, the comparison with previous studies using self-reported normal-hearing students (Figure 2) shows that variance by participant group – an overall mean difference of 7% – is far less than the differences due to individual segments or genre. Another distinction of the current study was the older age of the participants. Within the older population, age was correlated with overall performance. The potential role of additional age-related factors in lyric intelligibility as separate from the measurement of intelligibility (i.e., the short-term memory demands of recall) warrant further study. The correlation with word count (Figure 3) and lack of prediction (Figure 4) for the "pop" genre, consisting of more recent recorded material also warrants further exploration as to the interaction between performance attributes (e.g., lyrical word rate), familiarity – both broadly (style) and specifically (performer) – and listener age and cognitive capacity.

5 Conclusion

The current study measured lyric intelligibility in a small sample of adults with hearing loss. An objective metric for lyric intelligibility for people with hearing loss was developed that combined hearing-loss simulation with Whisper ASR. The metric was developed for the Cadenza challenges, to allow researchers to train machine learning systems to improve the intelligibility of music lyrics. Such improvements could provide novel solutions for some of the problems faced by those with hearing loss when listening to sung music.

The objective metric did correlate with listening test results, accounting for 56% of the variability in segment intelligibility, but there was significant underprediction. To overcome the challenges in predicting lyric intelligibility, focus may need to shift from the intelligibility as defined as words correctly recalled, to the subjective impression of intelligibility – a song's lyrical clarity.

Acknowledgements

This work was supported by funding from the Engineering and Physical Sciences Research Council (EP/W019434/1) and Medical Research Council (MR/X003620/1).

References

[1] Creech A, Hallam S, McQueen H, Varvarigou M (2013) The power of music in the lives of older adults. *Res Studies Mus Ed* 35: 87–102. DOI: 10.1177/1321103X13478862.

[2] Ali SO, Peynircioğlu ZF (2006) Songs and emotions: Are lyrics and melodies equal partners? *Psych Mus* 34(4): 511–534. DOI: 10.1177/0305735606067168.

[3] Barradas GT, Sakka LS (2022) When words matter: A cross-cultural perspective on lyrics and their relationship to musical emotions. *Psych Mus* 50(2): 650–669. DOI: 10.1177/03057356211013390.

[4] Brattico E, Alluri V, Bogert B, Jacobsen T, Vartiainen N, Nieminen S, Tervaniemi MA (2011) Functional MRI study of happy and sad emotions in music with and without lyrics. *Frontiers Psych* 2: 308. DOI: 10.3389/fpsyg.2011.00308.

[5] Gabrielsson A (2011) Strong experiences with music: Music is much more than just music. Oxford, UK: Oxford University Press.

[6] Lamont A (2011). University students' strong experiences of music: Pleasure, engagement, and meaning. *Musicae Scientiae* 15(2): 229–249. DOI: 10.1177/1029864911403368

[7] Madsen SMK, Moore BCJ (2014) Music and hearing aids. *Trends Hear* 18. DOI: 10.1177/2331216514558271.

[8] Greasley A, Crook H, Fulford R (2020) Music listening and hearing aids: Perspectives from audiologists and their patients. *Int J Audiol* 59(9): 694-706. DOI: 10.1080/14992027.2020.1762126.

[9] Fine PA, Ginsborg J (2014) Making myself understood: perceived factors affecting the intelligibility of sung text. *Frontiers Psych* 5: 809. DOI: 10.3389/fpsyg.2014.00809.

[10] Condit-Schultz N, Huron D (2015) Catching the lyrics: Intelligibility in twelve song genres. *Music Perc* 32(5): 470-483.

[11] Ibrahim KM, Grunberg D, Agres K, Gupta C, Wang Y (2017) Intelligibility of sung lyrics: a pilot study. *Proc 18th ISMIR*, pp. 686-693.

[12] https://cadenzachallenge.org/.

[13] British Society of Audiology (2018) *Pure-tone air-conduction and bone-conduction threshold audiometry with and without masking*. Retrieved from https://www.thebsa.org.uk/wp-content/uploads/2023/10/OD104-32-Recommended-Procedure-Pure-Tone-Audiometry-August-2018-FINAL-1.pdf.

[14] Moore BCJ, Alcántara JI, Stone MA, Glasberg BR (1999a) Use of a loudness model for hearing aid fitting II. Hearing aids with multi-channel compression. *Br J Audiol* 33(3): 157-170. DOI: 10.3109/03005369909090095.

[15] Moore BCJ, Glasberg BR, Stone MA (1999b) Use of a loudness model for hearing aid fitting III. A general method for deriving initial fittings for hearing aids with multi-channel compression. *Br J Audiol* 33(4): 241-258. DOI: 10.3109/03005369909090105.

[16] International Electrotechnical Commission (IEC) (2022) Electroacoustics - Hearing aids - Part 0: Measurement of the performance characteristics of hearing aids.

[17] Whitmer WM, Akeroyd MA (2011) Level discrimination of speech sounds by hearing-impaired individuals with and without hearing amplification. *Ear Hear* 32(3): 391-398. DOI: 10.1097/AUD.0b013e318202b620.

[18] Nejime Y, Moore BCJ (1997) Simulation of the effect of threshold elevation and loudness recruitment combined with reduced frequency selectivity on the intelligibility of speech in noise J Acoust Soc Am 102(1): 603–615.

[19] https://github.com/clarity challenge/clarity

[20] Radford A, Kim JW, Xu T, Brockman G, McLeavey C, Sutskever I (2023) Robust speech recognition via large-scale weak supervision. *Proc Int Conf Machine Learning*, pp. 28492–28518.

[21] Sharma B, Wang Y (2020) Automatic evaluation of song intelligibility using singing adapted STOI and vocal-specific features. *IEEE/ACM Trans Audio Sp Lang Proc* 28: 319-331. DOI: 10.1109/TASLP.2019.2955253.

[22] Lesimple C, Kuehnel V, Siedenburg K (2024) Hearing aid evaluation for music: Accounting for acoustical variability of music stimuli. *JASA Express Lett* 4(9): 093201. DOI: 10.1121/10.0028397.

[23] Carney L (2018) Supra-threshold hearing and fluctuation profiles: Implications for sensorineural and hidden hearing loss. *J Assoc Res Otolaryngol* 19: 331-352. DOI: 10.1007/s10162-018-0669-5.

Audio Engineering Society

Conference Paper 15

Presented at the AES International Conference on Machine Learning and Artificial Intelligence for Audio
2025 September 8–10, London, UK

This paper was peer-reviewed as a complete manuscript for presentation at this conference. This paper is available in the AES E-Library (http://www.aes.org/e-lib), all rights reserved. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Journal of the Audio Engineering Society.

Unstable Audio: Code Bending Text-to-Music Generation

Nick Collins[1]

[1]*Durham University Music Department*

Correspondence should be addressed to Nick Collins (nick.collins@durham.ac.uk)

ABSTRACT

Text-to-music models provide new audio generational and transformational capability, though their control is not a precise art. Embracing the aesthetics of 'code bending', glitch musicians' approach to software manipulation and the sounds of noise music, the open source Stable Audio 1.0 model is manipulated to enable radical new audio outputs. The domain of historical late 1980s Detroit techno is used as a constrained arena for generation, based on a fixed text prompt. A comparative study is then undertaken of perturbing model parameters for each of the 745 layers of the model . Adding random noise to layer parameters is more likely to lead to interesting results than zeroing (pruning) connections or randomly reconfiguring parameters throughout the model, though all manipulations had some chance of creating interesting outputs. Further potential code bending projects for text-to-music engines are discussed.

1 Introduction

Deep learning has established an immensely productive and active research area for sound synthesis, in neural audio synthesis. For instance, the original WaveNet model [1] has had over 6000 citations, and subsequent developments have included large dataset trained company efforts such as Meta's AudioCraft[1] with open source code and freely downloadable models [2], and multiple audio models available through pipelines in the Hugging Face transformers[2] and diffusers[3] python packages. Such publicly available software is an irresistible target for subversion in artistic projects, and neural audio engines are powerful sound synthesis resources which may be approached with more experimental aesthetics.

The area of code bending [3] is a development which crosses hardware hacking [4] and live coding [5, 6], promoting the deliberate rewriting and perturbation of existing software for novel aesthetic ends. In the area of deep neural networks, this has been called 'network bending' [7] and is applicable to generative models for the audio,visual and joint audiovisual modalities [8]. In Collins et al. [9] open 'brAIn' transplant surgery is explored swapping layers between multiple autoencoder models; Dzwonczyk et al. [10] bend the Stable Diffusion image generation engine to create music video outputs, by using audio features as control parameters to manipulate nodes to render a sequence of multiple images . The corruption of neural audio models through such techniques can lead to new timbres and new control possibilities, empowering new compositions utilising these sounds, and providing means to escape the more obvious close reproductive intent of standard text-

[1]https://github.com/facebookresearch/audiocraft
[2]https://huggingface.co/docs/transformers/en/index
[3]https://huggingface.co/docs/diffusers/index

to-audio models. Such a project steers closer to glitch [11] and noise music [12] aesthetics than conventional computer science oriented deep learning work.

In this paper we explore hacking Stable Audio Open (1.0) which is an open source text-to-music model. It utilises a DiT (Diffusion-Transformer) model on the inner representational (latent) space of an associated audio encoder-decoder. Technical details of Stable Audio 1.0 are discussed in reports from Stability-AI researchers [13, 14]; the model with all source code and weights is available online.[4]. This paper carries out a comparative investigation of manipulating the parameters at the many different layers of the Stable Audio generative model in section 2. We discuss the sound transformations and the potential for further code bending in section 3. Sound examples[5] and open source code[6] for the manipulations and figure generation accompany the paper.

2 Manipulations

Stable Audio 1.0 has around 1.2 billion float parameters, with 745 layers in the core model combining the autoencoder which accesses the lower dimensional latent space where generation can take place, a diffusion-transformer model for longer-structure aware generation, and text embedding module for conditioning generation on text prompts. The number of parameters available at each layer vary but can become three dimensional tensors with millions of parameters (for example, layer 570 labelled *pretransform.model.decoder.layers.1.layers.1.weight_v* has shape *[2048, 1024, 16]*). In section 2.1 we explore adding Gaussian noise of different amplitudes to the set of parameters at particular layers. In section 2.2 a subset of parameters across the entire model are either zeroed out, or set to a random value; this latter manipulation leads to more unstable outputs, including silent output.

2.1 Adding noise to model layers

In order to better quantify the effect of interventions on the many potential hacking points within a large model, an exhaustive search was carried out across all the 745 layers of the Stable Audio model. To each tensor of parameters for a given layer, Gaussian noise was added[7] of amplitude 0.03, 0.3 and 3 (in initial experiments, 0.01 was too little to hear much difference in many cases). Gaussian noise was selected as a baseline stochastic model rather than a uniform distribution, but given the order of magnitude differences here, the type of noise was not particularly critical. A fixed seed and fixed prompt was used so that the baseline generation would always be the same and only the change in weights would affect the output; outputs were all 10 seconds long, at 44.1KHz. The prompt was lifted from the history of electronic dance music, namely Derrick May's description of Detroit Techno: 'George Clinton and Kraftwerk are stuck in an elevator with only a sequencer to keep them company' [15]. This prompt was chosen because of the author's familiarity with the history of electronic dance music in this era and electronic music history more generally [16].

Rendering of three added noise variants for each of 745 layers took 31 hours on a recent MacBook Pro with Apple M4 Pro chip (utilising the Metal Performance Shader GPU). Once obtained, the 10 second outputs from Stable Audio were analyzed with respect to three timbral features (spectral centroid, spectral bandwidth and spectral flatness) extracted with librosa[8] [17] and averaged over the 10 second clips. To provide further context to the text-to-music generation, a small corpus of control cases was also introduced, and analyzed with the same three audio features. The additional audio sources consisted of 10 second extracts from Kraftwerk (15 famous tracks from 1975 to 1981), Parliament Funkadelic (eight tracks from the *Mothership Connection* album released in 1975), and Derrick May (10 well known tracks from 1985-1989, sourced from his compilation *Innovator* (1997)). Further, 10 variations (without weight manipulation and without any set seeds) were also created with Stable Audio 1.0 for each of the five prompts below, including the original prompt above and some related descriptions of the historical style:

- 'Detroit techno circa 1988'
- 'Derrick May electronic dance music'
- 'early techno music'

[4] https://github.com/Stability-AI/stable-audio-tools
[5] https://composerprogrammer.com/unstableaudio.html
[6] https://github.com/sicklincoln/UnstableAudio
[7] Using torch.randn, which gives a normal distribution of mean 0 and standard deviation 1, scaled by the coefficients here.
[8] https://librosa.org/doc/latest/index.html

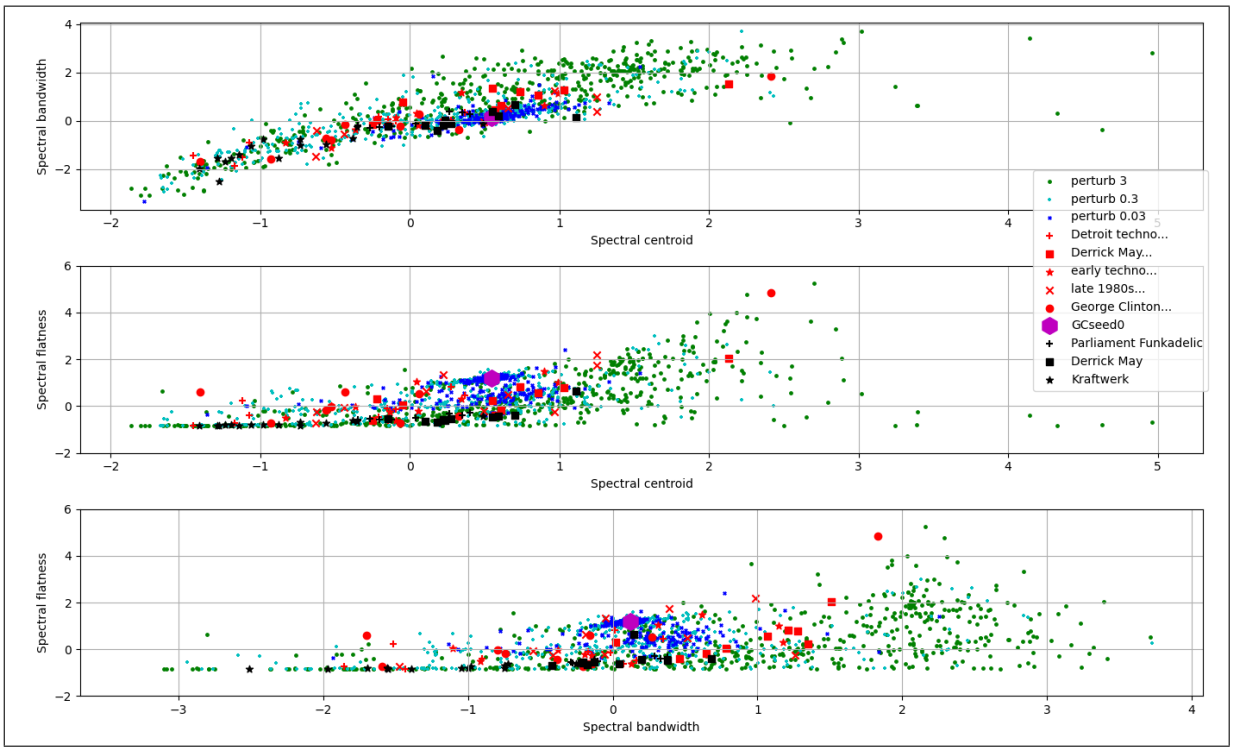


Fig. 1: Three scatter plots showing the interactions of three timbral features, plotted for Stable Audio generations with different prompts, with different layer parameter manipulations, and with ground truth historical audio from Kraftwerk, Parliament Funkadelic and Derrick May.

- 'late 1980s Detroit techno'
- 'George Clinton and Kraftwerk are stuck in an elevator with only a sequencer to keep them company'

Figure 1 plots these different conditions with respect to each other. The original Stable Audio generation based on the 'George Clinton' prompt with seed 0 and no weight variation is indicated as a larger point in purple. Figure 3 extends the analysis to rhythmic features, extracted using the SuperCollider Music Information Retrieval library [18]. The two features in the plot are the mean IOI (Inter-Onset Interval) as found by a percussive onset detector, and the 'metricity' which is a measure of how metrically aligned the beat histogram is in terms of simple integer harmonic or sub-harmonic relationships of peak locations to the largest peak. The historic human-composed music is generally separated especially on the metricity dimension (for convenience the figure also include audio examples of the overall zeroing and noise conditions of section 2.2).

The outputs from the Detroit techno prompts, from a musicological standpoint, are highly unreliable. Not only do they include anachronistic sounds, but they are often not electronic dance music at all but screaming walls of noise. The weight manipulated outputs did lead to interesting sonic variations, including the introduction of voice-like sounds (despite Stable Audio not being trained for vocal reproduction) and 'ambient' transformations from the original more noisy rhythmic material, to spaced out 'elevator music' far from techno's pulsing drive.

The three features used here are only a rough timbral guide to the music and do not include rhythmic or pitch information, or time varying 'spectromorphology' [19]. Nonetheless, they show that the prompt based generations, and the parameter manipulations of the model, swarm around but do not often come close to the original Kraftwerk, p-funk and Derrick May works, and more often diverge into unrelated territory. Human aural analysis confirms this.

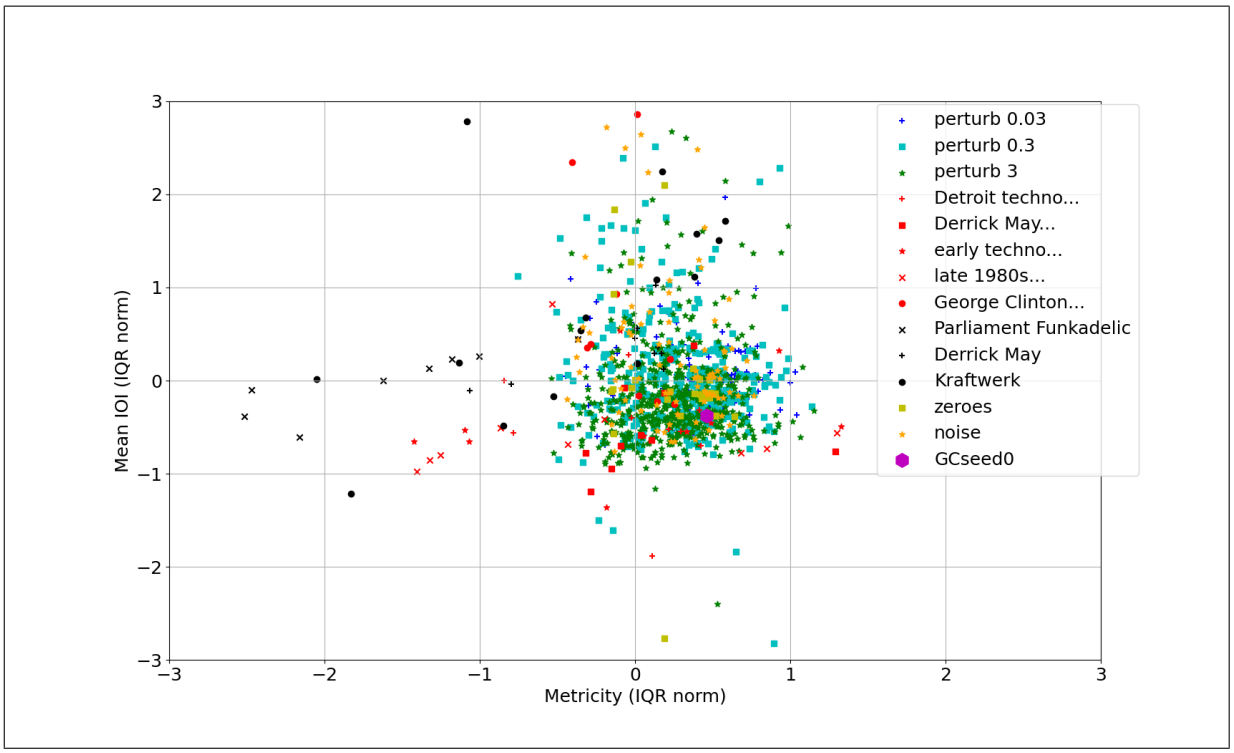


Fig. 2: Plot of two rhythmic features (mean IOI and metricity), plotted for Stable Audio generations with different prompts, with different layer parameter manipulations, and with ground truth historical audio from Kraftwerk, Parliament Funkadelic and Derrick May.

Interventions have different effect based not only on what amplitude of Gaussian noise is added to a layer's parameters, but on the stage of the layer through the model. Figure 3 plots the three features against layer. It is initially striking that a run of layers, counts 380 to 562, do not adjust the output at all, seen in the feature data at a fixed plot level; zooming revealed no finer grained structure, and aural checks showed only the original seed 0 audio result was being returned. However, closer investigation of the model proved that this was not impactful in final rendering because it corresponds to the *pretransform.model.encoder* layers, which are only utilised in audio to audio transformation and not when going from a text prompt direct to output audio (adjusting weights for layer 563, the beginning of the following *pretransform.model.decoder*, shows audible results again). If starting from input audio, it would be interesting to manipulate the encoder weights, but wasn't a task explored herein.

2.2 Zeroing out and randomising a subset of weights

Rather than adding a fixed tensor of perturbations, it is possible to directly change existing parameters, though this can have a more unsettling effect on the model.

Fixing the same prompt and seed as previously, a random set of parameters picked from any layer were set to zero, for ten different conditions: 1, 10, 30, 100, 300, 1000, 3000, 10000, 100000 and 1000000 zeroings. Each experiment was repeated 10 times (so across the 10 conditions, 100 runs of Stable Audio 1.0). The more parameters zeroed out, the more likely a silent output would be, though this only became a prominent issue once going past 1000 zeroes. Many outputs, especially for a smaller number of zeroes, had no perceptible difference to the reference starting point (seed 0, no zeroes, no parameter adjustments). However, a few more radical transformations were achieved for a moderate number of zeroes (300-3000). Since zeroing out parameters is equivalent to 'pruning' connections

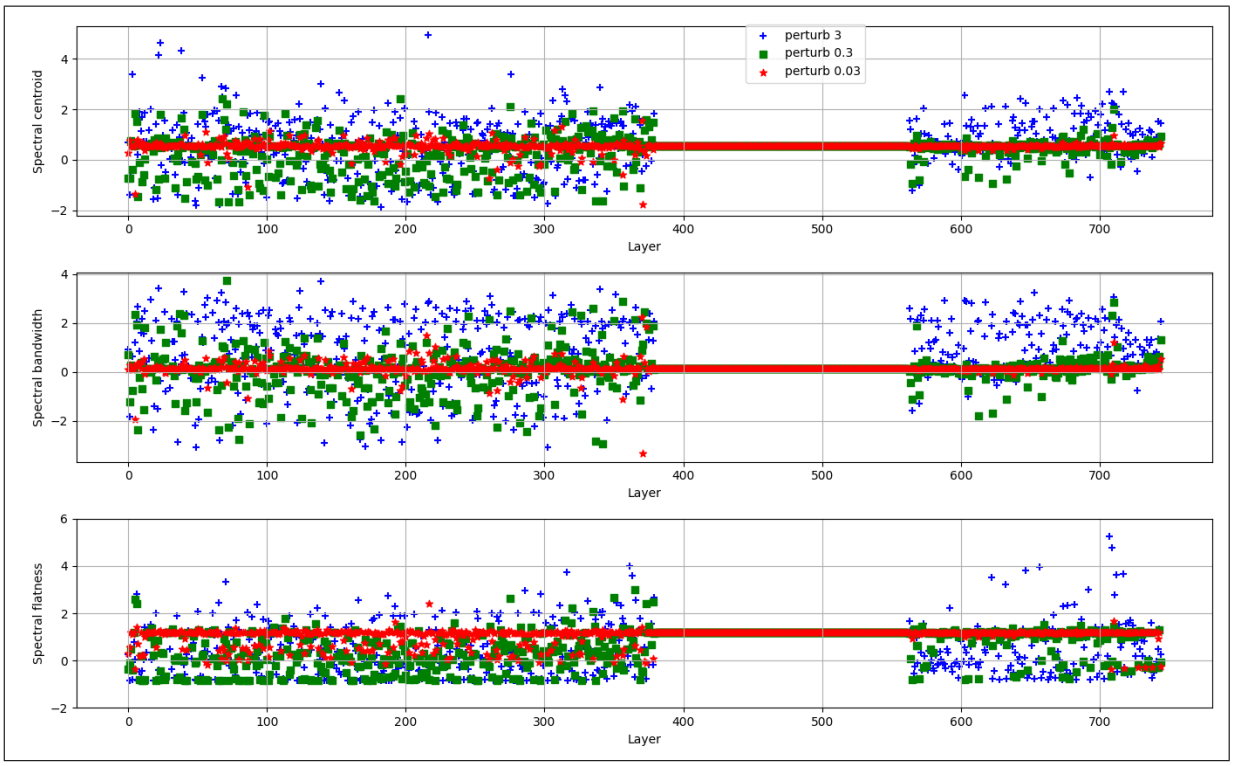


Fig. 3: Three scatter plots showing three timbral features plotted against the layer of network receiving a parameter adjustment.

in a network, the model was only robust to moderate pruning (around three in one million connections, 0.0003%).

We also investigated, again with the same prompt and seed as previously, setting a random set of parameters from any layer to a uniform random number from -1.0 to 1.0, for 1, 10, 30, 100, 300, 1000, 3000, 10000, 100000 and 1000000 parameter substitutions, each experiment repeated 10 times (another 100 total runs of Stable Audio 1.0). For higher numbers of randomised parameters (>=10000), outputs tended to become high frequency repetitions, sounding very coarse and not distinctive; nonetheless, when setting a moderate number of parameters (100, 300, 1000, 3000) it was possible to access new sounds; 1000-3000 zeros were especially good at transforming sound in surprising ways.

3 Discussion

Hacking the Stable Audio 1.0 model does lead to some productive and intriguing sounds. Audio examples are released to accompany this paper; they range from frequency filtering effects, to more radical slurring, timbral substitution, stuttering, and changes of reverberant acoustic. Most outputs were continuous in rhythmic or ambient texture; a few demonstrated isolated hits with silence inbetween. Occasionally, spectral processing artefacts were obvious, but most had a nice sound quality, if sometimes distorting. Unless parameters were entirely zeroed out, often leading to silent output, most parameter changes would allow through a transformed version of the original output. The systematic comparison by layer above only went up to a parameter multiplier of 3; higher values still are perfectly possible, but in qualitative trials more often lead to extreme distortion, or 'frozen' outputs with static frequency partials or constant stridulation.

Musicians are actively exploring text-to-audio generative AI models [20], and more facilities for their customisation, personalisation and sonically impactful code bending warrant additional research. The manipulations investigated here are not undertaken to repro-

duce well worn paths in existing music (one of the limitations of general text-to-audio training founded in specific databases) but to push models outside of the more obvious places in their generative space, and access new sound transformations. With this goal in mind, the project did reveal many unexpected sound outputs that could not be reached in alternative obvious ways with existing computer music algorithms. A more systematic polling of layer by layer weight perturbations led to more interesting results overall than a global manipulation such as substituting an activation function with another; however, given that only limited text prompts have been systematically compared herein, there is much scope to compare manipulations in the case of other prompts and no hard rule can be established at this point.

Future techniques to be explored might include:

Coarse tuning It is possible to further train the baseline Stable Audio 1.0 model on a given audio corpus, via fine tuning of all model weights, or substantially reduced numbers of parameters via techniques like Low Rank Adapters (LORAs) [21, 22, 23, 24]. Rather than fine tune an existing model, we might 'coarse tune', maximising unexpected outputs by training on non-congruent audio, connecting an under-sized parasitical LORA to awkward layer positions, using a deliberately wild cost function etc.

Model cannibalization Varying the experiments of Collins et al. [9], modular blocks such as the encoder, or specific layers, can be substituted from alternative models, shuffled (where there are compatibilities of tensor shape, for instance, between down and up sections of the encoder-decoder), or additional layers added in to disrupt normal model behaviour.

Short circuits Following the lead of hardware hacking [4], additional rogue connections can be added between any artificial neurons in any layers of the network.

The slowness of Stable Audio inference does not facilitate deployment in live performance except perhaps via background rendering.

At the time of writing, a Stable Audio 2.0 model exists and can be tried online[9], but has not been open sourced. The five Detroit techno prompts used above were tested with SA2; plausible techno could be created, but were not convincing historically. Just as for Stable Audio 1.0 examined herein, this is not especially unsurprising, since the model was ethically trained on non-copyrighted audio, not including actual Detroit techno in the training corpus. Specialisation of the model towards historical accuracy could be undertaken, at the likely expensive of generality (as found for MusicLDM in Ronchini et al. [20]). Whether the range of sound transformation possible through model hacking would be more compromised for an over-fitted fine-tuned model is an open question. Research in this area depends on open source code availability; hacking of the suno or udio proprietary models, for example, is not practicable at this time.

[9] https://stableaudio.com/

The manipulations detailed in this paper are not immediately available to musicians to explore without use of the source code provided directly (though it doesn't take many adjustments to hack a particular layer and change the prompt and noise amplitude). A more used friendly interface, and a more wide ranging user study across a free choice of prompts and model manipulations, would be a further interesting step.

4 Summary

This paper investigated sound transformations achievable by manipulating the parameters of the Stable Audio 1.0 model. Keeping generation fixed to a set random number seed and prompt, parameter changes within and across layers were investigated, often leading to interesting sound transformations. The model was able to access sounds well outside of the norm, not by reaching new places in the innermost latent space of the encoder-decoder, but by fundamentally changing the trained parameters utilised in the model itself for generation and rendering. Many more potential hacks are available including more profound manipulation of text-to-audio models.

References

[1] Van Den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., Kavukcuoglu, K., et al., "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 12, 2016.

[2] Copet, J., Kreuk, F., Gat, I., Remez, T., Kant, D., Synnaeve, G., Adi, Y., and Défossez, A., "Simple and controllable music generation," *Advances in Neural Information Processing Systems*, 36, pp. 47704–47720, 2023.

[3] Bergstrom, I. and Lotto, R. B., "Code Bending: A new creative coding practice," *Leonardo*, 48(1), pp. 25–31, 2015.

[4] Ghazala, R., *Circuit-Bending: Build your own alien instruments*, John Wiley & Sons, 2005.

[5] Blackwell, A. F., Cocker, E., Cox, G., McLean, A., and Magnusson, T., *Live coding: a user's manual*, MIT Press, 2022.

[6] Bovermann, T. and Griffiths, D., "Computation as material in live coding," *Computer music journal*, 38(1), pp. 40–53, 2014.

[7] Broad, T., Leymarie, F. F., and Grierson, M., "Network bending: Expressive manipulation of generative models in multiple domains," *Entropy*, 24(1), p. 28, 2022.

[8] Yee-King, M., "Latent spaces: A creative approach," in *The Language of Creative AI: Practices, Aesthetics and Structures*, pp. 137–154, Springer, 2022.

[9] Collins, N., Ruzicka, V., and Grierson, M., "Remixing AIs: mind swaps, hybrainity, and splicing musical models," in *Proc. The Joint Conference on AI Music Creativity*, 2020.

[10] Dzwonczyk, L., Cella, C. E., and Ban, D., "Network Bending of Diffusion Models for Audio-Visual Generation," 2024.

[11] Cascone, K., "The Aesthetics of Failure:"Post-Digital" Tendencies in Contemporary Computer Music," *Computer Music Journal*, 24(4), pp. 12–18, 2000.

[12] Hegarty, P., *Noise/Music: A History*, Continuum, New York, 2007.

[13] Evans, Z., Parker, J. D., Carr, C., Zukowski, Z., Taylor, J., and Pons, J., "Stable audio open," *arXiv preprint arXiv:2407.14358*, 2024.

[14] Evans, Z., Parker, J. D., Carr, C., Zuckowski, Z., Taylor, J., and Pons, J., "Long-form music generation with latent diffusion," in *Proceedings of the International Symposium on Music Information Retrieval*, 2024.

[15] Cosgrove, S., "Techno! The New Dance Sound of Detroit (liner notes)," 1988.

[16] Collins, N., Schedel, M., and Wilson, S., *Electronic music*, Cambridge University Press, 2013.

[17] McFee, B., Raffel, C., Liang, D., Ellis, D. P., McVicar, M., Battenberg, E., and Nieto, O., "librosa: Audio and music signal analysis in python." in *SciPy*, pp. 18–24, 2015.

[18] Collins, N., "SCMIR: A SuperCollider music information retrieval library," in *Proceedings of the International Computer Music Conference (ICMC)*, 2011.

[19] Smalley, D., "Spectromorphology: explaining sound-shapes," *Organised sound*, 2(2), pp. 107–126, 1997.

[20] Ronchini, F., Comanducci, L., Perego, G., and Antonacci, F., "PAGURI: a user experience study of creative interaction with text-to-music models," 2024.

[21] Karimi Mahabadi, R., Henderson, J., and Ruder, S., "Compacter: Efficient low-rank hypercomplex adapter layers," *Advances in Neural Information Processing Systems*, 34, pp. 1022–1035, 2021.

[22] Plitsis, M., Kouzelis, T., Paraskevopoulos, G., Katsouros, V., and Panagakis, Y., "Investigating personalization methods in text to music generation," in *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1081–1085, IEEE, 2024.

[23] Ding, Y. and Lerch, A., "Parameter-Efficient Transfer Learning for Music Foundation Models," *arXiv preprint arXiv:2411.19371*, 2024.

[24] Li, E., Liu, J., and Mehra, G., "Efficient Fine-Tuning of CLAP-Guided Latent Stable Diffusion for High Fidelity Music Generation," Technical report, Harvard, 2024.

Audio Engineering Society

Conference Paper 16

Presented at the AES International Conference on Machine Learning and Artificial Intelligence for Audio
2025 September 8–10, London, UK

This paper was peer-reviewed as a complete manuscript for presentation at this conference. This paper is available in the AES E-Library (http://www.aes.org/e-lib), all rights reserved. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Journal of the Audio Engineering Society.

Simulating 3D Acoustic Radiation and Scattering in the Frequency Domain with Fourier Neural Operators (FNOs)

James Hipperson[1,2], Trevor Cox[1], and Jonathan Hargreaves[1]

[1] *University of Salford*
[2] *Funktion-One Research Ltd.*

Correspondence should be addressed to James Hipperson (j.r.hipperson@edu.salford.ac.uk)

ABSTRACT

Acoustic wave behaviour is described by the acoustic wave equation, which is a partial differential equation (PDE). PDEs are difficult to solve except for in the simplest cases, which necessitate the use of simplified models (such as geometrical approaches) or numerical methods (such as Finite Element) for real-world applications like room acoustics and loudspeaker radiation. Neural Operators are an emerging class of deep learning architectures that learn to approximate a solution to a PDE. Fourier Neural Operators (FNOs) project input functions onto Fourier modes, and parametrise the kernel operator in Fourier space. However, a large amount of data is still required, and the training process can be challenging. This paper presents a proof-of-concept using a U-shaped Fourier Neural Operator to solve the 3D Helmholtz Equation in air with pairs of simplified loudspeakers. The network is able to learn wave behaviour with varying accuracy across a range of frequencies, ranging from a Mean Absolute Error (MAE) of 0.12 at 500 Hz to 0.28 at 4 kHz. Once trained, inference is 15 to 3000 times faster than the Boundary Element Method (BEM). Although still at an early stage of development, Neural Operators show significant potential for a wide range of simulation applications.

1 Introduction

Most physical phenomena in the universe can be described by partial differential equations (PDEs). Unfortunately, most PDEs are difficult to solve. For acoustics, the PDE is the wave equation, and its time harmonic / frequency domain equivalent, the Helmholtz equation. In audio, this can be used to model scenarios such as loudspeaker radiation and room acoustics. The conventional methods for solving the Helmholtz equation for non-trivial cases are numerical, such as the Finite Element Method (FEM), Boundary Element Method (BEM) and Finite Difference Time Domain (FDTD) [1]. These are well established, but have two significant disadvantages. Any change to the model, no matter how small, typically requires the model to be solved again from scratch, and they become computationally intractable with very large numbers of elements / degrees of freedom.

Neural Operators are an emerging class of deep learning methods that seek to address these two problems. Traditional deep learning models attempt to learn a map between two finite-dimensional sets of data. Physics-informed neural networks (PINNs) learn to solve a single PDE instance by having a PDE residual in the

Table 1: Comparison of machine learning paradigms

Name	Training	Generalisation	Discretization
Convolutional Neural Network (CNN)	Data only	Image data	Fixed regular grid
Physics-Informed Neural Network (PINN)	Data and/or Physics	Single PDE instance	Meshless
DeepONet	Data only	General PDE class	Meshless
Fourier Neural Operator (FNO)	Data only	General PDE class	Any regular grid
Physics-Informed Neural Operator (PINO)	Data and/or Physics	General PDE class	Any regular grid

loss function (physics loss). Neural operators learn an approximation of the solution operator for a PDE class; a comparison of common paradigms is given in Table 1. The motivation for trying to learn the physics, rather than just the outcomes, is that in acoustics, the free-space solutions to the PDEs are well known and understood, so can be used as building blocks for other problems. A more significant problem is that without physical constraints, data-trained models can produce physically incorrect predictions.

Deep Learning has been applied to many problems in audio signal processing, but there are fewer applications to solving the wave equation directly. Most applications are from Geophysics, where the heterogenous wave equation and inverse problems introduce additional complexity that make neural networks an especially attractive solution compared to other numerical methods.

Some examples of airborne acoustics in physics-informed machine learning: A novel method for Nearfield Acoustic Holography (NAH) [2]. Using a CNN combined with a linear layer performing the Kirchoff-Helmholtz propagation, in effect, introducing a physics constraint via uniqueness (the output of the neural network must map to the correct result via the KH layer). A data-free PINN (only physics loss) for solving the acoustic wave equation [3]. A PINN to solve the wave equation in 3D in a simple cuboidal volume [4]. Neural operator methods applied to acoustics are not common, but [5] is an example of applying DeepONet to predict sound fields in complex 3D rooms with parametric source positions.

In this paper, we will explore the use of Fourier Neural Operators to predict the sound field near simplified loudspeakers (reflecting cubes with one radiating face) in free field.

1.1 Neural Operators

Neural Operators are motivated by the idea that if neural networks can be universal function approximators, they could also work as universal operator approximators. An early example being [6] with DeepONet in 2019.

1.2 Fourier Neural Operators (FNO)

Neural Operators were introduced in [8, 9] in 2020-21, utilising the principles that convolution is a special type of integral operator, and that spatial convolution is multiplication in the Fourier domain. This can be substantially accelerated by the Fast Fourier Transform (FFT). [7] introduced the Fourier Neural Operator (FNO) in 2021. Eq. 1 summarises the approach. The result of the integral operator K acting on function v_t at point x is approximated by the inverse Fourier transform F^{-1} of R_ϕ (parameterised by the network) multiplied by the Fourier transform F of v_t (convolution in the spatial domain) for every point x in domain D. For a complete explanation of the approach, refer to [7].

$$\big(K(\phi)v_t\big)(x) = F^{-1}\Big(R_\phi \cdot F(v_t)\Big)(x) \qquad \forall x \in D \quad (1)$$

Fig. 1b shows the general structure of the Fourier layer, and Fig. 1a shows how many layers may be connected to form a deep network. The first fully connected layer P projects the input data to a higher-dimensional latent space. One or more Fourier layers F transform the input into the Fourier domain and truncate the higher modes before performing the inverse Fourier transform F^{-1}. A skip connection has a linear transformation/bias W and σ is a non-linear activation function. A final fully connected layer Q projects back to

the target dimension. Truncating high-order modes has similarities to image compression, exploiting the fact that in most physical systems, higher-frequency modes contain relatively little information about the system. This improves training and generalisation, at the cost of some small details. Parallels can be seen to the modal expansion approach in [10].

1.3 U-Shaped Neural Operators (U-NO)

The U-NO is inspired by similar structures in Convolutional Neural Networks (CNNs), Autoencoders and U-Nets for image-to-image applications. This approach was necessitated by the structure of images - even a 64x64 image has 4096 pixels. A deep, fully connected network operating on such high-dimensional data would have extreme memory requirements and be very difficult to train. CNNs dramatically reduced the number of parameters while still providing good performance, making them a viable solution for computer vision. [12] demonstrated the now-classic convolutional neural network in 1998 which famously outperformed previous computer vision techniques by using convolutional layers and hierarchical feature extraction. [13] introduced the U-Net structure in 2015 for medical imaging applications, where the dimensionality is reduced, then expanded again, with skip connections. [14] introduced the U-NO in 2023. Just like in [13], the contracting and expanding structure means that the dimensionality and number of parameters are reduced. This enables models of systems that would be difficult or impossible to solve using numerical methods (fluid

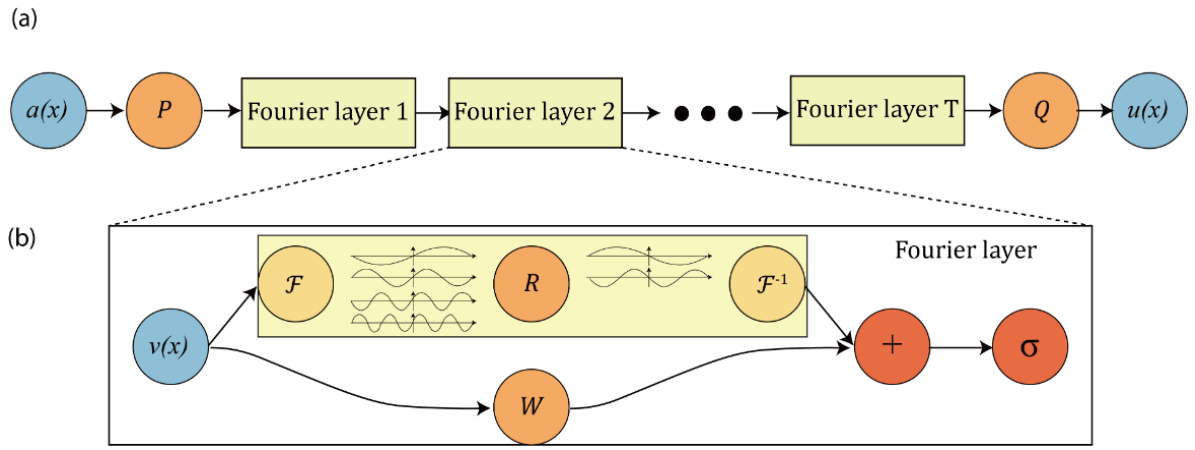


Fig. 1: (a) The use of multiple Fourier layers to map from a to u in a deep network (b) One Fourier Layer. From [7]

.

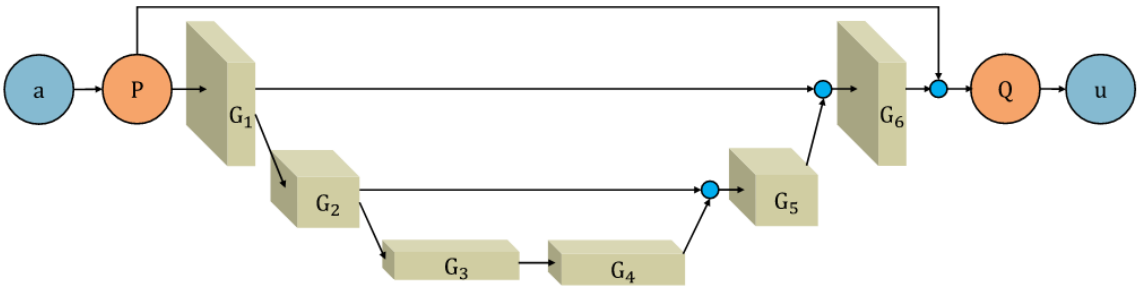


Fig. 2: U-NO structure (adapted from [11]).

dynamics, waveform inversion in geophysics) and, as demonstrated in this paper, enables significantly accelerated solution of acoustic models, opening up new applications and possibilities.

[11] demonstrates the effectiveness of a U-NO performing the challenging Geophysics task of seismic waveform inversion. Using a known source position and several frequencies, the network predicts the pressure field so that the velocity in the propagation medium can be estimated. Discontinuities in the velocity of the medium cause refraction, reflection and scattering. Correspondingly in airborne acoustics, reflection and scattering are frequently encountered in rooms and radiation from loudspeakers. Refraction occurs in outdoor sound propagation, where there are atmospheric temperature gradients. Fig. 2 shows the U-shaped network structure, with skip connections and contracting and expanding Fourier layer dimensions (G1-6). P and Q are fully connected layers as discussed in Section 1.2, and in practice are typically pairs of fully connected layers (Table 2).

2 Methods

The basic U-NO network from [11] was tested on a problem consisting of randomly generated pairs of solid reflecting cubes in free field. Each cube has one randomly chosen face radiating. Even with such a simple setup, the resulting sound pressure field can be very complex (See Fig. 3c)). The cubes were randomly positioned within the grid (without intersecting) and ranged from 2-20 cm in size. Fig. 3 shows three of the four network inputs. a) The 2D radiating faces within the 3D grid, represented by +1 or -1 depending on the orientation. b) Shows the volumetric binary mask of the cubes, represented by 1s in an empty grid of zeros. c) Shows the real component of the output from the Boundary Element Method (BEM) calculation, with the cube volumes masked off to zero pressure to remove any artefacts from the BEM calculation. The fourth input (not shown) is the frequency value scaled to a logarithmic spacing between 0 (0.5 kHz) and 1 (4 kHz). BEM is naturally suited to the problem of radiation in an infinite domain as only the boundaries must be meshed. Numerical methods are a convenient method for generating acoustic data for training. Disadvantages are computation time and that the data is perfectly clean, e.g. no noise, variation in acoustic parameters, temperature, measurement positions, etc. Neural networks operating on real measurement data would have all of these additional factors to contend with.

The network was implemented with Pytorch 2.6 [15]. The input data was generated programmatically with numpy and the output data (complex-valued pressure) was generated with BEM using bempp-cl ([16]). The mesh size was varied with frequency to use a (relatively coarse) minimum of 5 elements per wavelength. Standardisation was performed by scikit-learn [17]. In all cases, the predicted field was a 64x64x64 3D grid with real and imaginary components of sound pressure. Fig. 2 shows the U-shaped network structure.

An original dataset of 1000 pairs of cubes at 4 frequencies was augmented by spatially mirroring/flipping the data to expand the dataset to 2000 batches (8000 individual samples). This was split into a training set of 1600 batches, a validation set of 200 and a test set (unseen in training) of 200. The test set contained the same frequencies, but unseen configurations of size, position and radiating faces. The generalisability to

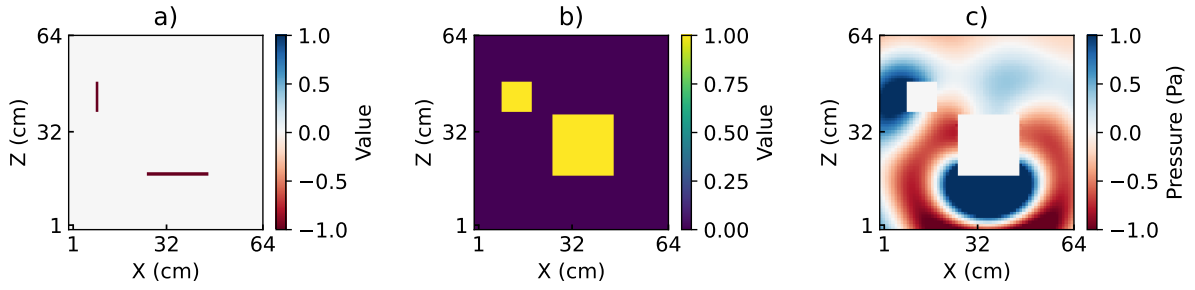


Fig. 3: Network inputs: a) Radiating Faces b) Binary Mask c) BEM Output (Frequency value channel not shown).

Table 2: Model Summary (width=8)

Layer	Width	Modes
Fully Connected 1	in,4	
Fully Connected 2	4,8	
Fourier 1	8,16	48,24
Fourier 2	16,32	32,16
Fourier 3	32,64	16,8
Fourier 4	64,32	32,8
Fourier 5	64,16	48,16
Fourier 6	32,16	64,24
Fully Connected 3	24,32	
Fully Connected 4	32,out	

unseen frequencies was not tested in this work, but is an important test for future work. The grid resolution was 1 cm per point, with octave band centre frequencies from 1 kHz to 4 kHz. The batch size was 4 (4 frequencies, 2 configurations = 8 total samples).

Width in the context of FNO has the same meaning as in a CNN: the base channel dimension, which is typically expanded as spatial dimensions are reduced, and vice versa (see Fig.2 and Table 2).

The learning rate was initially 0.001 and decayed by 0.5 every 50 epochs. The model was run for 150 epochs. Early stopping was implemented by saving the model state with the lowest validation loss. Training of the width=8 model took 14 hours on a single NVidia GTX 1080Ti (11 GB of memory).

The model structure is presented in Table 2. Fourier modes were not truncated to ensure accurate prediction of high-frequency wave effects, but this may not have been essential. Training of the width=16 model took 7 hours on a single NVidia A100 (40 GB of memory). The model structure was the same as in Table 2 but with the width of each layer doubled. The modes remained the same. The initial learning rate was reduced to 0.0005, decaying by 0.7 every 40 epochs. The loss function for both models was $0.9 \cdot MAE + 0.1 \cdot MSE$. This proportion was chosen empirically to encourage the network to focus on a generally correct solution, rather than getting stuck on outliers. Further improvement to the loss function may be possible.

$$MAE = \frac{\sum_{i=1}^{N} |y_{predicted} - y_{true}|}{N} \tag{2}$$

$$MSE = \frac{\sum_{i=1}^{N} (y_{predicted} - y_{true})^2}{N} \tag{3}$$

3 Results

3.1 Training

Fig. 4 shows the contrast between training on a single frequency vs. several frequencies. Convergence is much slower in the multiple frequency case (b), and validation loss stalls at around 0.33. In contrast, the single frequency case (a) is able to reach a validation loss of 0.2. This suggests that further optimisations to the training setup could be made. Increasing the layer width to 16 (c) did not significantly improve the performance which supports this.

3.2 Error vs. frequency and layer width

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (y_{predicted} - y_{true})^2}{N}} \tag{4}$$

Table 3 shows the Mean Absolute Error (MAE) (Eq. 2), Standard Deviation (SD) and Root Mean Squared Error (RMSE) (Eq. 4) vs. frequency, over the complete test set (200 samples of four frequencies each). The results clearly show the effects of spectral bias: the network is much more effective at learning lower-frequency solutions. The standard deviation shows that the variation in performance increases with frequency.

3.3 2D slices of 3D pressure fields

XZ plane slices at Y=32 cm of real and imaginary parts in Pascals are plotted in Figs. 5,6,7. Difference plotted as $P_{Predicted} - P_{True}$. Differences are highest at the reflecting boundaries. Both the pressure value and gradient are highest here. Fig. 5 and Fig. 6 show similar behaviour in that the general structure is correct. MAE throughout the domain can be less than 0.3, but the maximum absolute error can exceed 3.0 at the boundary or at very small singularities. Fig. 7 on the other hand demonstrates the difficulty of predicting the correct details as wavelength decreases and complexity of the pressure distribution increases. Different configurations are shown for each frequency to demonstrate the diversity of wavefields.

Table 3: Error over test set

	Width: 8			Width: 16		
Frequency	MAE	SD	RMSE	MAE	SD	RMSE
0.5 kHz	0.135	0.217	0.256	0.120	0.180	0.216
1 kHz	0.155	0.251	0.295	0.134	0.193	0.235
2 kHz	0.291	1.240	1.274	0.292	1.244	1.277
4 kHz	0.269	1.160	1.191	0.281	1.591	1.616

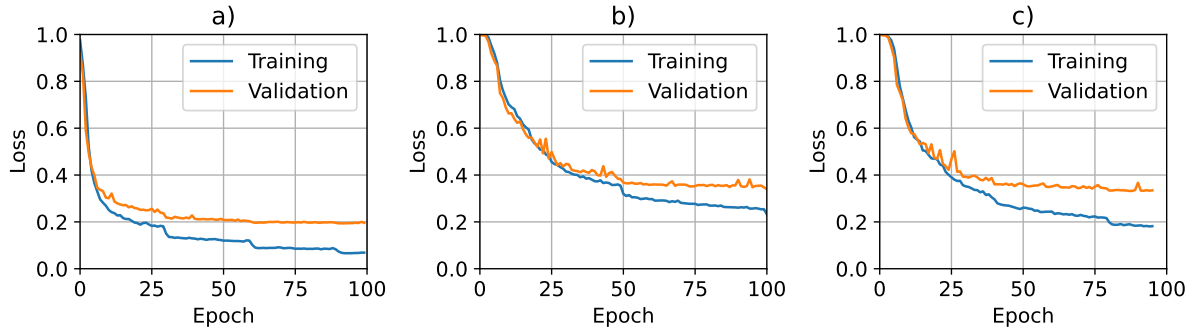


Fig. 4: Training curves: a) Width 8, trained on a single frequency (1 kHz) b) Width 8, trained on four frequencies (0.5 kHz, 1 kHz, 2 kHz, 4 kHz) c) Width 16, trained on four frequencies (0.5 kHz, 1 kHz, 2 kHz, 4 kHz).

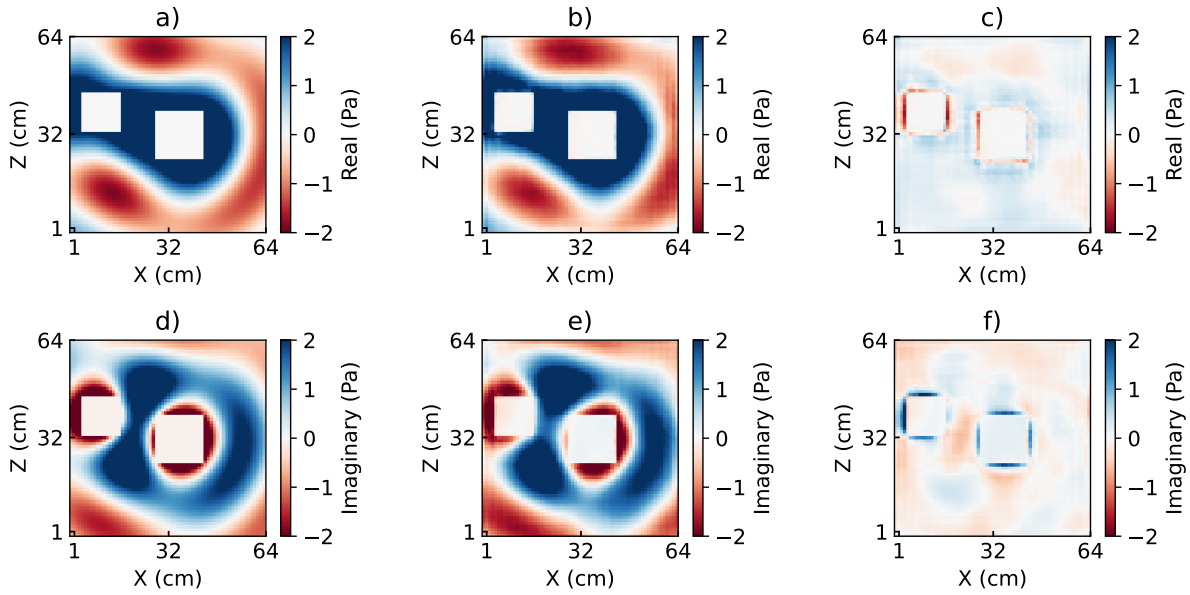


Fig. 5: Acoustic pressure at 1 kHz, Width=8 model: a) Ground Truth (real part) b) Prediction (real part) c) Difference (real part). d) Ground Truth (imaginary part) e) Prediction (imaginary part) f) Difference (imaginary part)

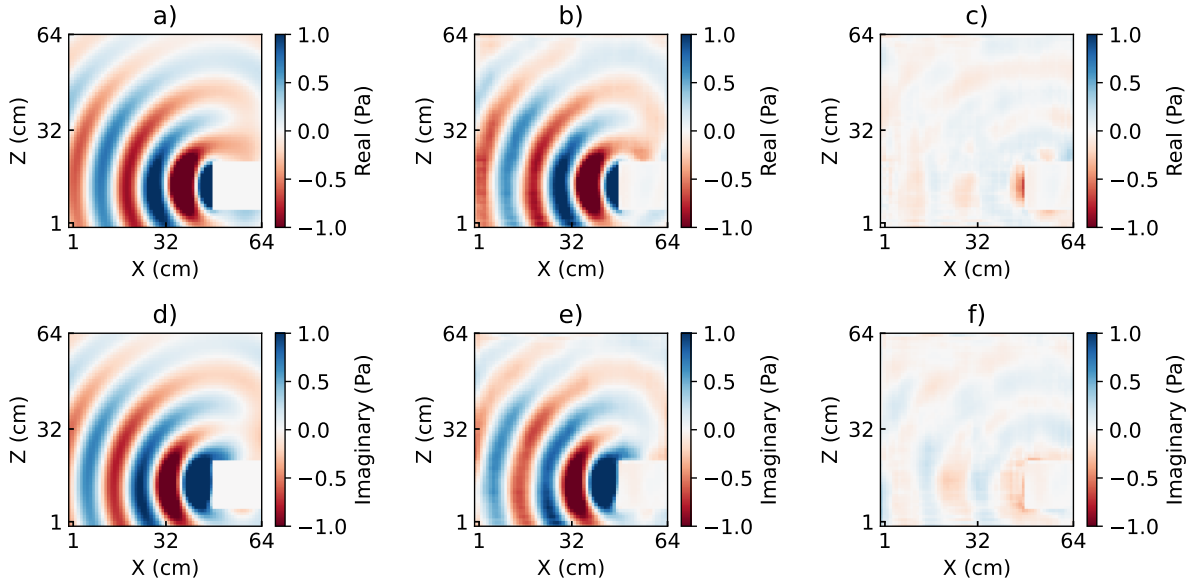


Fig. 6: Acoustic pressure at 2 kHz, Width=8 model: a) Ground Truth (real part) b) Prediction (real part) c) Difference (real part). d) Ground Truth (imaginary part) e) Prediction (imaginary part) f) Difference (imaginary part)

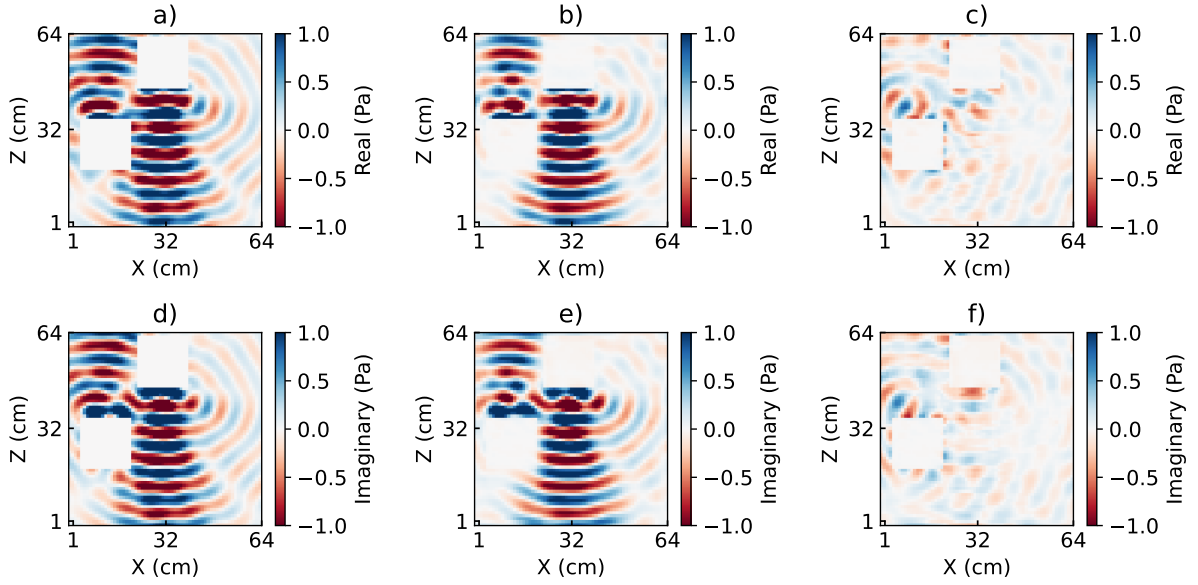


Fig. 7: Acoustic pressure at 4 kHz, Width=8 model: a) Ground Truth (real part) b) Prediction (real part) c) Difference (real part). d) Ground Truth (imaginary part) e) Prediction (imaginary part) f) Difference (imaginary part)

3.4 Inference time

Each prediction from the width 8 model took 0.05 seconds for all four frequencies. In comparison, the BEM calculations ranged from 1.5 seconds (for small meshes/high frequencies) to over 150 seconds (for large meshes/low frequencies). This equates to a speed-up range of 30-3000 times. Each prediction from the width 16 model took 0.1 seconds, a minimum speed-up of 15 times. Inference for both models was performed on a GTX 1080Ti GPU. BEM calculations were performed on a CPU (AMD Ryzen 9 3900X). Some preliminary tests were performed running BEM calculations on a GPU in the interest of experimental fairness, but performance was typically worse than on a CPU. Two reasons for this are the main memory to GPU memory transfer bottleneck, and that solution of the dense global matrix equation is not easily parallelisable. [18]

4 Discussion

The results show that FNOs can learn acoustic wave behaviour in 3D across a range of frequencies, but with diminishing accuracy at higher frequencies. However, the error is not general across all samples, but concentrated in a small number of samples where the network fails significantly. GPU memory limited the layer width and batch size substantially, which limited the accuracy of the results and likely contributed to the training difficulties. However, the results show that inference on conventional (consumer) computing hardware is possible, and useful results with improved accuracy will likely become possible in the near future, as the available GPU memory in devices increases.

The most challenging aspect was hyperparameter tuning. The model readily learned single-frequency solutions, but in the multi-frequency case, the model failed to converge in most configurations. A full ablation study would be useful to understand whether the model can be further simplified for airborne homogenous acoustics problems. The training curves in Fig. 4 show the hyperparameter optimisation difficulty: Overfitting, which indicates excessive model complexity, but poor performance when learning multiple frequencies. The layer width was significantly reduced from the Geophysics model (from 32 to 8) but all six layers and all Fourier modes were retained, which may not have been necessary. Larger layer width but fewer layers or Fourier modes may produce better results more efficiently.

A significant limitation of FNOs is the requirement for a regular grid. [19] explores a method for deforming a geometry mesh into a regular grid for use with FNO layers, then transforming back to the original mesh geometry. Another limitation is that the network in this paper (and FNOs generally) only have indirect physics constraints in the form of inductive bias (trained on the output from a numerical method) and the parameterisation of the kernel operator in Fourier space. There may be better network architectures that embed the constraints of the wave equation, leading to easier training and more accurate results. It is remarked in [20] that compared to data-based machine learning (which has existed for longer), the network architectures in operator learning are still primitive.

In terms of further work, comparison of operator learning models with a conventional image-to-image CNN baseline for the same task would be valuable. Interpretability is also a key area of interest. Currently it is unclear exactly what FNOs learn, and whether the latent space has any similarity to known physics. One way of testing this would be to make predictions for problems with well-understood analytical solutions, and inspect the intermediate data passing between layers in the network.

A residual learning approach could have benefits. Rather than having the network learn and predict the full solution, it would learn and predict the difference between an approximate solution (point source model) and an accurate solution (BEM). This could train faster and produce more accurate results.

Another important direction for further work is how to efficiently and usefully represent complex and arbitrary geometry in operator learning. Additionally, architectural physics constraints should be explored to determine whether this can improve the ease of training and the accuracy of results. The dataset of 8000 samples was around 20 GB. Optimal datasets for more complex problems may be several times larger, becoming increasingly time-consuming to generate and train on. Making training more efficient with physics constraints is therefore a key area for further work.

5 Summary

FNO was applied to an acoustic radiation and scattering problem with simplified loudspeaker geometry, at medium to high frequencies. FNOs are capable of approximating a solution operator for the 3D wave equation. The accuracy of results is reduced at high frequencies, but inference is 15-3000x faster than BEM, as a forward pass through the network is a constant computational complexity (depending only on the model size) whereas numerical methods scale by $O(N)$ (FEM) to $O(N^2)$ (BEM), with higher frequencies (or larger domains) requiring more degrees of freedom and therefore more computation.

FNOs can be thought of as a generalisation of image-to-image CNNs, but because they are general, the network must learn the underlying physics from scratch. However, as mentioned in the introduction, the physics of acoustic waves is well known already. Therefore, the most important and obvious direction for further work is how to incorporate domain knowledge and known physics constraints to build operator learning networks that are more specific to acoustic problems.

6 Acknowledgements

This work was supported by Funktion-One Research Ltd. as part of the CDT in Sustainable Sound Futures.

References

[1] Sakuma, T., Sakamoto, S., and Otsuru, T., editors, *Computational Simulation in Architectural and Environmental Acoustics: Methods and Applications of Wave-Based Computation*, Springer, Tokyo, 1 edition, 2014, ISBN 978-4-431-54453-1, doi:10.1007/978-4-431-54454-8.

[2] Olivieri, M., Pezzoli, M., Antonacci, F., and Sarti, A., "A Physics-Informed Neural Network Approach for Nearfield Acoustic Holography," *Sensors*, 21(23), p. 7834, 2021, ISSN 1424-8220, doi:10.3390/s21237834.

[3] Sethi, H., Pan, D., Dimitrov, P., Shragge, J., Roth, G., and Hester, K., "Hard enforcement of physics-informed neural network solutions of acoustic wave propagation," *Computational Geosciences*, 27(5), pp. 737–751, 2023, ISSN 1573-1499, doi:10.1007/s10596-023-10232-3.

[4] Schoder, S., "Physics-Informed Neural Networks for Modal Wave Field Predictions in 3D Room Acoustics," *Applied Sciences*, 15(2), 2025, ISSN 2076-3417, doi:10.3390/app15020939.

[5] Borrel-Jensen, N., Goswami, S., Engsig-Karup, A. P., Karniadakis, G. E., and Jeong, C.-H., "Sound propagation in realistic interactive 3D scenes with parameterized sources using deep neural operators," *Proceedings of the National Academy of Sciences*, 121(2), p. e2312159120, 2024, ISSN 0027-8424, 1091-6490, doi:10.1073/pnas.2312159120.

[6] Lu, L., Jin, P., and Karniadakis, G. E., "DeepONet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators," *Nature Machine Intelligence*, 3(3), pp. 218–229, 2021, ISSN 2522-5839, doi:10.1038/s42256-021-00302-5.

[7] Li, Z., Kovachki, N. B., Azizzadenesheli, K., liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A., "Fourier Neural Operator for Parametric Partial Differential Equations," in *International Conference on Learning Representations*, 2021.

[8] Kovachki, N. B., Li, Z., Liu, B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A. M., and Anandkumar, A., "Neural Operator: Learning Maps Between Function Spaces," *CoRR*, abs/2108.08481, 2021.

[9] Anandkumar, A., Azizzadenesheli, K., Bhattacharya, K., Kovachki, N., Li, Z., Liu, B., and Stuart, A., "Neural Operator: Graph Kernel Network for Partial Differential Equations," in *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2019.

[10] Xu, B. and Sommerfeldt, S. D., "A hybrid modal analysis for enclosed sound fields," *The Journal of the Acoustical Society of America*, 128(5), pp. 2857–2867, 2010, ISSN 0001-4966, 1520-8524, doi:10.1121/1.3493429.

[11] Zou, C., Azizzadenesheli, K., Ross, Z. E., and Clayton, R. W., "Deep neural Helmholtz operators for 3-D elastic wave propagation and inversion," *Geophysical Journal International*, 239(3), pp. 1469–1484, 2024, ISSN 0956-540X, 1365-246X, doi:10.1093/gji/ggae342.

[12] Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P., "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, 86(11), pp. 2278–2324, 1998, ISSN 00189219, doi:10.1109/5.726791.

[13] Ronneberger, O., Fischer, P., and Brox, T., "U-Net: Convolutional Networks for Biomedical Image Segmentation," in N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pp. 234–241, Springer International Publishing, Cham, 2015, ISBN 978-3-319-24574-4.

[14] Rahman, M. A., Ross, Z. E., and Azizzadenesheli, K., "U-NO: U-shaped Neural Operators," *Transactions on Machine Learning Research*, 2023, ISSN 2835-8856.

[15] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E. Z., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S., "PyTorch: An Imperative Style, High-Performance Deep Learning Library," *CoRR*, abs/1912.01703, 2019.

[16] Betcke, T. and Scroggs, M., "Bempp-cl: A fast Python based just-in-time compiling boundary element library." *Journal of Open Source Software*, 6(59), p. 2879, 2021, ISSN 2475-9066, doi:10.21105/joss.02879.

[17] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E., "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, 12, pp. 2825–2830, 2011.

[18] Molina-Moya, J., Martínez-Castro, A. E., and Ortiz, P., "An Iterative Parallel Solver in GPU Applied to Frequency Domain Linear Water Wave Problems by the Boundary Element Method," *Frontiers in Built Environment*, 4, p. 69, 2018, ISSN 2297-3362, doi:10.3389/fbuil.2018.00069.

[19] Li, Z., Kovachki, N. B., Choy, C., Li, B., Kossaifi, J., Otta, S. P., Nabian, M. A., Stadler, M., Hundt, C., Azizzadenesheli, K., and Anandkumar, A., "Geometry-Informed Neural Operator for Large-Scale 3D PDEs," in *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[20] Azizzadenesheli, K., "Machine Learning on Function Spaces," Tutorial at the International Conference on Machine Learning, 2024.

Audio Engineering Society

Conference Paper 17

Presented at the AES International Conference on Machine Learning and Artificial Intelligence for Audio
2025 September 8–10, London, UK

This paper was peer-reviewed as a complete manuscript for presentation at this conference. This paper is available in the AES E-Library (http://www.aes.org/e-lib), all rights reserved. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Journal of the Audio Engineering Society.

AudioGAN: A Compact and Efficient Framework for Real-Time High-Fidelity Text-to-Audio Generation

HaeChun Chung[1]

[1] *KT Corporation, Seoul, Republic of Korea*

Correspondence should be addressed to HaeChun Chung (hc.chung@kt.com)

ABSTRACT

Text-to-audio (TTA) generation can significantly benefit the media industry by reducing production costs and enhancing work efficiency. However, most current TTA models—primarily diffusion-based—suffer from slow inference speeds and high computational costs. In this paper, we introduce AudioGAN, the first successful Generative Adversarial Networks (GANs)-based TTA framework that generates audio in a single pass, thereby reducing model complexity and inference time. To overcome the inherent difficulties in training GANs, we integrate multiple ,contrastive losses and propose innovative components—Single-Double-Triple (SDT) Attention and Time-Frequency Cross-Attention (TF-CA). Extensive experiments on the AudioCaps dataset demonstrate that AudioGAN achieves state-of-the-art performance while using 90% fewer parameters and running 20 times faster, synthesizing audio in under one second. These results establish AudioGAN as a practical and powerful solution for real-time TTA.

1 Introduction

In today's competitive media environment, producing high-quality audiovisual content is essential. However, creating professional-grade sound typically requires expensive field or studio recordings, which drive up production costs and restrict creative endeavors for producers. Text-to-audio (TTA) generation can address these challenges by synthesizing sound directly from text, reducing production costs and streamlining workflows. Nevertheless, current TTA models face significant technical limitations including slow inference speeds and large model sizes that restrict their practical deployment and real-time applications, creating barriers to widespread adoption in professional media production pipelines.

Recent research in TTA has seen a surge of interest, with various approaches emerging to synthesize realistic and expressive audio from textual descriptions. Diffusion-based models [1, 2, 3, 4, 5, 6, 7] achieve impressive quality by iteratively refining noisy latent representations, while autoregressive models like AudioGen [8] generate audio token by token to closely match the input text. However, diffusion-based methods require iterative denoising steps, resulting in high computational costs and slow inference speeds. Meanwhile, autoregressive approaches often suffer from error accumulation and large model sizes. These limitations underscore the need for more efficient and compact TTA models that can generate high-quality audio with rapid inference and low computational cost for practical deployment.

According to these requirements, we propose AudioGAN, a novel text-to-audio generation framework based on Generative Adversarial Networks (GANs) [9]. Unlike diffusion-based approaches that rely on iterative processing, AudioGAN generates audio in a single pass with a compact architecture. To address the inherent training challenges of GANs, we incorporated multiple contrastive losses into the training process. Moreover, to effectively embed textual information into the audio, we introduced innovative methods—namely, Single-Double-Triple (SDT) Attention and Time-Frequency Cross-Attention (TF-CA). We conducted various experiments to evaluate our method on the AudioCaps dataset. Our results show that AudioGAN not only achieves state-of-the-art performance but also significantly reduces model parameters and GPU memory usage. Notably, AudioGAN generates high-quality audio from text description in less than one second. To our knowledge, this is the first successful application of a GANs model in TTA, and the experimental results demonstrate AudioGAN a practical solution for real-time media production.

2 Related Works

Recent advances in TTA generation have led to a variety of approaches that aim to synthesize realistic and expressive audio directly from textual descriptions. Key milestones in this field include diffusion-based methods such as DiffSound [1], AudioLDM [2, 6], Make-an-Audio [3, 5], and Tango [4, 7], which typically combine text-conditioned latent diffusion with VAE decoders [10, 11]. These two-stage architectures have pushed the boundaries of audio fidelity by iteratively refining noisy latent representations. Additionally, autoregressive models like AudioGen [8] leverage sequential token prediction to generate audio that closely aligns with the input text. However, despite their impressive performance in terms of quality, diffusion-based approaches are hindered by their slow inference speeds—often requiring hundreds of iterative denoising steps even when techniques like DDIM [12] are employed—while autoregressive models tend to suffer from error accumulation and are computationally intensive due to large model sizes [13, 14].

To address these challenges, our work departs from the conventional diffusion and autoregressive paradigms by employing a GANs [9] as the core generative model for TTA. By using a single generator model, our method eliminates the need for the multi-stage process that introduces performance gaps between separate components. This single-stage approach not only streamlines the generation process, significantly reducing both model size and computational overhead, but also enables rapid inference, making real-time application feasible.

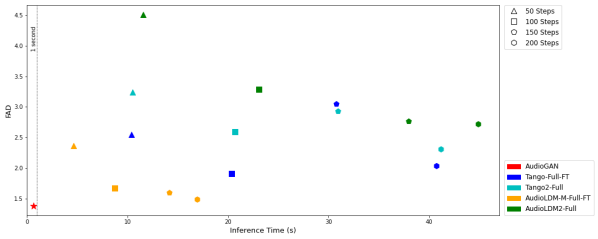


Fig. 1: Comparison of FAD scores (lower is better) and average inference times for various models on the AudioCaps test set. AudioGAN is represented by a red star, while diffusion-based models are depicted using different shapes to indicate their respective denoising steps, with colors differentiating individual models.

3 METHODS

AudioGAN comprises three integral components: a text encoder utilizing the pre-trained Contrastive Language-Audio Pre-training (CLAP) model [15], a generator constructed from GANs [9] trained via proposed methodology, and a vocoder employing the pre-trained HiFi-GAN [16]. The subsequent sections delineate the architectural design and training process of the GANs.

3.1 The training of the GAN

This section outlines our GANs training methodology. The generator synthesizes audio from textual features, while the discriminator leverages the text information to distinguish between real and synthesized audio. Multiple loss functions are employed to stabilize training and enhance performance, as illustrated in Figure 2.

3.1.1 Adversarial Loss

Within a mini-batch of N audio-text pairs, let $i \in I \equiv \{1...N\}$ denote the index of a sample. Each audio sample is converted into a mel-spectrogram m_i, and its corresponding text is processed by a text encoder to extract word features w_i and sentence features s_i. The

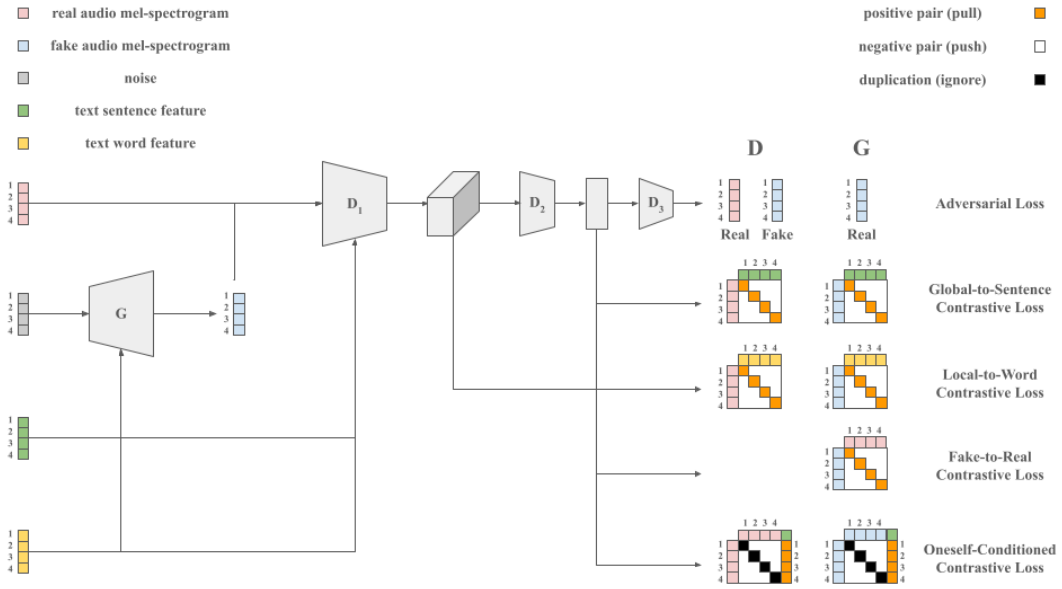


Fig. 2: A schematic of AudioGAN's training losses. The discriminator (D) optimizes four losses, while the generator (G) optimizes five losses. The audio local feature is first extracted from D_1, then processed through D_2 to obtain the global audio feature. Finally, the logit for the adversarial loss is extracted via D_3.

generator G then uses a noise vector z_i together with these text features to produce a text-conditioned mel-spectrogram, while the discriminator D evaluates both the generated and real mel-spectrograms for authenticity. We employ the hinge loss function [17] as the adversarial loss, with the objective functions for D and G detailed in the equations below.

$$\begin{aligned}\mathscr{L}_D = &-\frac{1}{N}\sum_{i\in I}\min\left(0,-1+D\left(m_i,w_i,s_i\right)\right)\\ &-\frac{1}{N}\sum_{i\in I}\min\left(0,-1-D\left(G\left(z_i,w_i,s_i\right),w_i,s_i\right)\right),\\ \mathscr{L}_G = &-\frac{1}{N}\sum_{i\in I}D\left(G\left(z_i,w_i,s_i\right),w_i,s_i\right)\end{aligned} \tag{1}$$

3.1.2 Contrastive Loss

Generated audio must accurately reflect the input text. To ensure this, we maximize mutual information by aligning global audio features with sentence-level text, local audio features with word-level text, and generated audio features with those from real audio sharing the same text [18]. We further enhance diversity by applying a oneself-conditioned contrastive loss that reduces similarity between distinct audio samples [19].

Global-to-Sentence (G2S) Contrastive Loss. To enhance mutual information at the global level, we apply a contrastive loss between the audio global features g and the sentence features s. These global features, extracted from the discriminator's penultimate layer for both real and generated audio, are encouraged to be similar for matching audio-text pairs while being pushed apart for non-matching pairs. In the equation below, $\cdot$ denotes the dot product, τ is the temperature parameter.

$$\mathscr{L}_{G2S} = -\frac{1}{N}\sum_{i\in I}\log\frac{\exp(g_i\cdot s_i/\tau)}{\sum_{k\in I}\exp(g_i\cdot s_k/\tau)} \tag{2}$$

Local-to-Word (L2W) Contrastive Loss. Each word in the text includes significant meaning, and segments of the audio should reflect these words. To capture this without explicit annotations, we use an attention mechanism that aligns audio local features l (extracted from the discriminator's final convolutional layer) with word features w. First, soft attention $\alpha_{i,j}$ is computed between l and w. Then, a contrastive loss based on a scoring function $\mathscr{S}(l,w)$ strengthens this alignment.

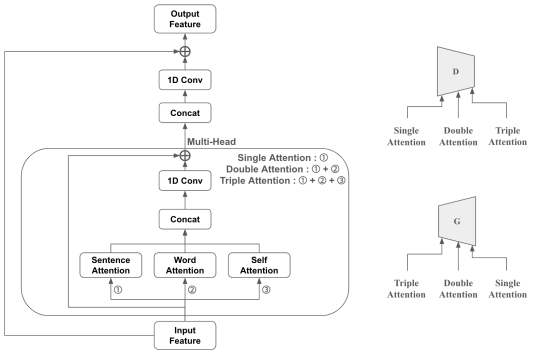


Fig. 3: (Left) Flowchart of Single-Double-Triple (SDT) Attention and combination of attention mechanism at each stage. (Right) Schematic illustrating the application of SDT Attention within the discriminator and generator.

$$\alpha_{i,j} = \frac{\exp(\gamma_1(w_i \cdot l_j))}{\sum_{k=1}^{R}\exp(\gamma_1(w_i \cdot l_k))},\ c_i = \sum_{j=1}^{R}\alpha_{i,j} l_j \tag{3}$$

$$\mathscr{S}(l,w) = \log\left(\sum_{t=1}^{T}\exp(\gamma_2(w_t \cdot c_t))\right)^{\frac{1}{\gamma_2}} \tag{4}$$

$$\mathscr{L}_{L2W} = -\frac{1}{N}\sum_{i\in I}\log\frac{\exp(\gamma_3\mathscr{S}(l_i,w_i))}{\sum\limits_{k\in I}\exp(\gamma_3\mathscr{S}(l_i,w_k))} \tag{5}$$

Where R is the number of local features, T is the number of words, and $\gamma_1, \gamma_2, \gamma_3$ are smoothing parameters.

Fake-to-Real (F2R) Contrastive Loss. To produce high-quality audio, the generator can reference real audio samples. However, directly consulting real audio limit diversity. Instead, we encourage the generated audio to resemble its real counterpart in feature space by applying a contrastive loss between the global audio features of generated samples g^f and those of real samples g^r. These features, extracted from the penultimate layer of the discriminator, guide the generator's learning without affecting the discriminator.

$$\mathscr{L}_{F2R} = -\log\frac{\exp(g_i^f \cdot g_i^r/\tau)}{\sum\limits_{k\in I}\exp(g_i^f \cdot g_k^r/\tau)} \tag{6}$$

Oneself-Conditioned Contrastive (OCC) Loss. The previous three contrastive losses ensure that each sample aligns with its corresponding text condition and that generated samples reference real data for enhanced quality. However, they focus solely on individual conditional alignment without regulating the relationships between different samples. As a result, even if text descriptions are similar, subtle differences may be lost, leading to redundant features and reduced diversity. To address this, we introduce an OCC Loss [19] that pushes apart all samples while pulling each one closer to its respective sentence feature. This loss is applied to both generated and real samples, ensuring consistent regulation across the data distribution.

$$\mathscr{L}_{OCC} = -\frac{1}{N}\sum_{i\in I}\log\frac{\exp(g_i \cdot s_i/\tau)}{\exp(g_i \cdot s_i/\tau) + \sum\limits_{k\in I} 1_{k\neq i}\exp(g_i \cdot g_k/\tau)} \tag{7}$$

3.2 The architecture of the GANs

This section describes the key components of our GANs model. To enhance text-conditioned audio generation, we integrate both self- and cross-attention mechanisms into the generator and discriminator. To balance performance with model complexity, we introduce a Single-Double-Triple Attention and a novel Time-Frequency Cross-Attention mechanism that effectively captures the temporal and spectral characteristics of audio data.

3.2.1 Single-Double-Triple (SDT) Attention

In our GANs framework, we employ both cross- and self-attention [20, 14] in a complementary manner. Cross-attention is split into sentence attention, which captures the overall meaning of the text, and word attention, which focuses on fine-grained details. Self-attention captures long-range dependencies and intricate relationships within the audio feature. We introduce a hierarchical Single-Double-Triple (SDT) Attention scheme tailored to the distinct needs of the discriminator and generator, and each combination of attention is shown in Figure 3. In the discriminator, single attention captures global semantic content, double attention refines detailed alignment, and triple attention comprehensively assesses semantic authenticity. In the generator, triple attention is first performed to build a robust initial audio representation from noise and text, then gradually reduce complexity with double and single attention layers to refine the text-audio aligning and generate high-quality audio. This structured approach enables the discriminator to evaluate both global and fine-grained details and guides the generator to develop progressively refined audio representations.

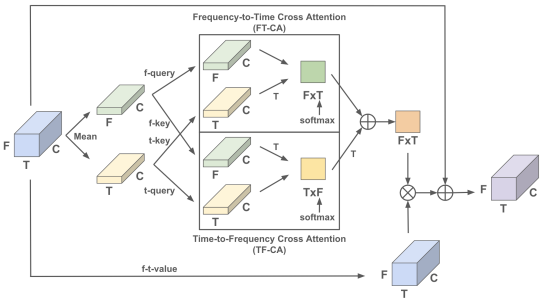


Fig. 4: Illustration of the Self Time-Frequency Cross-Attention mechanism.

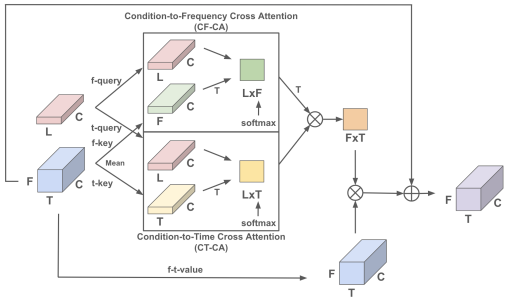


Fig. 5: Illustration of Multi Time-Frequency Cross-Attention mechanism.

3.2.2 Time-Frequency Cross-Attention

Conventional attention mechanisms [21, 22] compute the query, key, and value from the same input feature, allowing every element to attend to every other element. However, in audio data, the two axes—time and frequency—encode distinct types of information. To leverage this structure, we separate audio features into two axes, enabling us to specialize in the unique characteristics of each. In our framework, we design novel attention mechanisms for self-attention and multi-modal cross-attention.

Self Time-Frequency Cross-Attention Previous studies [23, 24, 25] have applied attention separately to the time and frequency axes to model temporal and spectral cues, respectively. However, these methods do not account for the interdependency between these dimensions. We propose a self time-frequency cross-attention mechanism that decomposes the attention process into two complementary operations: frequency-to-time and time-to-frequency (Fig. 4).

The audio features from previous hidden layers are denoted by $x \in \mathbb{R}^{F\times T\times C_x}$ where F is the number of frequency bins, T is the number of time steps, and C_x is the channel dimension. To derive frequency-specific and time-specific representations, we first average over the time and frequency dimensions, respectively:

$$x^f = \frac{1}{T}\sum_{t=1}^{T} x(:,t,:),\; x^t = \frac{1}{F}\sum_{f=1}^{F} x(f,:,:) \tag{8}$$

These aggregated features are projected into query and key spaces using trainable parameters $q^f, k^f, q^t, k^t \in \mathbb{R}^{C_x\times C_h}$ where C_h is the hidden channel dimension. In frequency-to-time cross-attention, attention weights are computed from the frequency domain to capture temporal dynamics; each element of the resulting weight matrix quantifies the degree to which the i-th frequency feature attends to the j-th time feature:

$$W_{f2t}(i,j) = \frac{\exp\left(\left(x_i^f \cdot q_i^f\right)\cdot\left(x_j^t \cdot k_j^t\right)^\top\right)}{\sum_{j=1}^{T}\exp\left(\left(x_i^f \cdot q_i^f\right)\cdot\left(x_j^t \cdot k_j^t\right)^\top\right)} \tag{9}$$

In contrast, time-to-frequency cross-attention computes attention weights in the time domain to capture spectral details. Each element represents the extent to which the i-th time feature attends to the j-th frequency feature:

$$W_{t2f}(i,j) = \frac{\exp\left(\left(x_i^t \cdot q_i^t\right)\cdot\left(x_j^f \cdot k_j^f\right)^\top\right)}{\sum_{j=1}^{F}\exp\left(\left(x_i^t \cdot q_i^t\right)\cdot\left(x_j^f \cdot k_j^f\right)^\top\right)} \tag{10}$$

These attention maps are fused into a unified weights:

$$W = W_{f2t} + W_{t2f}^\top \tag{11}$$

Finally, the output audio feature x' is obtained by element-wise multiplying the value feature—derived from the audio feature using a trainable parameter $v \in \mathbb{R}^{C_x\times C_x}$—with the combined attention weight W, followed by a residual connection:

$$x' = x + (x\cdot v)\times W \tag{12}$$

Table 1: Comparison of AudioGAN with previous models on AudioCaps dataset.

Model	Inference GPU Memory	#Param	FAD(↓)	FD(↓)	IS(↑)	KL(↓)	CLAP(↑)
AudioLDM-M-Full	10.25GB	416M	3.84	32.2	5.82	1.67	0.45
AudioLDM-M-Full-FT	10.25GB	416M	1.52	20.03	9.0	1.14	0.54
AudioLDM-L-Full	15.87GB	739M	3.69	34.69	7.77	1.56	0.42
Tango	5.65GB	866M	1.44	15.09	7.41	1.08	0.54
Tango-Full	5.65GB	866M	1.87	20.94	5.14	1.23	0.5
Tango-Full-FT	5.65GB	866M	2.0	12.53	7.75	0.95	0.56
AudioLDM2-Full	8.17GB	346M	2.74	18.5	8.08	1.25	0.46
AudioLDM2-L-Full	10.95GB	712M	2.92	20.29	7.25	1.22	0.46
Tango2	5.65GB	866M	2.83	14.51	8.79	0.97	0.58
Tango2-Full	5.65GB	866M	2.28	13.79	8.72	0.96	0.58
AudioGAN	1.34GB	28M	1.38	12.66	9.63	1.15	0.49

By separating time and frequency, this mechanism effectively captures both temporal dynamics and spectral details, improving audio synthesis quality.

Multi Time-Frequency Cross-Attention Beyond self attention, we extend our approach to condition the audio features on external textual information. Given an audio feature $x \in \mathbb{R}^{F\times T\times C_x}$ and a condition feature $c \in \mathbb{R}^{N\times C_c}$ (where N represents the number of conditions—1 for sentence-level attention and the number of tokens for word-level attention), we first aggregate the audio features along the time and frequency dimensions as in Equation (8). These aggregated representations, x_f and x_t, along with the condition c, are projected into query and key spaces using trainable parameters $q^f, q^t \in \mathbb{R}^{C_c\times C_h}$ and $k^f, k^t \in \mathbb{R}^{C_x\times C_h}$.
Unlike conventional cross-attention, where the main input is used as the query and the condition as the key, our method reverses this order: the condition serves as the query while each domain of the main input is treated as a key. This approach allows the model to evaluate the importance of time and frequency features from the condition's perspective, effectively emphasizing the most critical aspects of each domain. In the condition-to-frequency cross-attention, each element quantifies how strongly the i-th condition feature attends to the j-th frequency feature:

$$W_{c2f}(i,j) = \frac{\exp\left(\left(c_i \cdot q_i^f\right)\cdot\left(x_j^f \cdot k_j^f\right)^\top\right)}{\sum_{j=1}^{F}\exp\left(\left(c_i \cdot q_i^f\right)\cdot\left(x_j^f \cdot k_j^f\right)^\top\right)} \quad (13)$$

Similarly, in the condition-to-time cross-attention, each element reflects the degree to which the i-th condition feature aligns with the j-th time feature:

$$W_{c2t}(i,j) = \frac{\exp\left(\left(c_i \cdot q_i^t\right)\cdot\left(x_j^t \cdot k_j^t\right)^\top\right)}{\sum_{j=1}^{T}\exp\left(\left(c_i \cdot q_i^t\right)\cdot\left(x_j^t \cdot k_j^t\right)^\top\right)} \quad (14)$$

We then fuse these two attention maps using a dot product operation:

$$W = W_{c2f}^\top \cdot W_{c2t} \quad (15)$$

Finally, the updated audio feature x' is computed by applying the fused attention weights W to the value feature, followed by a residual connection (Eq. (12)). This mechanism effectively aligns temporal and spectral audio features with textual conditions, ensuring fine-grained text details are accurately reflected in the generated audio.

4 Experiment Setup

Dataset. We used only the AudioCaps dataset [26] for training and testing. We used approximately 49,000 clips for training and 964 clips for testing. Each sample is 10 seconds long with a 16 kHz sampling rate and was converted to mel-spectrograms (64 mel channels, 1024-sample window, 160-sample hop length) for training.

Training Details. The generator and discriminator learning rates were set to 0.0001 and 0.0004, respectively [27]. The discriminator was updated twice per generator update. We use Adam optimizer [28] with $\beta_1 = 0.5$ and $\beta_2 = 0.999$ for training. For all contrastive losses, τ was set to 0.1, and for word contrastive loss, γ_1, γ_2, and γ_3 were set to 5, 5, and 50 [29], respectively.

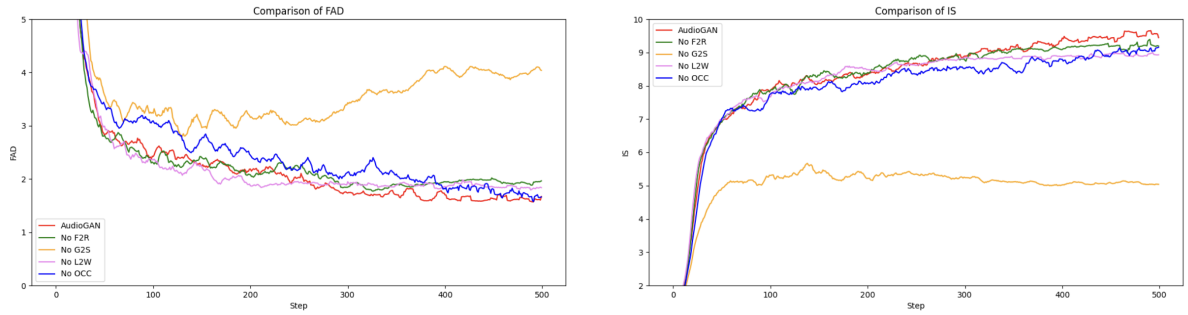


Fig. 6: Impact of removing the specific contrastive loss.

Evaluation Metrics. We evaluated our generated audio using five key metrics: Frechet Audio Distance (FAD) [30], Frechet Distance (FD) [31], Inception Score (IS) [32], Kullback-Leibler Divergence (KL) [33], and CLAP Score [15]. FAD compares the distribution of features from generated and real audio using the VGGish [34] model, while FD does the same using the PANN model [35]. IS measures the quality and diversity of generated samples. KL divergence quantifies the difference between the distributions of real and generated audio based on a pre-trained classifier's output. Finally, the CLAP score assesses how well the generated audio aligns with the textual prompt. These metrics together provide a comprehensive evaluation of our model.

5 Results

5.1 Comparison with Existing Methods

Recent advances in TTA generation have been significantly advanced by diffusion-based models, such as AudioLDM [2, 6] and Tango [4, 7], which rely on large U-Net architectures [36] to iteratively denoise latent representations. While these methods can achieve impressive audio quality with a sufficient number of denoising steps, they face a fundamental trade-off: increasing the number of steps improves fidelity but significantly escalates computational cost and inference time, whereas reducing the steps lowers computational cost and inference time but compromises audio quality. This limitation poses a major challenge for real-time applications such as interactive audio generation and on-the-fly sound design, where efficiency and responsiveness are critical.

Table 2: Ablation results with different contrastive losses.

Method	G2S	L2W	F2R	OCC	FAD(↓)	FD(↓)	IS(↑)
No G2S		✓	✓	✓	2.87	23.65	5.7
No L2W	✓		✓	✓	1.73	11.88	8.86
No F2R	✓	✓		✓	1.59	13.27	9.11
No OCC	✓	✓	✓		1.82	13.57	9.16
AudioGAN	✓	✓	✓	✓	1.38	12.66	9.63

In contrast, our method, AudioGAN, employs a GANs-based strategy for TTA. Instead of relying on an iterative inference process, AudioGAN generates audio in a single pass, significantly improving efficiency and reducing computational costs. In Table 1, we compare our model with existing methods on the AudioCaps test dataset, keeping the experimental conditions. Because generative models can produce slightly varied outputs from the same prompt, we averaged the results over three runs per approach. The results show that AudioGAN achieves state-of-the-art performance in key metrics, notably FAD and IS, and remains competitive in FD. The notably higher IS score indicates not only superior audio quality but also broader output diversity. Additionally, we measured the number of model parameters and GPU memory usage during inference. AudioGAN stands out with its compact architecture, significantly reducing both model size and memory requirements. As illustrated in Figure 1, while diffusion-based models improve performance with more diffusion steps at the cost of longer inference times, AudioGAN completes inference in under one second. These results highlight the advantages of our GANs-based strategy, which achieves competitive performance while remaining practical for real-time and resource-constrained TTA applications.

Table 3: The ablation study of various attention.

Attention					
First	Second	Third	FAD(↓)	FD(↓)	IS(↑)
self	self	self	1.6	13.31	9.08
word	word	word	1.73	13.0	8.8
sentence	sentence	sentence	2.0	12.53	9.01
sentence	word, sentence	self, word, sentence	1.38	12.66	9.63

5.2 Ablation Study

To verify the effectiveness of our proposed training techniques, we conducted comprehensive ablation studies targeting key components of our GANs training framework. First, we evaluated the contribution of each contrastive loss by removing one loss term at a time from the overall training objective and analyzing its impact on training stability and performance. Figure 6 and Table 2 show that excluding specific contrastive loss resulted in measurable changes in both stability and performance. Notably, the global-to-sentence (G2S) contrastive loss—which aligns the global audio features with the sentence-level text features extracted from the CLAP model—had the most pronounced impact. When the G2S loss was removed, the training became considerably unstable, and overall performance degraded significantly. This finding demonstrates that textual features aligned with auditory features are essential for effectively guiding model training in text-to-audio generation. Furthermore, our experiments show that incorporating the oneself-conditioned contrastive (OCC) loss led to a substantial improvement in the Inception Score (IS), indicating its effectiveness in promoting data diversity and enhancing the overall quality of generated audio. Collectively, these results highlight the critical role of each contrastive loss, ensuring stable training and improved synthesis performance.

We proposed a single-double-triple (SDT) attention that combines three different types of attention mechanisms (self, word, and sentence attention). To assess the individual impact of different attention mechanisms, we conducted experiments for the models trained using only one type of attention (Table 3). The results show that each individual attention mechanism provides decent performance, with self-attention yielding slightly higher results; however, self-attention incurs higher computational costs as feature sizes grow, compared to other methods. In contrast, SDT attention, which integrates all three mechanisms in a structured and balanced manner, enhances overall performance while mitigating the computational drawbacks of self-attention.

Table 4: The ablation study of Time-Frequency Cross Attention (TF-CA).

TF-CA		FAD(↓)	FD(↓)	IS(↑)
Self	Multi			
		1.76	13.82	8.62
✓		1.67	13.68	8.86
	✓	1.86	14.42	9.05
✓	✓	1.38	12.66	9.63

We also acknowledge that numerous alternative combinations are possible, and exploring these variations remains a promising direction for future work.

Finally, to assess the effectiveness of our time-frequency cross-attention (TF-CA) mechanism, we conducted ablation studies comparing two variants: self TF-CA and multi TF-CA (Table 4). Self TF-CA was applied to the self-attention, and multi TF-CA was applied to the word and sentence attention, respectively. Our results show that incorporating self TF-CA yields modest performance gains over the baseline model without TF-CA. When multi TF-CA is used, both FAD and FD scores improve slightly, while the IS score increases modestly. Notably, combining both variants leads to even greater improvements. These results demonstrate that our TF-CA mechanism effectively captures the unique temporal and spectral characteristics of audio, thereby enhancing both the quality and diversity of the generated outputs.

6 Conclusion

This paper propose the first successful attempt at using a GANs-based framework for TTA generation. Our approach, AudioGAN, overcomes the inherent challenges of diffusion-based methods—such as large model sizes, slow inference, and high computational costs—by generating audio in a single pass with fewer parameters in under one second. To achieve this, we incorporate multiple contrastive losses and introduce novel methods, namely Single-Double-Triple (SDT) Attention and Time-Frequency Cross-Attention (TF-CA). Extensive experiments on the AudioCaps dataset demonstrate that AudioGAN not only achieves state-of-the-art performance but also significantly reduces model parameters and GPU memory usage. These results highlight the advantages of our GANs-based approach in generating high-fidelity and diverse audio while maintaining computational efficiency, thereby paving the way for real-time applications and advancing the field of TTA.

References

[1] Yang, D., Yu, J., Wang, H., Wang, W., Weng, C., Zou, Y., and Yu, D., "Diffsound: Discrete diffusion model for text-to-sound generation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 31, pp. 1720–1733, 2023.

[2] Liu, H., Chen, Z., Yuan, Y., Mei, X., Liu, X., Mandic, D., Wang, W., and Plumbley, M. D., "Audioldm: Text-to-audio generation with latent diffusion models," *arXiv preprint arXiv:2301.12503*, 2023.

[3] Huang, R., Huang, J., Yang, D., Ren, Y., Liu, L., Li, M., Ye, Z., Liu, J., Yin, X., and Zhao, Z., "Make-an-audio: Text-to-audio generation with prompt-enhanced diffusion models," in *International Conference on Machine Learning*, pp. 13916–13932, PMLR, 2023.

[4] Ghosal, D., Majumder, N., Mehrish, A., and Poria, S., "Text-to-audio generation using instruction guided latent diffusion model," in *Proceedings of the 31st ACM International Conference on Multimedia*, pp. 3590–3598, 2023.

[5] Huang, J., Ren, Y., Huang, R., Yang, D., Ye, Z., Zhang, C., Liu, J., Yin, X., Ma, Z., and Zhao, Z., "Make-an-audio 2: Temporal-enhanced text-to-audio generation," *arXiv preprint arXiv:2305.18474*, 2023.

[6] Liu, H., Yuan, Y., Liu, X., Mei, X., Kong, Q., Tian, Q., Wang, Y., Wang, W., Wang, Y., and Plumbley, M. D., "Audioldm 2: Learning holistic audio generation with self-supervised pretraining," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2024.

[7] Majumder, N., Hung, C.-Y., Ghosal, D., Hsu, W.-N., Mihalcea, R., and Poria, S., "Tango 2: Aligning diffusion-based text-to-audio generations through direct preference optimization," *arXiv preprint arXiv:2404.09956*, 2024.

[8] Kreuk, F., Synnaeve, G., Polyak, A., Singer, U., Défossez, A., Copet, J., Parikh, D., Taigman, Y., and Adi, Y., "Audiogen: Textually guided audio generation," *arXiv preprint arXiv:2209.15352*, 2022.

[9] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y., "Generative adversarial nets," *Advances in neural information processing systems*, 27, 2014.

[10] Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B., "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

[11] Kingma, D. P., "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[12] Song, J., Meng, C., and Ermon, S., "Denoising diffusion implicit models," *arXiv preprint arXiv:2010.02502*, 2020.

[13] Arora, K., Asri, L. E., Bahuleyan, H., and Cheung, J. C. K., "Why exposure bias matters: An imitation learning perspective of error accumulation in language generation," *arXiv preprint arXiv:2204.01171*, 2022.

[14] Vaswani, A., "Attention is all you need," *Advances in Neural Information Processing Systems*, 2017.

[15] Wu, Y., Chen, K., Zhang, T., Hui, Y., Berg-Kirkpatrick, T., and Dubnov, S., "Large-scale contrastive language-audio pretraining with feature fusion and keyword-to-caption augmentation," in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5, IEEE, 2023.

[16] Kong, J., Kim, J., and Bae, J., "Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis," *Advances in neural information processing systems*, 33, pp. 17022–17033, 2020.

[17] Lim, J. H. and Ye, J. C., "Geometric gan," *arXiv preprint arXiv:1705.02894*, 2017.

[18] Zhang, H., Koh, J. Y., Baldridge, J., Lee, H., and Yang, Y., "Cross-modal contrastive learning for text-to-image generation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 833–842, 2021.

[19] Chung, H., Lee, Y., and Jung, J., "FOLEY SOUND SYNTHESIS BASED ON GENERATIVE ADVERSARIAL NETWORKS USING ONESELF-CONDITIONED CONTRASTIVE LEARNING," ????

[20] Bahdanau, D., "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[21] Wang, X., Girshick, R., Gupta, A., and He, K., "Non-local neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7794–7803, 2018.

[22] Dong, L., Xu, S., and Xu, B., "Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition," in *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 5884–5888, IEEE, 2018.

[23] Tang, C., Luo, C., Zhao, Z., Xie, W., and Zeng, W., "Joint time-frequency and time domain learning for speech enhancement," in *Proceedings of the twenty-ninth international conference on international joint conferences on artificial intelligence*, pp. 3816–3822, 2021.

[24] Mu, W., Yin, B., Huang, X., Xu, J., and Du, Z., "Environmental sound classification using temporal-frequency attention based convolutional neural network," *Scientific Reports*, 11(1), p. 21552, 2021.

[25] Zhao, Y. and Wang, D., "Noisy-Reverberant Speech Enhancement Using DenseUNet with Time-Frequency Attention." in *Interspeech*, volume 2020, pp. 3261–3265, 2020.

[26] Kim, C. D., Kim, B., Lee, H., and Kim, G., "Audiocaps: Generating captions for audios in the wild," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 119–132, 2019.

[27] Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S., "Gans trained by a two time-scale update rule converge to a local nash equilibrium," *Advances in neural information processing systems*, 30, 2017.

[28] Kingma, D. P. and Ba, J., "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[29] Xu, T., Zhang, P., Huang, Q., Zhang, H., Gan, Z., Huang, X., and He, X., "Attngan: Fine-grained text to image generation with attentional generative adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1316–1324, 2018.

[30] Kilgour, K., Zuluaga, M., Roblek, D., and Sharifi, M., "Fréchet Audio Distance: A Reference-Free Metric for Evaluating Music Enhancement Algorithms." in *INTERSPEECH*, pp. 2350–2354, 2019.

[31] Tailleur, M., Lee, J., Lagrange, M., Choi, K., Heller, L. M., Imoto, K., and Okamoto, Y., "Correlation of Fr\'echet Audio Distance With Human Perception of Environmental Audio Is Embedding Dependant," *arXiv preprint arXiv:2403.17508*, 2024.

[32] Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X., "Improved techniques for training gans," *Advances in neural information processing systems*, 29, 2016.

[33] Iashin, V. and Rahtu, E., "Taming visually guided sound generation," *arXiv preprint arXiv:2110.08791*, 2021.

[34] Hershey, S., Chaudhuri, S., Ellis, D. P., Gemmeke, J. F., Jansen, A., Moore, R. C., Plakal, M., Platt, D., Saurous, R. A., Seybold, B., et al., "CNN architectures for large-scale audio classification," in *2017 ieee international conference on acoustics, speech and signal processing (icassp)*, pp. 131–135, IEEE, 2017.

[35] Kong, Q., Cao, Y., Iqbal, T., Wang, Y., Wang, W., and Plumbley, M. D., "Panns: Large-scale pretrained audio neural networks for audio pattern recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28, pp. 2880–2894, 2020.

[36] Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B., "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Audio Engineering Society

Conference Paper 18

Presented at the AES International Conference on Machine Learning and Artificial Intelligence for Audio
2025 September 8–10, London, UK

This paper was peer-reviewed as a complete manuscript for presentation at this conference. This paper is available in the AES E-Library (http://www.aes.org/e-lib), all rights reserved. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Journal of the Audio Engineering Society.

Predicting binaural colouration using VGGish embeddings

Thomas McKenzie[1], Alec Wright[1], Daniel Turner[2], and Pedro Lladó[3]

[1]*Acoustics and Audio Group, Reid School of Music, University of Edinburgh, United Kingdom*
[2]*Vision, Learning, and Control Group, School of Electronics and Computer Science, University of Southampton, United Kingdom*
[3]*Institute of Sound Recording, University of Surrey, United Kingdom*

Correspondence should be addressed to Thomas McKenzie (thomas.mckenzie@ed.ac.uk)

ABSTRACT

An initial feasibility study is presented exploring the use of a pre-trained feature extractor, designed for large-scale audio classification, applied to the task of predicting the colouration between binaural signals. A multilayer perceptron (MLP) is trained to predict binaural colouration using feature embeddings obtained from the VGGish network and data from five previously conducted listening tests. The evaluation compares seven versions of the network, each trained using different data augmentation methods, to three existing signal processing methods: basic spectral difference (BSD), log. spectral distance (LSD) and an auditory model for predicting binaural colouration (PBC-2). Results show that while the MLP networks are comparable to BSD and LSD, further work is needed to compete with the more accurate PBC-2; such as using specific audio features relevant for colouration.

1 Introduction

Colouration is a perceptual attribute that is central to spatial audio quality evaluation, and often considered more important than localisation [1]. As with any perceptual attribute, the most accurate way to measure it is through controlled listening experiments. However, numerical assessment methods that correctly estimate the perceived colouration are desirable, since they enable quick monitoring of the introduced colouration during development phases of spatial audio systems [2, 3, 4].

A recent study [5] proposed an auditory model that showed strong correlation between listening test data and the model predictions. This model follows a traditional signal processing approach combining signal rectification, high frequency smoothing, equivalent rectangular bandwidth frequency weighting, and signal-dependent weighting of the left and right colouration values to produce a single binaural colouration prediction. A Matlab implementation of the resulting auditory model was made available as part of the Auditory Modeling Toolbox under the model name *mckenzie2025* [5].

To the best of the authors' knowledge, no data-driven model of colouration has yet been explored. Recently, data-driven models have been proposed for functions of auditory processing, often with promising results. These models are either trained using listening experiment data [6] or with true objective data [7, 8, 9]. In many cases, the models can capture nuances of auditory processing that are not fully understood from a psychophysical or neurophysiological point of view,

generalising to unseen conditions and outperforming traditional models in certain tasks. A third strategy is to train data-driven models with estimated data using existing auditory models [10, 11]. However, while synthetic training data is easy to obtain in these cases, and while the resulting models may run significantly faster, their performance is not expected to exceed that of the original models.

This study presents an initial exploration of a data-driven method as an alternative to the auditory model introduced in [5]. The method presented here adopts a pre-trained feature extractor [12] designed for large-scale audio classification, whose feature embeddings are used to train a multilayer perceptron (MLP) regression model. Since the long-term goal is to obtain better results than the ones obtained using [5], the model is trained with listening experiment data as labels. Within the context of the limited colouration data from listening experiments and aiming to optimise the amount of information obtained from them, feature extraction and data augmentation techniques are evaluated for their effect on the model predictions. The data-driven methods explored here are compared against existing colouration metrics obtained using signal processing approaches.

The paper is organised as follows. Section 2 presents a methodology for predicting binaural colouration, whereby a pre-trained feature extractor is used to train MLP regression models and the data augmentation methods are detailed. Section 3 then presents an evaluation of the proposed methods against existing models on their prediction of colouration using a previously unseen subset of signals from the listening tests on perceived spectral similarity. The results of the evaluation are discussed in Section 4, identifying the models that perform the strongest and suggesting why, as well as discussing limitations of the current methodology. Finally, the paper is concluded in Section 5 along with future work.

2 Methods

This section describes the methodology of this study, including the pre-trained VGGish feature extraction model, the multilayer perceptron (MLP) regression model and the training data, along with details of the tested data augmentation methods. All audio signals used in this study are binaural with a sample rate of 48 kHz and 16-bit depth. Fig. 1 presents a block diagram illustrating the training pipeline of the MLP model.

2.1 Listening test data

The dataset used consists of 252 binaural signals, featuring different audio contents and colouration types from five previously conducted listening tests [13, 14, 15, 4, 16]. The number of signals, their audio contents and their colouration types, along with the number of participants in each listening test, are summarised in Table 1.

All listening tests followed a paradigm similar to the multiple stimulus test with hidden reference and anchor (MUSHRA [17]), whereby the similarity between a reference and multiple test signals was rated for the quality of *colouration* on a scale from 0 to 100, whereby a rating of 100 means entirely the same (no colouration). In this study, perceived colouration is therefore derived as the reversed ratings of perceived similarity (where 0 indicates entirely the same, and 100 means as coloured as the low anchor). A single result for each stimulus is taken as the mean of all participants individual ratings.

For each trial in all five listening tests, the reference stimulus was a convolution of the monophonic audio content (see again Table 1) with head-related impulse responses (HRIRs), and anchors were a low-pass filtered version of the reference with a cut-off frequency $f_o = 3.5$ kHz. Test stimuli were presented statically (with a fixed head orientation) over open-back headphones in quiet listening rooms. While the majority of stimuli are simple scenes (single stationary sound sources at a single location), some were complex scenes utilising a mix of multiple sound sources located at different locations on the sphere [15, 16]. The signals with pink noise audio content were around one second in duration, whereas some of the complex scenes (e.g. the pink noise bursts with a train station recording and the percussion recordings) were up to 8 seconds in duration.

The listening test data was randomised prior to being divided into subsets for training, validation and testing, to try and ensure an equal representation of colouration scores within each set. The randomisation used a fixed seed to ensure reproducibility between experiments. The data was then split into training, validation and test subsets, with a split of $60\% : 20\% : 20\%$, meaning a total of 152 binaural signals reserved for training, 50 for validation and 50 for testing.

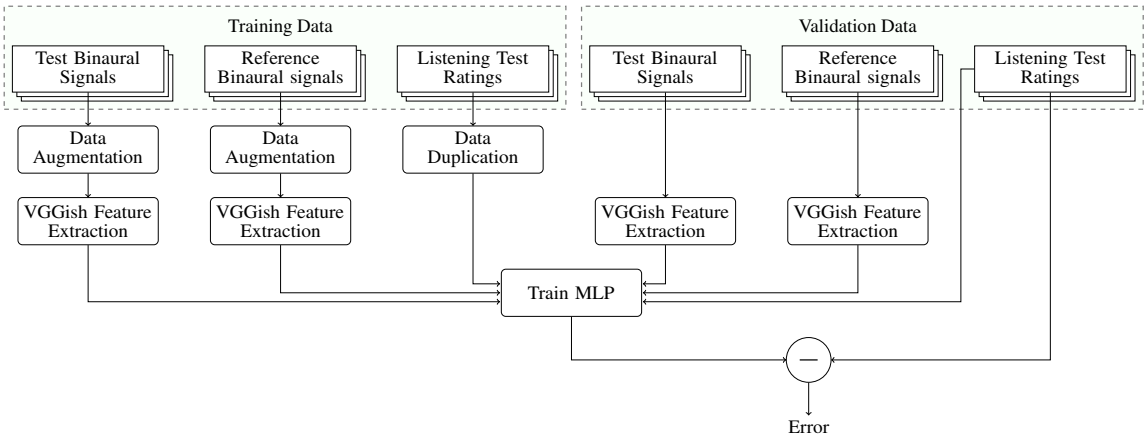


Fig. 1: Block diagram summary of the training of the multilayer perceptron (MLP) models.

Table 1: Brief details of the listening test data used in this study.

Test	# Stimuli	Audio content	Colouration type	# Participants
Mc18 [13]	48	Pink noise	Equalised Binaural Ambisonic HRIRs	20
Mc19a 1 [14]	42	Pink noise	Equalised Binaural Ambisonic HRIRs	20
Mc19a 2 [14]	21	Pink noise bursts and train station recording	Equalised Binaural Ambisonic HRIRs	20
Mc19b 1 [15]	15	Pink noise	Pre-processed Binaural Ambisonic HRIRs	20
Mc19b 2 [15]	15	Percussion recordings	Pre-processed Binaural Ambisonic HRIRs	20
Mc22 [4]	56	Pink noise	10 band equalised HRIRs	9
Ll22 1 [16]	25	Pink noise	HRIRs made wearing different headphones	15
Ll22 2 [16]	25	Speech recordings	HRIRs made wearing different headphones	15
Ll22 3 [16]	5	Rainfall recordings	HRIRs made wearing different headphones	15

2.2 Feature extraction

Given the limited available data, a pre-trained feature extractor was employed to leverage the compact representations learned from a much larger corpus of general audio data. This allows the MLP regression model to be trained on a meaningful, but lower dimensionality, representation of the dataset. The feature extraction mechanism aims at reducing the number of input features into the MLP while maintaining as much relevant information as possible.

In this study, the neural network model detailed in Section 2.3 was trained using features extracted using the VGGish network [12]. VGGish, a modified version of VGG16 architecture, is an 11-layer convolutional neural network (CNN) that has been pre-trained on audio data from the YouTube-8M dataset [18], which incorporates log-mel spectrogram-based inputs for feature extraction. The VGGish model produces 128 features per test signal for each second of audio in a single audio channel. For audio signals that are longer than 1 second, only the first 128 features are used in this study. This is equivalent to truncating the signal to one second, and is done to ensure all signal durations produce results with the same dimensions. It should be adequate, as colouration in the used listening test data is assumed to be approximately constant over the entire signal. As all audio signals are binaural, the left and right channel features are concatenated, resulting in 256 features per binaural signal.

2.3 Model

An MLP was trained to predict binaural colouration using the features extracted using the VGGish extractor as input. A block diagram illustration of the model is

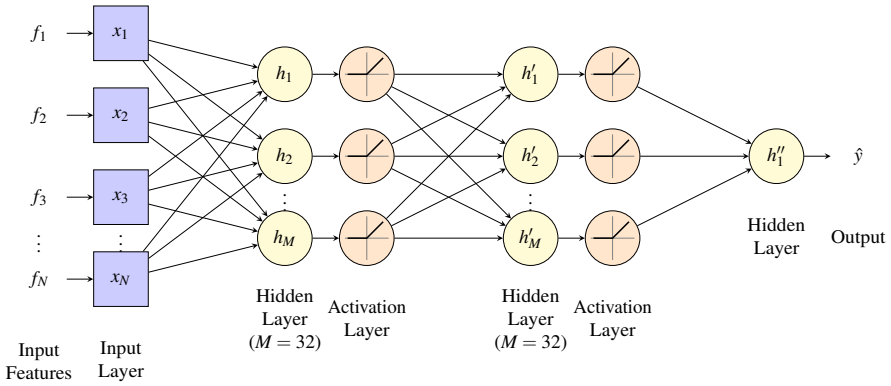


Fig. 2: Structure of the multilayer perceptron (MLP) regression model used in this study to predict binaural colouration.

presented in Fig. 2. The model consists of a feature input layer, the size of which is equal to the number of features outputted by the feature extractor, which in this study was $N = 256$. The feature input layer is followed by two fully connected hidden layers, both of size $M = 32$, and the final output layer, which consists of a single unit to predict the final colouration value. A Rectified Linear Unit (ReLU) activation layer is also applied to the output of each hidden layer.

The model is optimised using mini-batch Stochastic Gradient Descent with Momentum (SGDM), with the mean-squared-error (MSE) between the predicted and ground truth colouration values as the loss function to be minimised. The initial learning rate is 5×10^{-4}, the maximum number of epochs set to 200, and the batch size is set to 32. The validation loss is calculated every 20 epochs, with a validation patience of 10 to allow early stopping. The model with the best validation loss is the one taken forward to be evaluated on the test set.

2.4 Data augmentation

Due to the limited available training data, two data augmentation methods were explored to investigate their potential to improve the performance of the model through effectively increasing the amount of training data. The results and effects on the validation set are reported here.

Many established methods for audio data augmentation either directly or indirectly alter the spectral content of the data [19]. As such, these methods are not suitable for this study as they would likely cause a change in perceived colouration and thus invalidate the associated ground truth value. Therefore, the options available were limited and only temporal changes were employed. The training data was duplicated and augmented, whereas the listening test results were simply duplicated.

The first augmentation method is a time-reversal of the signals (herein referred to as 'reverse'). The discreet time signal $x(n)$, where n denotes the sample, is reversed by

$$x_{\text{rev}}(n) = x(N - n) \tag{1}$$

where N is the total number of samples in the signal.

The second augmentation method is a time-rotation of the signals (herein referred to as 'circshift', whereby the beginning of the signal is moved to the end and the end is brought forward. A signal $x(n)$ processed with circshift[M] becomes

$$x_{\text{circshift}[m]}(n) = x(\frac{mN}{M+1} + n \mod (N)) \tag{2}$$

where m is the index of the duplication and M is the total number of duplications. For example, one duplication of data (denoted here as circshift[1]) would mean the first half of the signal is swapped in position with the second half of the signal. For two duplications (circshift[2]), the first duplication moves first third to the end while the second two thirds are moved to the start; the second duplication moves the first two thirds to the end while the last third is moved to the start. While 'reverse' can only augment the data once, 'circshift' allows for many more augmentations of the data.

Table 2: Results of initial model training using different data augmentation methods. $\overline{\text{RMSE}}$ and STD denote the mean and standard deviation of the root-mean-square error between the ten repeats, respectively.

	Training		Validation	
Augmentation	$\overline{\text{RMSE}}$	STD	$\overline{\text{RMSE}}$	STD
none	8.51	3.9	13	0.867
reverse	8.66	2.78	13.3	0.849
circshift[1]	8.69	3.21	12.5	1.15
circshift[2]	10.1	2.97	12.2	0.586
circshift[3]	8.03	1.26	11	0.528
circshift[4]	8.77	3	11.5	0.994
circshift[5]	7.77	1.42	10.5	0.56

The configurations of augmentation methods employed in this study are summarised as follows:

- none: No data augmentation. Training data dimensions [152,256]
- reverse: Training data signals duplicated with signals time-reversed. Training data dimensions [304,256]
- circshift[1-5]: Training data signals duplicated with signals time-shifted. Training data dimensions [304,256], [456,256], [608,256], [760,256] and [912,256], respectively

To evaluate the effects of the different data augmentation methods, an MLP was trained ten times for each. To ensure repeatability, the random seed was fixed for each iteration, such that the first repeat with each augmentation method would use the same seed, and the next repeat would use a different common seed. Table 2 presents the initial results on the training and validation data for the models trained using seven training datasets: one of which is the original without data augmentation ('none'), and six with data augmentation. Two metrics are reported: the mean ($\overline{\text{RMSE}}$) and the standard deviation (STD) between the root-mean-square error (RMSE) of the ten repeats.

The first observation of these initial results is that overfitting appears evident from the lower RMSE values for training than validation (see Table 2). This is likely due to the small amount of training data. Secondly, the standard deviation between RMSE for each repeat of the training is far higher than that for validation. A likely explanation for this was found by observing that in some training repeats, the loss would briefly jump, and never recover to the values found in other repeats. It may also be a result of there being less data in the validation set.

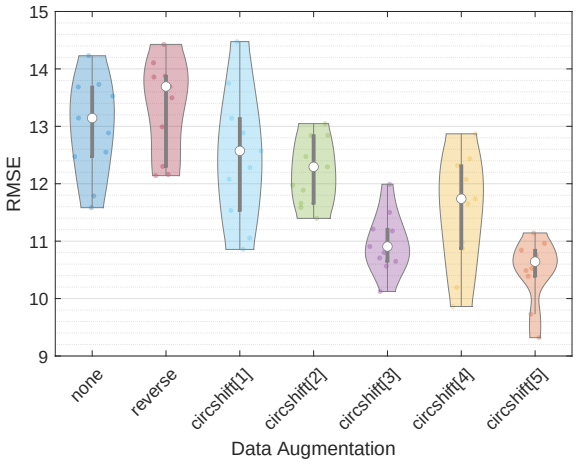


Fig. 3: Root-mean-square error (RMSE) between the validation results with different data augmentation options pre-feature extraction, for 10 repeats of training the model with different random seeds.

To look in more detail at the effect of augmentation methods on the validation set, Fig. 3 presents violin plots of the validation RMSE. Here it seems that the first augmentation method, reverse, does not generally appear to improve the validation loss. The more promising results seem to be as the number of signals in the training dataset increases significantly in size (e.g. for circshift[3-5]): here the validation loss is significantly lower than for no data augmentation. These results helped to inform the evaluation in the following section.

3 Evaluation

To evaluate the final models, the mean listening test ratings (perceived colouration) were compared to predicted colouration values made using the trained neural networks with the as-before-unseen test subset of the data. For an additional comparison to the current state-of-the-art, the results from the proposed neural

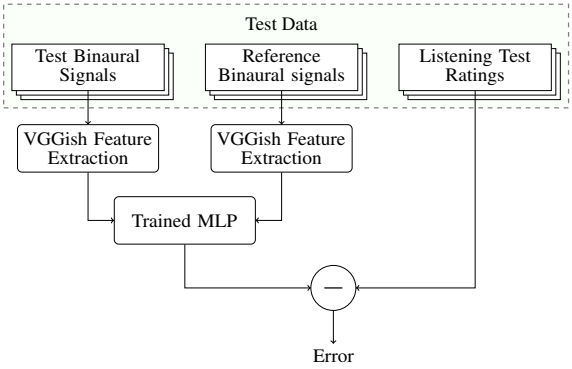


Fig. 4: Block diagram summary of the testing of the trained MLPs.

networks are also compared to three signal processing-based binaural colouration methods. These are basic spectral difference (BSD), log spectral distance (LSD) and PBC-2 (*mckenzie2025* in the auditory modelling toolbox):

- BSD: mean absolute difference in dB between the reference and test spectra
- LSD: Root-mean-square difference in dB between the reference and test spectra
- PBC-2: Auditory model with features including signal rectification, high frequency smoothing, equivalent rectangular bandwidth frequency weighting, binaural weighting

Fig. 4 and Fig. 5 present block diagrams for the evaluation stage using the trained MLP and the PBC-2 auditory model, respectively. This illustrates the differences between the two methodologies.

While the MLP networks have been trained to produce colouration predictions within the same range as the listening test results, the raw BSD and LSD predictions require transformation to fit a range of 0-100. This was done using the linear regression coefficients as calculated in [5], with respective slope and intercept values of $a = 4.803$ and $b = 13.379$ for BSD, and $a = 4.101$ and $b = 10.252$ for LSD. The PBC-2 model already transforms outputs to fit within this range.

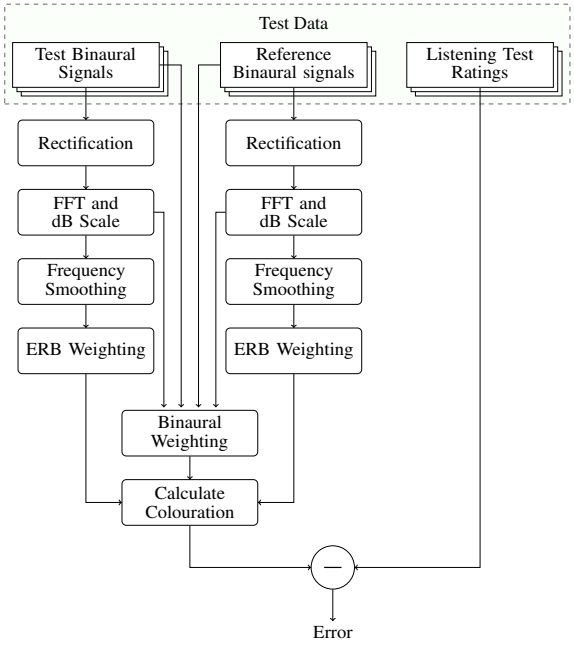


Fig. 5: Block diagram summary of the PBC-2 auditory model [5] and its use in this study.

3.1 Results

The predictions are analysed using a linear regression fitting; the results of which are presented in Table 3. Three measures are reported: R^2_{adj}, the proportion of explained variance (the squared correlation coefficient but corrected for the number of predictors, which is one in this case); MxAE, the maximum absolute error between the ratings and predictions; and RMSE, the root-mean-square error between the ratings and predictions. Note: more accurate predictions of the listening test results are indicated by higher R^2_{adj} values, and lower MxAE and RMSE values.

Firstly, all tested models produce a positive correlation between the predicted and perceived colouration values, suggesting that VGGish features can be used to predict binaural colouration. The range in R^2_{adj} values show that all variants of the MLP networks perform somewhat comparably to the simple signal-processing colouration methods BSD and LSD. However, they are all significantly worse than the more tailored PBC-2. The best performing network was the one with circ-shift[4] training data augmentation, which achieved an $R^2_{\text{adj}} = 0.639$. This demonstrates a notable improvement on BSD and LSD.

Table 3: Results from the linear regression between the predicted and perceived colouration, of the neural networks (top) and signal processing colouration methods (bottom). $R^2_{\text{adj.}}$, MxAE and RMSE denote adjusted explained variance, maximum absolute error and root-mean-square error, respectively.

Processing	$R^2_{\text{adj.}}$	MxAE	RMSE
none	0.549	28.87	12.9
reverse	0.535	36.39	13.1
circshift[1]	0.502	39.83	13.88
circshift[2]	0.531	33.88	13.3
circshift[3]	0.445	37.62	14.38
circshift[4]	0.639	29.59	11.45
circshift[5]	0.541	33.82	13.08
BSD	0.557	34	12.92
LSD	0.521	37.3	13.34
PBC-2	0.901	16.3	6.164

To further understand the results of the regression analysis and how the different model iterations compare, Fig. 6 illustrates the predicted versus perceived colouration and the prediction error. An interesting observation can be made by comparing the predicted error plots. While BSD and LSD seem to significantly overestimate colouration values below 20, the MLP networks and PBC-2 appear somewhat less susceptible to this trend.

4 Discussion

The results show that feature embeddings from VGGish demonstrate a certain degree of generalisability and can be applied to specific tasks, such as predicting binaural colouration. Its performance was comparable to baseline signal processing methods, such as BSD and LSD, with some of the tested feature augmentation and distillation techniques enabling it to outperform these approaches. Generally, the models that better correlated with the listening test results also had lower maximum and average errors. However, even the best-performing MLP configurations performed significantly poorer than the auditory model PBC-2.

A key limitation observed in the results is the likely presence of significant overfitting. This is evident from the pattern of RMSE values across training, validation, and test sets, which typically increased from the training to validation to test set. Notably, models that exhibited consistently lower training and validation loss, such as those employing circshift[3] and circshift[5], produced higher errors for the unseen test set. To address this, future development should employ procedures such as K-fold cross-validation to obtain a more accurate measure of general model performance.

One of the primary constraints affecting model performance is the limited availability of data. The small dataset likely contributed to overfitting and played a significant role in the diminished performance observed on the test set. This suggests that the models were not generalising well beyond the training and validation data. Addressing this limitation would require access to a more extensive dataset, which would be costly and time consuming to acquire.

Additionally, what data there is is still somewhat limited in terms of content and colouration type. Almost 75% of sounds use pink noise as the stimulus. A greater variety is needed with more music and speech, and a greater range of both transient and periodic sounds. Over 50% are binaural renders of Ambisonic signals. Whilst the processing on these signals differs, from equalisation to different order-dependent colouration to HRIR pre-processing steps, more variation in types of colouration should be explored such as between HRIRs from different heads and degradations from perceptual coding algorithms (e.g. lossy data compression). More than 80% are single-source stationary anechoic signals. Reverberant signals, multiple-source complex scenes, moving sources, and varying source widths should be better represented in the data. Finally, the listening tests used in this study primarily featured male Caucasian participants between the ages 20 and 40. This is not representative of the general population and should be addressed, as models trained on the current data will be biased.

With regard to data augmentation, the current approach involved duplicating listening test results. A potential refinement would be to replace the duplication of the mean across all participants and repetitions with a leave-one-out strategy, introducing variation into the duplicated mean perceptual values. Other audio augmentation methods that do not affect colouration should continue to be explored too, as it is likely that even a larger dataset of listening experiment data will be insufficient in many cases.

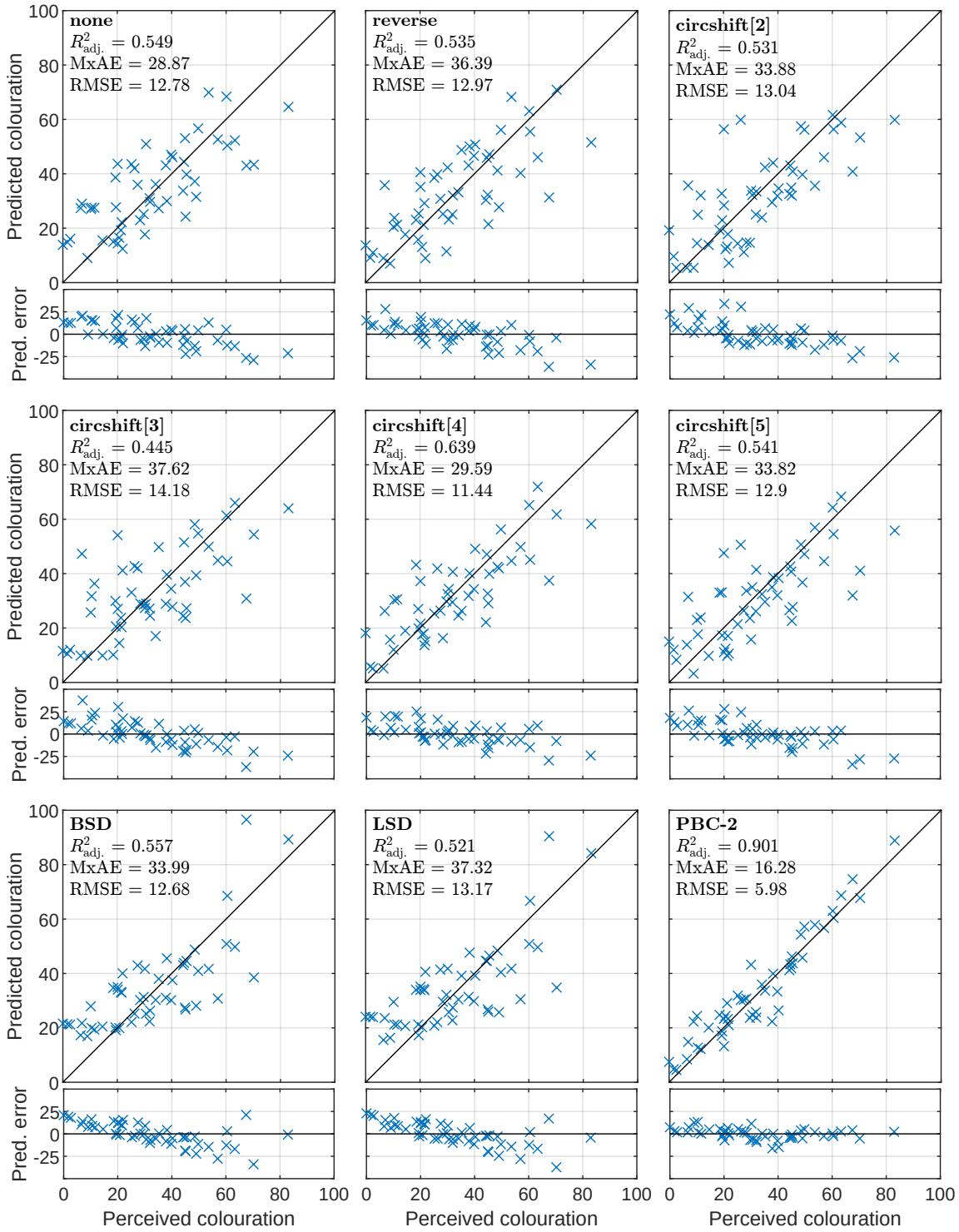


Fig. 6: Model performance by means of the predicted versus rated colouration and the prediction error (pred. error) for each model. $R^2_{\text{adj.}}$, MxAE and RMSE denote adjusted explained variance, maximum absolute error and root-mean-square error, respectively. Note: circshift[1] omitted for readability.

Another point of discussion is the suitability of the used feature extractor. The VGGish network used in this study was originally trained at a sample rate of 16 kHz on a dataset that may not be directly relevant to the task at hand (YouTube audio clips). As many of the important spectral cues in binaural signals are present at high frequencies (above 5 kHz), and a 16 kHz sample rate will have a nyquist frequency of 8 kHz, it is possible that many important spectral features will not be represented in the extracted VGGish features. Given these constraints, it is somewhat surprising that VGGish was able to generalise at all. Future developments will explore alternative feature extraction methods that are more specifically suited to binaural colouration. At a minimum, incorporating some form of auditory front-end modeling prior to feature extraction could be beneficial. Alternatively, leveraging the PBC-2 model in conjunction with a neural network could be investigated.

Finally, other avenues for improving the current methodology include hyperparameter optimisation in the MLP training process. Fine-tuning the model parameters could enhance performance and mitigate some of the observed limitations. However, this is expected to offer minor improvements when compared to the other more significant developments such as a larger and more diverse dataset, greater and more effective data augmentation methods, and more binaural colouration-specific feature extraction methods.

5 Summary

This paper has presented an exploratory study into the use of shallow neural networks in predicting the colouration between binaural signals, utilising feature embeddings obtained from a VGGish pre-trained convolutional neural network for large-scale audio classification. Seven versions of the proposed model with different data augmentation methods have been tested and compared to three signal processing methods for predicting binaural colouration. Results show that the MLP networks offer comparable, and in some cases improved, performance to the simple signal processing methods BSD and LSD, but fall short of the more complex PBC-2.

Although the results of this study were not close to the state-of-the-art PBC-2 model, this approach shows promise. A number of avenues can be explored to improve the results. First and foremost is the need for a larger dataset with more diverse signals in areas such as: complexity of scene (number of sources), reverberance, colouration type, sound source content (transient and periodic, wide-band and narrow-band), among others. Secondly, other features could be explored: while VGGish does work, using features specific to colouration may be more appropriate, such as those produced by the PBC-2 model's peripheral processing stages. Additional data augmentation and hyperparameter optimisation may serve to improve model performance with the given data. Finally, future work to reduce the overfitting will only serve to improve the overall generalisability of the models.

6 Acknowledgements

This work was supported by the UK Engineering and Physical Sciences Research Council (EPSRC) grant no. EP/X032914/1, project Challenges in Immersive Audio Technology.

References

[1] Rumsey, F., Zieliński, S., Kassier, R., and Bech, S., "On the Relative Importance of Spatial and Timbral Fidelities in Judgments of Degraded Multichannel Audio Quality," *J. Acoust. Soc. Am.*, 118(2), pp. 968–976, 2005, doi:10.1121/1.1945368.

[2] Ono, K., Pulkki, V., and Karjalainen, M., "Binaural Modeling of Multiple Sound Source Perception: Methodology and Coloration Experiments," *111th AES Convention*, 2001.

[3] Ono, K., Pulkki, V., and Karjalainen, M., "Binaural Modeling of Multiple Sound Source Perception: Coloration of Wideband Sound," in *112th AES Convention*, 2002.

[4] McKenzie, T., Armstrong, C., Ward, L., Murphy, D. T., and Kearney, G., "Predicting the Colouration Between Binaural Signals," *Appl. Sci.*, 12(5), pp. 1–15, 2022, doi:10.3390/app12052441.

[5] McKenzie, T. and Brinkmann, F., "Toward an Improved Auditory Model for Predicting Binaural Coloration," *J. Audio Eng. Soc.*, 73(3), 2025, doi:10.17743/jaes.2022.0192.

[6] Lladó, P., Hyvärinen, P., and Pulkki, V., "Auditory Model-Based Estimation of the Effect of Head-Worn Devices on Frontal Horizontal Localisation," *Acta Acustica*, 6(1), 2022, doi: 10.1051/aacus/2021056.

[7] May, T., Van De Par, S., and Kohlrausch, A., "A Pobabilistic Model for Robust Localization Based on a Binaural Auditory Front-End," *IEEE Trans. on Audio, Speech and Lang. Proc.*, 19(1), pp. 1–13, 2010, doi:10.1109/TASL.2010.2042128.

[8] Francl, A. and McDermott, J. H., "Deep Neural Network Models of Sound Localization Reveal how Perception is Adapted to Real-World Environments," *Nature Human Behaviour*, 6(1), pp. 111–133, 2022.

[9] Saddler, M. R. and McDermott, J. H., "Models Optimized for Real-World Tasks Reveal the Task-Dependent Necessity of Precise Temporal Coding in Hearing," *Nature Communications*, 15(1), pp. 1–29, 2024.

[10] Baby, D., Van Den Broucke, A., and Verhulst, S., "A Convolutional Neural Network Model of Human Cochlear Mechanics and Filter Tuning for Real-Time Applications," *Nature Machine Intelligence*, 3(2), pp. 134–143, 2021.

[11] Leer, P., Jensen, J., Carney, L. H., Tan, Z.-H., Østergaard, J., and Bramsløw, L., "Hearing-Loss Compensation Using Deep Neural Networks: A Framework and Results From a Listening Test," *IEEE Trans. on Audio, Speech and Lang. Proc.*, pp. 828–841, 2025, doi:10.1109/TASLPRO.2025.3536183.

[12] Hershey, S., Chaudhuri, S., Ellis, D. P., Gemmeke, J. F., Jansen, A., Moore, R. C., Plakal, M., Platt, D., Saurous, R. A., Seybold, B., et al., "CNN Architectures for Large-Scale Audio Classification," in *IEEE Int. Conf. Acoust., Speech, and Sig. Proc.*, pp. 131–135, 2017.

[13] McKenzie, T., Murphy, D. T., and Kearney, G. C., "Diffuse-Field Equalisation of Binaural Ambisonic Rendering," *Appl. Sci.*, 8(10), 2018, doi:10.3390/app8101956.

[14] McKenzie, T., Murphy, D. T., and Kearney, G., "Towards a Perceptually Optimal Bias Factor for Directional Bias Equalisation of Binaural Ambisonic Rendering," in *EAA Spatial Audio Sig. Proc. Symp.*, 2019, doi:10.25836/sasp.2019.08.

[15] McKenzie, T., Murphy, D. T., and Kearney, G., "An Evaluation of Pre-Processing Techniques for Virtual Loudspeaker Binaural Ambisonic Rendering," in *EAA Spatial Audio Sig. Proc. Symp.*, pp. 149–154, 2019, doi:10.25836/sasp.2019.09.

[16] Lladó, P., McKenzie, T., Meyer-Kahlen, N., and Schlecht, S. J., "Predicting Perceptual Transparency of Head-Worn Devices," *J. Audio Eng. Soc.*, 70(7/8), pp. 585–600, 2022, doi:10.17743/jaes.2022.0024.

[17] ITU-R BS.1534-3, *Methods for the Subjective Assessment of Intermediate Quality Level of Audio Systems*, International Telecommunication Union, Geneva, Switzerland, 2015.

[18] Abu-El-Haija, S., Kothari, N., Lee, J., Natsev, P., Toderici, G., Varadarajan, B., and Vijayanarasimhan, S., "Youtube-8M: A Large-Scale Video Classification Benchmark," *arXiv preprint arXiv:1609.08675*, 2016.

[19] Singh, N. K., Chanu, Y. J., and Pangsatabam, H., "A Study of Various Audio Augmentation Methods and Their Impact on Automatic Speech Recognition," in *Lecture Notes in Electrical Engineering*, volume 1071 LNEE, pp. 481–491, Springer Science and Business Media Deutschland GmbH, 2024, doi:10.1007/978-981-99-4713-3_46.

Audio Engineering Society

Conference Paper 19

Presented at the AES International Conference on Machine Learning and Artificial Intelligence for Audio
2025 September 8–10, London, UK

This paper was peer-reviewed as a complete manuscript for presentation at this conference. This paper is available in the AES E-Library (http://www.aes.org/e-lib), all rights reserved. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Journal of the Audio Engineering Society.

A Dataset of Amateur Karaoke Singing and Audiobook Narration

Elena Georgieva[1], Pablo Ripollés[1,2,3], and Brian McFee[1,4]

[1]*New York University Music and Audio Research Lab (MARL)*
[2]*NYU Center for Language Music and Emotion (CLaME)*
[3]*NYU Department of Psychology*
[4]*NYU Center for Data Science*

Correspondence should be addressed to Elena Georgieva (elena@nyu.edu)

ABSTRACT

Singing and narration are fundamental forms of vocal expression. Listener-evaluated data is essential for developing AI systems that better capture the nuances of vocal expression and resonate with listeners perceptually and emotionally. In this paper, we present a dataset with 4300 ratings of 940 recordings of amateur karaoke singing and audiobook narration, annotated by 86 participants. Participants rated multiple aspects of audio quality and vocal performance including skill, likability, passion, sincerity, and intelligibility, among others. We release this dataset alongside baseline analyses comparing ratings of singing and audiobook narration, as well as ratings of clean excerpts versus those with added degradations. We also release a linear regression model that estimates listener ratings from audio. This dataset will serve as a valuable resource for future research in music and speech, with applications in intelligent music production, vocal audio effect recommendation, audiobook editing, content personalization, and automated assessment of singing and speaking voices, to name a few.

1 Introduction

The intelligent music production community could benefit from listener-evaluated datasets to enhance machine learning models, particularly for vocals—one of the most inherently human forms of musical expression [1]. Notably, among the available AI audio datasets, only a few include information from human subjects.[1] At the same time, audiobooks are becoming more popular than ever, changing traditional notions of reading and storytelling, and reshaping the way listeners engage with literary works. The integration of AI is expected to transform audio literature through AI-generated narration, and human annotations provide crucial insights into how listeners perceive vocal performance, expressiveness, and intelligibility in spoken-word audio [2, 3]. Ultimately, more listener-evaluated data is essential for developing AI systems that replicate or analyze the human voice.

Singing is accessible to anyone with a voice, and even those who do not self-identify as "singers" can express themselves through vocal performance. While much research focuses on trained vocal professionals, who develop their skills through voice lessons and choral singing [4, 5]—less is known about unskilled amateur

[1]https://github.com/Yuan-ManX/ai-audio-datasets

performances, which this study explores. We study karaoke recordings because they are performed by amateurs, likely without formal musical training.

In parallel, we examine the same vocal characteristics in audiobook narration performance. A skilled narrator can contribute considerably to the success of a book, and a poor narrator can ruin it [6]. Understanding what makes an effective narration is valuable, as audiobook narration is a performative vocal art form that calls for significantly more vocal expression than typical conversational speech [7]. We choose to study audiobook narration as a proxy for expressive speech more broadly—including podcasts and radio—due to its performative nature and the availability of LibriSpeech, the dataset we use.

Skilled singing performance and audiobook narration both demand vocal expressiveness, precise rhythm, and emotional depth. Singing requires sustained pitch accuracy and musical phrasing, and audiobook narration involves clear enunciation and pacing. Both challenge performers to effectively convey meaning and engage their audience, with performances varying in skill level and technique.

We collected subjective ratings from 86 participants who listened to isolated karaoke recordings and audiobook narration performances of varying audio quality and levels of performer skill. Listeners provided subjective ratings on performer skill, likability, passion, sincerity, emotional effectiveness, power, intelligibility, familiarity, background noise, intelligibility due to recording quality, and whether there were multiple speakers/singers. This dataset not only offers valuable insights into how amateur performances are perceived, but also serves as a resource for machine learning and AI applications related singing and speech.

We release the responses as a dataset, along with baseline analyses.[2] As preliminary analyses, we include results of a linear regression model to estimate the behavioral responses based on a set of extracted acoustic features, including pitch and spectral characteristics. In addition, we fit linear mixed-effects models to compare subjective ratings between karaoke and audiobook narration excerpts, examining how listeners assess recordings in these distinct contexts. We discuss how listeners rate clean versus noisy excerpts differently, leading to implications for audio effects recommendation in intelligent music production and audiobook production. Through our analysis, we confirm the reliability of the listeners' ratings, demonstrating that the data collection was thorough and that participants provided attentive and meaningful evaluations.

2 Related Work

In the field of Intelligent Music Production, listener evaluations exist for mix datasets [8, 9], and audio quality has been listener-evaluated for popular music [10]. However, other studies have not examined listener evaluations of amateur vocal performances. Audiobooks, more popular than ever, are also being shaped by advancements in artificial intelligence, including generated voices and audiobook recommendation [11]. These developments highlight the need for human evaluations to better understand how listeners perceive AI-generated and human-performed vocal expression. The Kaggle Dataset provides metadata on popular audiobooks, including user ratings and reviews, but lacks direct human evaluations of narration quality. [3] Despite AI's growing role in singing and speech production, listener-evaluated datasets for spoken and sung performances remain scarce—highlighting a critical gap and an opportunity for meaningful contribution.

Previous research has explored the automatic prediction of singing skill using acoustic features. Past work has found that the singing power ratio (SPR) is a useful metric [12]. SPR calculates the ratio between peak intensities in the 2–4 kHz and 0–2 kHz frequency bands. Watts et al., (2006) found that there is a notable difference in SPR values between trained and untrained singers, with trained singers having a more resonant sound with vocal "ring". Other research has focused on automatically assessing perceived singing quality using a variety of classifiers and features [13]. Another approach to skill evaluation relies on pitch-based analysis [14, 15, 16]. While pitch is an important measure of vocal skill, it is only one of many factors that shape listener perception of vocal skill.

We also investigated how listeners perceived audio quality. A widely used listener-based metric for evaluating speech quality is the Mean Opinion Score (MOS), where listeners rate audio quality on a five-point Likert

[2] https://github.com/elenatheodora/KaraokeAudiobook

[3] https://www.kaggle.com/datasets/s4lman/top-audiobooks-dataset

scale (Bad, Poor, Fair, Good, Excellent) [17]. Recently, research has focused on blind MOS estimation, using audio signals online. One model estimates MOS as well as reverberation time (T60) and clarity (C50) [18]. Another uses a convolutional neural network to estimate MOS for individual speech utterances [19].

On the singing side, researchers have explored sung lyric intelligibility—defined as a listener's ability to accurately comprehend lyrics [20]. Despite adjacent work on speech quality assessment and singing skill prediction, no prior study has focused on predicting subjective listener evaluations of vocal recordings in different domains (i.e., karaoke, audiobook narration) and across the range of characteristics studied here.

3 Methods

3.1 Dataset

Participants in this study listened to karaoke and audiobook narration recordings. The karaoke singing voice recordings were sourced from the Digital Archive of Mobile Performances (DAMP), recorded and collated by Smule, Inc [21]. The archive includes solo vocal recordings from smartphone users using Smule's karaoke mobile application,[4] and has at least 35,000 vocal performances of contemporary American pop songs like "Love Yourself" by Justin Bieber and "Rockabye" by Clean Bandit. The dataset includes a great diversity of vocal training and performance styles, as it is mostly amateur singers singing karaoke for fun. Recordings come from a wide variety of locations around the world, and have varying recording qualities depending on the device used and level of background noise. The recordings were captured using smartphones and may include compression artifacts, background noise (depending on the recording environment), and device-specific limitations such as poor microphone quality, that could result in an unbalanced frequency response, limited dynamic range, distortion at high input levels, and other artifacts.

The audiobook narration performance recordings were drawn from the LibriSpeech dataset—a corpus of approximately 1000 hours of English audiobook narration [22]. Similarly to the DAMP dataset, LibriSpeech has a variety of narrators with different levels of training, different performance styles, and varying audio quality. LibriSpeech is derived from audiobooks that are part of the LibriVox project [23], a selection of free public domain audiobooks read by volunteers from around the world. We randomly sampled a subset of recordings from both datasets and manually screened them based on specific selection criteria.

For the karaoke recordings, we selected a diverse set of performances while ensuring variety in song choices. The dataset contained multiple performances of the same songs, and we limited the number of times any single song appeared to avoid overrepresentation. We also included a range of performance skill levels, assessed perceptually by co-author E. Georgieva, who has 18 years of experience singing in various vocal ensembles. For audiobook narration, we selected passages from different genres, fiction and non-fiction, while avoiding content with violent or distressing themes.

In both datasets, recordings were chosen to minimize background noise based on objective and subjective listening. We first used spectral contrast to estimate the most clean audio clips. Then, co-author E. Georgieva, who also has training and experience as an audio engineer, reviewed the the files resulting from the spectral contrasts selection and chose the cleanest files. The final study included 940 audio clips: 470 karaoke and 470 audiobook narration. All audio files were monophonic with a 48 kHz sampling rate.

For 840 of these recordings, we used the clean, unedited clip. To control for the quality of the ratings, we degraded 100 recordings (50 karaoke, 50 audiobook) using one of four methods designed to intensify existing recording degradations: low distortion, high distortion, low noise, and high noise. For the distortion files, we used the Pedalboard library to apply nonlinear harmonic distortion via a hyperbolic tangent waveshaping function at two effect levels: low (20 dB drive) and high (a greater drive setting) [24]. This simulates distortion from mic clipping in recordings, but with a greater amount than is naturally present in the data. For the noise conditions, we added pink noise to the audio files at two signal-to-noise ratio (SNR) levels: low noise (0.02 amplitude) and high noise (0.1 amplitude), using SoX's pink noise generator [25]. This process introduced controlled background noise to simulate environmental background noise, alongside the original audio signal. All clips, clean and with added noise, were normalized to the European Broadcasting Union (EBU) R128 loudness standard using FFmpeg [26].

[4] https://apps.apple.com/us/app/smule-sing-record-karaoke/id509993510

3.2 Listener Study Methodology

The vocal characteristics that we are interested in are subjective; therefore, evaluations from human listeners are the best data to describe them. Data were collected online through the SONA platform[5] for recruitment of participants at New York University, as well as via a university email list, and word of mouth. Experiments were hosted on Qualtrics.[6] The Qualtrics survey had audio playback functionality, allowing participants to listen to each 15-second clip before responding. All responses were collected anonymously.

Participants were first presented with the informed consent and then with instructions about the task to be completed. Participants were required to use headphones and underwent a headphone test to determine if they were doing so. This test has been previously validated and is widely used in online experiments using auditory stimuli [27, 28]. Listeners judge which of three pure tones is quietest, with one tone presented 180° out of phase between stereo channels. The task is easy with headphones but difficult with loudspeakers due to phase cancellation. The test is repeated six times, accounting for all possible order combinations of the three pure tones presented.

Participants were then presented with a series of 15-second audio clips of either karaoke singing or audiobook narration and asked to evaluate each recording. The 15-second audio clips included 500 ms of fade-in and 500 ms of fade-out. Listeners could pause/play as they liked and listen to the audio clip as many times as desired. Each participant rated 50 audio recordings (25 karaoke songs; 25 audiobook narrations), where 42 recordings were clean and eight (four karaoke; four audiobook) had the added degradations. Audio files were presented in an interspersed and randomized order, and completed some demographic questions along with the Goldsmith-MSI (measuring musical training) and Barcelona Music Reward Questionnaire (measuring sensitivity to musical reward) [29, 30]. The study lasted around 60 minutes and listeners were compensated for their time either with SONA academic credit or a $15 gift card.

After listening to each audio clip, respondents rated their agreement with seven statements regarding the performance and audio quality of the recording on a five-point Likert scale with options "Completely Disagree(1)", "Disagree(2)", "Neither Agree Nor Disagree(3)", "Agree(4)", and "Completely Agree(5)". Statements assessed perceptions of skill, performance quality, emotional expression, vocal power, intelligibility, and familiarity with the audio content. Participants were required to spend at least 25 seconds on each page before proceeding to the next question. The seven statements were:

- "The performer was highly skilled in delivering the spoken or sung text."
- "I really liked the way the excerpt was performed."
- "The performer conveyed a lot of passion in their performance."
- "The performer's expression was very sincere."
- "The performer conveyed emotions effectively (regardless of whether emotions were positive or negative)."
- "The performer's voice was very powerful."
- "I could clearly discern the words being spoken/sung."

A second section of questions invited participants to evaluate specific attributes of the audio recordings. Respondents answered three trinary-choice questions (options: "no(1)", "maybe(3)", "yes(5)"), addressing intelligibility and sound quality. The number choices one, three, and five were used during analysis to preserve the scale among all questions. The three questions were:

- "Is there more than one singer/speaker in the recording?"
- "Are there sounds other than the speaker/singer in the recording?"
- "The recording had unintelligible words due to recording quality."

For succinctness, we abbreviate the eleven questions to: skill, likability, passion, sincerity, emotional effectiveness, power, intelligibility, familiarity, multiple speakers, background noise, and recording quality.

We used two different strategies to identify and discard low-quality responses where listeners were likely not

[5] https://www.sona-systems.com/
[6] https://qualtrics.com

focused on the task: the headphone test and a set of catch trials. Throughout the study, participants encountered six "catch trials," in which they were presented with an audio file and asked: 'Is the following audio the same or different from the audio you heard on the previous page?' In half of these trials, the correct answer was "yes" and the audio file was identical to the previous one. In the other half, the correct answer was "no" and the audio file was different, though it came from the same dataset (e.g., audiobook or karaoke).

147 participants completed the study. The majority of these participants were recruited via the SONA platform at New York University, and were granted course credits for taking part in the study. Some participants were recruited from a NYU music technology program email list, and finally four participants were recruited by word of mouth. The non-SONA participants received a $15 gift card as compensation.

Data from 52 participants were removed because listeners failed three or more of the headphone test questions (see section 3.2). 25 participants failed two or more of the catch trials (16 of them had already been discarded for failing the headphone test). We kept recruiting participants until each recording had been evaluated by at least four individuals. Ultimately, 86 participants were included in the analysis. 63 were female, 1 non-binary/ third gender. The median age was 20.32 years, with a standard deviation of 2.89. Among the participants, the most common identities were Asian (47), White (23), and Black or African American (9). Eight participants identified with multiple racial/ethnic groups, and eight identified as Hispanic or Latino. When asked about their mother tongue, 37 participants reported English, 18 mentioned English and another language, and 31 listed "Other" as their mother tongue. In the open text field for "Other," 19 participants wrote Chinese (without specifying a dialect), eight mentioned Mandarin, and two cited Cantonese. Additionally, seven participants listed Spanish, four said Russian, three mentioned Korean, and several other languages were mentioned once or twice. We collected a total of 4300 ratings, with each of the 940 recordings being evaluated by at least four participants, averaging 4.6 ratings per audio file with a standard deviation of 1.6. All participants provided written informed consent, as approved by the local institutional review board (New York University's Committee on Activities Involving Human Subjects). All data were anonymized for sharing purposes.

We split questions into two categories: stimulus questions and participant-stimulus questions. Participant-stimulus questions were quite subjective, and responses vary between participants, as indicated by their larger standard deviations. In the stimulus questions: intelligibility, background noise, recording quality, and multiple speakers, the participants evaluated the stimuli in a more objective manner. These have smaller standard deviations across responses (see Table 2).

3.3 Statistical Analyses

To examine the relationship between acoustic features and listener ratings, we applied an Elastic Net linear regression model with 5-fold cross-validation using Python Scikit-learn [31]. In preparing this linear regression, we extracted a range of acoustic features from the audio files to capture various aspects of vocal expression. These features included the mean, standard deviation, minimum, maximum, and 25th and 75th percentiles of the estimated fundamental frequency (f_0) and the root mean square (RMS) energy. Fundamental frequency (f_0) and RMS energy were chosen because they capture key characteristics of vocal expressiveness and acoustic properties. These features provide a representation of the pitch and energy, which are essential for distinguishing differences in vocal performance. The use of f_0 estimation via Pyin and RMS energy analysis aligns with common practices in speech and music research, as they are both reliable indicators of vocal and audio signal characteristics. For f_0 estimation we used Pyin as implemented in Librosa, with default parameters, other than the sampling rate (48 kHz) [32, 33]. We computed an f_0 estimate approximately every 11 milliseconds.

Additionally, we analyze several acoustic features of vocal expressiveness: total variation of pitch (measuring the rate of pitch change [34]), the fraction of time with no voice, and pitch class entropy (which quantifying the degree of unpredictability for the set of vocal pitches). The pitch features and fundamental frequency (f_0) estimation reflects pitch variation in singing and speech, which can provide insight into performer expression and skill [15, 7].

We also computed spectral features, including spectral contrast, centroid, bandwidth, 85% spectral rolloff, and the ratio of spectral energy in the 2–4 kHz frequency range to the total spectral energy of an audio signal. These spectral features offer insights into the

timbral qualities of the voice, which can influence listener perceptions [12]. All features were computed with a standard 2048 frame length and hop size of 512. Finally, we calculate the mean, standard deviation, min, and max of the first 20 MFCCs, widely used speech features, aggregated over time [35]. These features were used as input for the Elastic Net models.

The models aimed to predict listener ratings for each stimulus, using average ratings for each of the 940 stimuli, each rated by four or more listeners. Each model was trained separately for each question, and included both karaoke and audiobook recordings in the same model. We optimized the alpha (regularization strength) and ℓ_1 ratio (balance between ℓ_1 and ℓ_2 regularization) parameters using a grid search on the validation set. The dataset was split into five folds using GroupKFold cross-validation, ensuring that all samples from the same artist appeared in either the training or test set, but not both. We performed the cross-validation five times, each producing distinct training and test sets that were saved for subsequent use.

We evaluated model performance using mean R^2 score and root mean squared error (RMSE). R^2 is the percentage of the variation in the data that the model can explain, and RMSE measures how far off the model's predictions are from the actual values, with lower RMSE indicating better predictions. We repeated this process for each question, allowing us to assess which aspects of vocal expression were most predictable from acoustic features. We evaluate our model twice: once with a full test set, and once on a clean-only test set with the noisy excerpts omitted.

To analyze the ratings, we used R (4.2.2) and RStudio (2024.12.1+563) to implement a linear model with the lm function to quantify the effect of excerpt type (karaoke vs. audiobook; i.e. skill $\sim$ ExcerptType) and noise type (clean vs. noisy; i.e. skill $\sim$ Noise) on each of the questions. We used the lm function to fit the models and used the ggpredict function from the ggeffects package to generate predictions. The models were computed separately for each question. With these linear models, we quantified how the conditions shape listeners' evaluations of the performance.

Finally, we fit a linear mixed-effects model using maximum likelihood estimation to examine the relationship between a stimulus-participant questions and the stimulus questions (e.g., Skill $\sim$ Intelligibility + Multiple_Speakers + Background_Noise + Recording_Quality + (1 | Filename) + (1 | ParticipantID)). The models included random intercepts for filename and participant identification code to account for variability across recordings and participants. To do so we used the lme4 package in R.

4 Results

The results of the user study are illustrated in Figure 1. Among the 11 questions assessed using Elastic Net linear regression models, we obtained the highest R^2 score for "Are there sounds other than the speaker/singer in the recording?" (R^2 = 0.33, RMSE = 0.75), suggesting that background noise was relatively well predicted by the extracted acoustic features. The high RMSE indicates considerable prediction error, possibility due to the variability of background noise and differences in how listeners interpret the question. We observed the second highest R^2 for "The recording had unintelligible words due to recording quality" (R^2 = 0.20, RMSE = 0.56, α = 0.01, ℓ_1 ratio = 0.01), followed by "I could clearly discern the words being spoken/sung" (R^2 = 0.20, RMSE = 0.50, α = 1.0, ℓ_1 ratio = 0.01).

Predictions for more subjective qualities, such as passion (R^2 = 0.07, RMSE = 0.51, α = 0.7, ℓ_1 ratio = 0.01) and skill (R^2 = 0.06, RMSE = 0.53, α = 1.0, ℓ_1 ratio = 0.01), showed lower predictive power, suggesting that these listener judgments were less directly tied to extracted acoustic features. Similarly, ratings for emotional effectiveness (R^2 = 0.047, RMSE = 0.49, α = 0.1, ℓ_1 ratio = 0.01), sincerity (R^2 = 0.041, RMSE = 0.46, α = 0.5, ℓ_1 ratio = 0.01), and power (R^2 = 0.037, RMSE = 0.54, α = 0.7, ℓ_1 ratio = 0.01) were only modestly predictable from acoustic features (see Table 1).

We evaluated the Elastic Net linear regression without clips with added noise in the test set (see Table 1). When comparing the results of the complete test set (including both noisy and clean data) to the clean-only test set, almost all rating questions showed a decrease in R^2 scores after removing the noisy clips, with background noise dropping significantly from 0.33 to 0.16, and recording quality dropping from 0.20 to -0.01.

To quantify the effect of excerpt type (karaoke vs. audiobook) on the average ratings provided for each question, we ran a series of linear regression models to determine the relationship between the variable of interest and excerpt type (i.e. skill $\sim$ ExcerptType). For

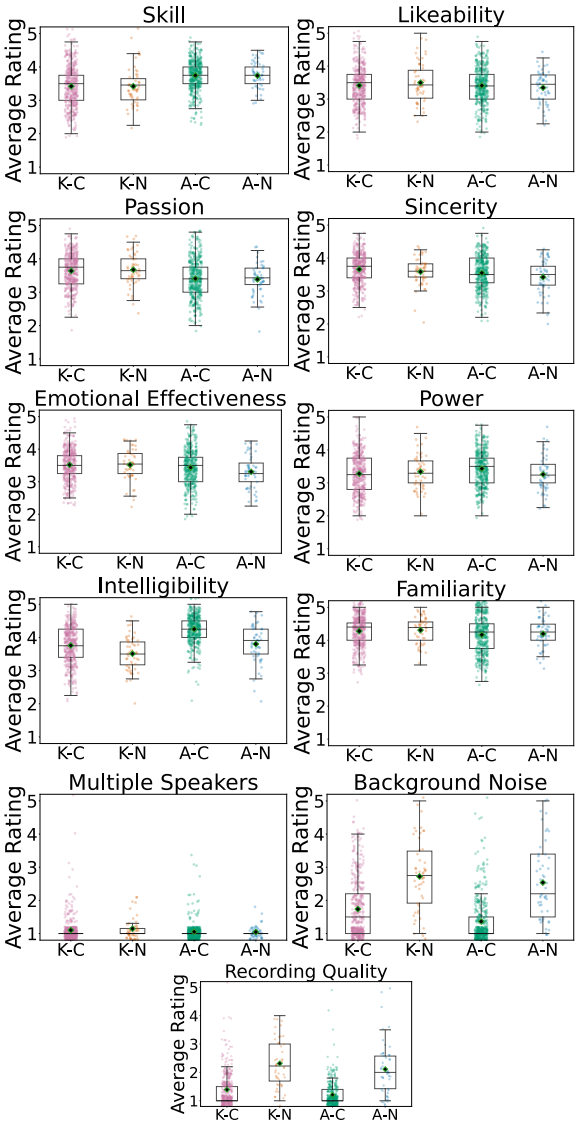


Fig. 1: Distribution of mean ratings for the 11 questions, split up by Excerpt Type (karaoke vs. audiobook) and Noise Type (clean vs. noisy). Each vertical box represents the 25th percentile, median, and 75th percentile, with whiskers extending 1.5x the interquartile range. The mean is represented by a diamond.

skill, excerpt type had a significant negative effect (β = -0.324, t = -9.459, p < 0.001), indicating that karaoke excerpts were rated lower in skill. Likability showed no significant difference between excerpt types (β = 0.019, t = 0.539, p = 0.59). In contrast, passion was

Table 1: Comparison of Elastic Net linear regression results with 5-fold cross-validation, including R^2 and RMSE for each question. Results with the full test set are in the left column, and results and with only clean clips in the test set (degraded clips omitted) are in the right. The model parameters were optimized via grid search for each listener-rated question.

Question	**Full Test**		**Clean Test**	
	R^2	RMSE	R^2	RMSE
Skill	0.06	0.53	0.06	0.53
Likability	-0.01	0.53	-0.01	0.53
Passion	0.07	0.51	0.07	0.51
Sincerity	0.04	0.46	0.03	0.46
Emotional Effect.	0.05	0.49	0.04	0.50
Power	0.04	0.54	0.03	0.54
Intelligibility	0.20	0.50	0.19	0.49
Familiarity	-0.02	0.52	-0.02	0.53
Multiple Speakers	-0.01	0.31	-0.02	0.31
Background Noise	0.33	0.75	0.16	0.74
Recording Quality	0.20	0.56	-0.01	0.51

rated significantly higher for karaoke excerpts (β = 0.237, t = 7.072, p < 0.001). Sincerity (β = 0.115, t = 3.793, p < 0.001), emotional effectiveness (β = 0.094, t = 2.841, p = 0.005), and familiarity (β = 0.109, t = 3.276, p = 0.001) were also rated higher for karaoke excerpts. However, power (β = -0.129, t = -3.619, p < 0.001) and intelligibility (β = -0.471, t = -14.21, p < 0.001) were rated significantly lower for karaoke performances. These results are expected and validate the reliability of the listener ratings we release in this work.

Background noise ratings were significantly higher for karaoke excerpts than audiobook excerpts (β = 0.352, t = 6.015, p < 0.001). Recording quality was also higher for karaoke excerpts (β = 0.182, t = 4.472, p < 0.001), indicating "the recording had unintelligible words due to recording quality," and reinforcing the perception that the karaoke recordings had poorer audio quality. "Multiple speakers" was also higher for karaoke (β = 0.057, t = 2.881, p = 0.004).

The regression models examining the impact of noise type (clean vs. noisy) on various perceptual ratings showed minimal effects across most attributes. Noise had no significant effect on skill (β = -0.003, t = -0.051, p = .959), likability (β = 0.011, t = 0.187, p = .852), passion (β = 0.008, t = 0.136, p = .892), emotional

effectiveness (β = -0.065, t = -1.209, p = .227), power (β = -0.057, t = -0.986, p = .324), familiarity (β = 0.030, t = 0.560, p = .576), and multiple speakers (β = 0.019, t = 0.601, p = .548), suggesting that those elements remain unchanged in the presence of noise, at least the two types of noise in the levels added in this work. For sincerity, noise significantly reduced ratings (β = -0.102, t = -2.059, p = .040), indicating that voices in noisy conditions are perceived as less sincere. For intelligibility, noise significantly lowered ratings (β = -0.349, t = -6.000, p < .001), indicating that speech in noisy conditions is perceived as less clear. Finally, for background noise and recording quality, noise significantly increased ratings (that is, more perceived noise) (β = 1.079, t = 11.98, p < .001) and (β = 0.913, t = 15.26, p < .001), respectively, confirming that listeners perceived more background noise and lower intelligibility due to recording quality in noisy conditions. Our linear mixed-effects model examined the relationship between the more subjective stimulus-participant questions and the more objective stimulus questions (intelligibility, background noise, recording quality, and multiple speakers), and included random intercepts for filename and participant ID code to account for variability across recordings and participants. For skill, the conditional R^2 value, which accounts for both fixed and random effects, was 0.23, indicating that 23% of the variance in skill ratings was explained by the full model. The marginal R^2, representing the variance explained by fixed effects alone, was 0.05.

Table 2: Average standard deviations for each of the 11 questions.

Question	St. Dev.
Skill	0.97
Likability	1.05
Passion	0.92
Sincerity	0.85
Emot. Effectiveness	0.93
Power	0.97
Intelligibility	0.75
Familiarity	1.06
Multiple Speakers	0.14
Background Noise	0.78
Recording Quality	0.58

For likability, the conditional R^2 value was 0.21, while the marginal R^2 was 0.02. For passion, the conditional R^2 was 0.20, and the marginal R^2 was 0.01. For sincerity, the model explained 20% of the variance (conditional R^2 = 0.20), with fixed effects contributing minimally (marginal R^2 = 0.02). For emotional effectiveness, the model explained 21% of the variance (conditional R^2 = 0.21), with fixed effects contributing modestly (marginal R^2 = 0.02). For power, the model explained 24% of the variance overall (conditional R^2 = 0.24), with fixed effects contributing about 4% (marginal R^2 = 0.038). Finally, for familiarity, the model explained 57% of the variance overall (conditional R^2 = 0.57), but the fixed effects contributed very little (marginal R^2 = 0.0025), suggesting that most of the variance is due to differences between respondents rather than the predictors tested.

5 Discussion

As may be expected, the results of our Elastic Net linear regression indicate that aspects of intelligibility and recording quality were more predictable from audio than subjective ratings of vocal recordings (see Table 1). Three of our four stimulus questions (intelligibility, background noise, recording quality, and multiple speakers) had the highest R^2 scores. The exception is "multiple speakers," where there was a floor effect. Likability and familiarity, arguably the most subjective questions, had the lowest R^2 scores, as well as the highest average standard deviations. As expected, more subjective questions were harder to predict with a model using acoustic features.

The results of our linear mixed-effects model, which explored how objective stimulus responses predict subjective participant-stimulus responses, can be compared with those from our Elastic Net linear regression, where acoustic features predicted subjective participant-stimulus responses (see Table 1). The linear mixed-effects models explained between 20% and 24% of the variance in subjective ratings (conditional R^2 .20-.24), but the fixed factors explain only 2% to 5% of the variance (marginal R^2 .02-.05) which is in line with the results from the Elastic Net analysis. Both analyses show that it is difficult to predict the subjective participant-stimulus ratings from the selected objective acoustic features and more objective stimulus ratings. Notably, familiarity differed from the other questions, as it was primarily explained by individual differences, with minimal contribution from the fixed factors (marginal R^2 = 0.0025) and a high overall model fit (conditional R^2 =

0.57). This is expected, and suggests that the variability in familiarity ratings is driven more by participant-specific factors rather than the audio files' characteristics, and further validates the methodology used.

Listeners clearly picked up on the degradations we added to some of the audio files, as seen in the higher (more noisy) average ratings for background noise and recording quality, for both excerpt types. Intelligibility also decreased in noisy files for both excerpt types (see Figure 1), though less dramatically than for background noise and recording quality. The success of the linear regression for background noise and recording quality decreases greatly when the test set contained only clean data (with the added-noise excerpts omitted). The drop is likely due to the difference in listener ratings between clean and noisy recordings. Also, the ratings for the clean karaoke and audiobook files are quite low, and there is a floor effect, as illustrated in Figure 1.

For Excerpt Type, we found that karaoke excerpts were rated lower in skill but higher in passion, sincerity, emotional effectiveness, and familiarity compared to audiobook excerpts. In contrast, karaoke performances were rated significantly lower in power and intelligibility. These findings suggest that listeners are controlling for excerpt type in their likability ratings, find singing performances to be more emotionally engaging, but are more harsh when rating singer skill and power. As may be expected, participants found speech in karaoke harder to understand than in narrations. The recording quality was likely lower in the Smule dataset than in the Librispeech dataset, based on the higher ratings of background noise and recording quality for both excerpt types (Figure 1).

This study provides a new dataset and a baseline predictive model. Future work can expand acoustic features to incorporate into the model, and explore alternative models to improve performance. It is also important to acknowledge some limitations of the dataset: the listener ratings primarily come from individuals with a Western cultural background, and the participant pool largely consists of young, college students. These factors may shape perceptual judgments and limit the generalizability of our findings across diverse populations. Additionally, perceptions of traits like sincerity, passion, and likability may vary significantly across cultures, and our current design does not account for potential cultural biases in how these qualities are interpreted. Future work can include more demographically and culturally diverse listeners.

References

[1] Noufi, C., May, L., and Berger, J., "The Role of Vocal Persona in Natural and Synthesized Speech," in *IEEE International Conference on Automatic Face and Gesture Recognition*, pp. 1–4, 2023.

[2] Walsh, B., Hamilton, M., Newby, G., Wang, X., Ruan, S., Zhao, S., He, L., Zhang, S., Dettinger, E., Freeman, W. T., and Weimer, M., "Large-Scale Automatic Audiobook Creation," in *Interspeech*, ISCA, Dublin, Ireland, 2023.

[3] Speechify Inc., "Speechify: AI-Powered Text-to-Speech App," `http://speechify.com`, 2024.

[4] Scherer, K., "The expression of emotion in the singing voice: Acoustic patterns in vocal performance," *J. Acoust. Soc. Am.*, 142(1805), 2017.

[5] Sundberg, J., Salomão, G. L., and Scherer, K. R., "Emotional expressivity in singing. Assessing physiological and acoustic indicators of two opera singers' voice characteristics," *Journal of the Acoustical Society of America*, 2024.

[6] Komar, S., "Listened To Any Good Books Lately? The Prosodic Analysis of Audio Book Narration," *ELOPE: English Language Overseas Perspectives and Enquiries*, 3, p. 85–98, 2006.

[7] Ji, D., Liu, B., Xu, J., and Gong, J., "Why Do We Listen to Audiobooks? The Role of Narrator Performance, BGM, Telepresence, and Emotional Connectedness," *Sage Open*, 14(2), 2024.

[8] De Man, B., Boerum, M., Leonard, B., and et al., "Perceptual evaluation of music mixing practices," *AES Journal*, 1, pp. 129–136, 2015.

[9] Man, B. D. and Reiss, J. D., "The Mix Evaluation Dataset," in *Digital Audio Effects (DAFx)*, 2017.

[10] Wilson, A. and Fazenda, B., "Perception of Audio Quality in Productions of Popular Music," *AES Journal*, 64, pp. 23–34, 2016.

[11] Palanisamy, B. and V, R., "The listening renaissance: a theoretical exploration of audio-based digital narratives in literature," *Humanities and Social Sciences Communications*, 12, p. 96, 2025.

[12] Watts, C., Barnes-Burroughs, K., Estis, J., and Blanton, D., "The singing power ratio as an objective measure of singing voice quality in untrained talented and nontalented singers," *Journal of Voice*, 20(1), pp. 82–88, 2006.

[13] Böhm, J., Eyben, F., Schmitt, M., and et al., "Seeking the SuperStar: Automatic assessment of perceived singing quality," in *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 1560–1569, 2017.

[14] Nakano, T., Goto, M., and Hiraga, Y., "An automatic singing skill evaluation method for unknown melodies using pitch interval accuracy and vibrato features," in *Proc. INTERSPEECH*, pp. 1706–1710, ISCA, 2006.

[15] Bozkurt, B., Baysal, O., and Yüret, D., "A dataset and baseline system for singing voice assessment," in *International Symposium on Computer Music Multidisciplinary Research (CMMR)*, 2017.

[16] Panteli, M., Bittner, R., Bello, J. P., and Dixon, S., "Towards the Characterization of Singing Styles in World Music," *IEEE ICASSP*, 2017.

[17] International Telecommunication Union, "Subjective evaluation of speech quality with a crowdsourcing approach," ITU-T Recommendation P.808, Geneva, 2018.

[18] Akrami, H. and Gamper, H., "Speech MOS multi-task learning and rater bias correction," in *ICASSP*, pp. 1–5, IEEE, 2023.

[19] Gamper, H., Reddy, C., Cutler, R., Tashev, I., and Gehrke, J., "Intrusive and Non-Intrusive Perceptual Speech Quality Assessment Using a Convolutional Neural Network," in *WASPAA*, 2019.

[20] Ibrahim, K. M., Grunberg, D., Agres, K., Gupta, C., and Wang, Y., "Intelligibility of Sung Lyrics: A Pilot Study," in *ISMIR Conference*, 2017.

[21] Smule, Inc., "DAMP-VPB: Digital Archive of Mobile Performances - Smule Vocal Performances," 2017, doi:10.5281/zenodo.2616690.

[22] Panayotov, V., Chen, G., Povey, D., and Khudanpur, S., "Librispeech: An ASR corpus based on public domain audio books," in *2015 IEEE ICASSP*, South Brisbane, QLD, Australia, 2015.

[23] LibriVox, "LibriVox | Free Public Domain Audiobooks," 2005, accessed: 2025.

[24] Sobot, P., "Pedalboard," 2021, doi:10.5281/zenodo.7817838.

[25] SoX Development Team, "SoX - Sound eXchange," 1991, accessed: 2025.

[26] FFmpeg Developers, "FFmpeg," 2000, accessed: 2025.

[27] Woods, K. J. P., Siegel, M. H., Traer, J., and et al., "Headphone screening to facilitate web-based auditory experiments," *Attention, Perception, & Psychophysics*, 79, pp. 2064–2072, 2017.

[28] Orpella, J., Bowling, D. L., Tomaino, C., and Ripollés, P., "Effects of music advertised to support focus on mood and processing speed," *PLOS ONE*, 20(2), p. e0316047, 2025.

[29] Mas-Herrero, E., Marco-Pallares, J., Lorenzo-Seva, U., and et al., "Individual Differences in Music Reward Experiences," *Music Perception*, 31(2), pp. 118–138, 2013.

[30] Müllensiefen, D., Gingras, B., Musil, J., and Stewart, L., "The Musicality of Non-Musicians: An Index for Assessing Musical Sophistication in the General Population," *PLOS ONE*, 2014.

[31] Pedregosa, F., Varoquaux, G., Gramfort, A., and et al., "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, 2011.

[32] Mauch, M. and Dixon, S., "Pyin: A fundamental frequency estimator using probabilistic threshold distributions," in *IEEE ICASSP*, 2014.

[33] McFee, B., McVicar, M., Faronbi, D., and et al., "librosa/librosa: 0.10.2.post1," 2024.

[34] Georgieva, E., Ripollés, P., and McFee, B., "The Changing Sound of Music: An Exploratory Corpus Study of Vocal Trends Over Time," in *ISMIR Conference*, 2024.

[35] Davis, S. and Mermelstein, P., "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(4), pp. 357–366, 1980.

Audio Engineering Society

Conference Paper 20

Presented at the AES International Conference on Machine Learning and Artificial Intelligence for Audio
2025 September 8–10, London, UK

This paper was peer-reviewed as a complete manuscript for presentation at this conference. This paper is available in the AES E-Library (http://www.aes.org/e-lib), all rights reserved. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Journal of the Audio Engineering Society.

It's All About Speed: AI's Impact on Workflow in Music Production

Finn McClellan[1] and Fabio Morreale[1,2]

[1] *Waipapa Taumata Rau - University of Auckland, Auckland (Aotearoa - New Zealand)*
[2] *Sony AI, Barcelona (Spain)*

Correspondence should be addressed to Finn McClellan (finnmcclellan@gmail.com)

ABSTRACT

In this paper, we present the results of an ethnographic study into the impact of AI and automated tools on music production workflow. Focusing specifically on professional participants who identified as recording engineers, mixers, and producers, we discuss their usage of common AI and automated software, as well as their sentiments on the proliferation of these tools. We discuss tensions that may be created between users and automated tools in key areas such as the need for speed and efficiency, controllability, and maintaining creative agency, and how these tensions may be alleviated through tool design.

1 Introduction

The music recording industry is driven by technological innovation [1]. Digital Audio Workstations, in particular, have revolutionised the way in which we record, produce, mix, and master records. Rapidly increasing computational power has relegated the track limits of tape recorders and room-sized recording desks to phenomena of the past. The next revolution in recording is poised to be the increased automation of digital processes and artificial intelligence (AI).

Professionals represent a unique market for automated and AI tools in music production, as they often have ingrained workflows and high standards for the tools they use [2]. However, while some studies on the adoption of these technologies like Sai Vanka et al. [2] have included professionals among their participants, to the best of our knowledge, no study delved deep into understanding professionals. Given that the workflow of these professionals impacts the sounds of the popular music industry, and they represent a lucrative market to pro-audio software manufacturers, we argue that an in-depth examination of their habits, opinions, and uptake of these tools represents a valuable space for research.

In the context of this paper, we primarily discuss workflow on what we term *micro-scale* (e.g., the individual processes that go into mixing a record), as opposed to a *macro-scale* (e.g., the role of mixing in the greater production process). Notably, the impacts of changes in workflow are important, representing changes in the career and employment of professionals worldwide. Many professionals in the industry today have what may be referred to as portfolio careers [3]. It is feasible for one person to take the roles of producer, recording, and mixing engineer, with many professionals either doing this simultaneously, or switching from one role to the next depending on the project. Therefore, the

potential to automate one or more aspects of the job may have an outsize impact on the way professionals work.

The primary research question we aim to answer is: How are automated production tools affecting the workflow of music recording professionals? In order to answer this question, we explore 1) the sentiments of professionals on the proliferation of automation in their industries; 2) the specific tasks that offer the most potential for automation; and 3) the extent to which the design of AI tools affect professionals' relationship with them.

To answer these questions, we analyse qualitative data from professionals, uncovering themes which help us to understand the complex relationship between music professionals and current AI/automation software. Specifically, we discuss how recording professionals navigate the fine balance between speed and control, trust and comfort, and sharing versus maintaining creative agency.

In the next section, we examine current scholarship on automated systems in production, reporting existing academic discourses in creative agency and tool appropriations. Then, we report the methodology of this study and discuss the results of our study with the participants.

2 Background

In this section, we review related scholarship in the fields of music production, automation and AI, and technological appropriation.

2.1 Automation and Creative Agency

Sai Vanka et al. [2] consider implications of automation on workflow as paramount. In particular, they consider the sentiment around AI assisted tools for mixing in a variety of contexts, including amateur, semi-professional, and professional. In their work, they highlight how the popularisation of digital recording and production tools has "democratised" the production process, leading to the creation of these different user groups (ibid.). Different subsets of users have different requirements for AI or automatic production tools [4], which further motivates a specific investigation into professionals. Thorley [3] provides a case-study of the ways that professional mixing engineers' workflows have changed since the digital recording revolution. Improvements in computing power and internet bandwidth have enabled the rise of "remote mix engineers" (ibid.), who often work out of their own studios. Utilising automation to improve efficiency and supplement human creativity may be seen as a logical progression of this phenomenon.

Shneiderman [5] expands on this theme of automation as an aide rather than a replacement for the human, warning against the conflation of automated machines with intelligence. "If you confuse the way you treat machines with the way you treat people, you may end up treating people like machines, which devalues human emotional experiences, creativity, individuality, and relationships of trust" (ibid.). Roy et al. [6] agree that "automation should be part of, and designed as a tool to augment human intellect." However, "Beyond technological limitations, a trade-off can also be about resource allocation, or where to focus research efforts: better automation, or better controllability?" [6] Despite end-users often desiring more control, for designers "relying on the user's help is often perceived as a failure" (ibid.), representing a potential disconnect between products and their desired markets.

Roy et al. [6] also state that "the main promise of better automation is the reduction of humanity's workload." In the case of the professional this statement appears to be the primary concern, where amateur users may actually prefer for a tool to take more control, as they have less technical ability to execute the work themselves [2]. Therefore, designers must decide whether they cater their tools to one market, or potentially compromise, for example by having a hidden UI for advanced options. The outcome may depend largely on the context of the tool's intended use, as menu-diving is undesirable when equipment must be operated in real time [7]. In the experiment conducted by Roy et al. [6] they conclude that "while extrapolated results must always be taken with caution, this suggests people may always use some manual control when they can… This may be a strong argument in favour of focusing on controllability before accuracy during the design of automated tasks." However, the "black box" [2] design model of automation in music software presents a challenge to controllability, and therefore to technological appropriation as an integral source of innovation. Moffat [8] describes how "a large proportion of these technological innovations are a result of borrowing, using, or misappropriating technology." Therefore, reducing the

controllability of a tool may reduce chances for "happy accidents" (ibid.) in music production processes.

2.2 Autonomous Mixing Systems

The process of mixing has become integrated in music production workflow. Although there are many specialist mix engineers, for lower-budget projects the roles of producer, recording engineer, and mixer are often combined [3]. Izhaki [9] states that the mix engineer is responsible for helping to "deliver the emotional context of a musical piece." The fundamental quality of a mix can be measured by how well the "sonic vision" of a mixing engineer aligns with that of the artist (ibid.).

This question of subjectivity and the importance of understanding musical context are important factors when considering the automation of the mixing process. Pestana et al. [10] categorise these factors into assumptions through extensive analysis of successful mixes and professional engineers, through which they form 'best practices' for an Autonomous Mixing System (AMS). Scholars argue that AMSs can "lower the barrier of entry," required to engage in mixing [11]. However, they also state that intelligent music production (IMP) software may help "expedite the workflow of professional engineers." Steinmetz et al. [11] posit a new way to create an AMS using a deep learning model instead of classical machine learning or rule-based models. While the model demonstrates that autonomous mixing tools are improving, a tool such as this is unlikely to be useful to professionals without a significant ability to execute an artist's specific vision. Similarly, Koo et al. [12] demonstrate the difficulties of training a model to replicate stylistic mixing effects from one track to another. Perez-Gonzalez and Reiss [13] further emphasise the importance of context in the mixing process. Their system was able to maintain a balanced stereo field and reduce masking through spectral analysis (ibid.), but was not able to consider contextual panning decisions such as consistency of perspective for instruments recorded in stereo (such as drums and piano), or intentional subversion of panning norms. Training of these models is made more difficult because of the lack of large, ethical databases containing multitracks and high-quality mixes [14].

Recording industry clients are often "demanding and very precise in what they want on a granular level" [2], requiring professionals to both understand the artistic vision, and have the technical knowledge to implement it into the medium. Similarly, Loor Paredes [15] notes that "human creativity and improvisation raise tensions, affordances, and concerns when confronted with AI and digital technologies," highlighting the challenge of effectively integrating automation into creative workflows.

2.3 Designing for Appropriation

Technological appropriation is seen as an "important and positive phenomenon" [16], allowing a tool's uses to be decided by its user, rather than its designer [17]. Appropriation may also be the deliberate misuse [16] of a tool to achieve a desired creative outcome. Dix [16] discusses principles designers can use to create tools which promote creativity and appropriation in users, including providing visibility or transparency, and exposing the intentions of the tool. These concepts promote an open-ended design philosophy that may allow tools to integrate into established workflows. By nature of their work however, the professional is more concerned with time efficiency than the hobbyist, so endless customisability may be a detriment to workflow.

Zappi and McPherson [18] state that "appropriation does not require the designer to explicitly make a tool customisable," suggesting that "highly constrained instruments encourage the discovery and exploration of hidden affordances." While their study specifically highlights physical instruments, it is possible that this could also be the case for digital tools. Crucially, these constraints should not infringe on functional usability or creative agency. However, in a professional context, this philosophy of experimentation may be at odds with a reality where engineers are commonly paid a flat fee for mixing services, or studio time is limited by artist budget. In this environment, balancing efficiency with creativity is paramount, and designing tools in a way that makes them easily appropriable may offer competitive advantages to developers.

Theories in human-computer interaction (HCI) may help in understanding the workflow of modern music recording. Computers have significantly democratised the music creation process [3], enabling amateur and semi-professional users to have access to many of the same tools as professionals. Therefore, the professional's value lies in taste, intangible skills, and efficiency. Sai Vanka et al. [2] state that "the right amount of control and automation has always been a matter of

debate in human-computer interaction." Roy et al. [6] argue that this is especially important in automation, where user experience can be significantly affected by the ease of manual adjustment in an automated task.

Conversely, Fels and Lyons [19] state that the "flexibility of new technologies may impose a confusing array of choices for composer and performers." The theory that restrictions have a positive impact on creativity is well documented in many fields [20] [18], but it is important to distinguish between this type of restriction, and a lack of control, which can negatively interfere with the ability to complete work. McDermott et al. [7] assert that some difficulty and complexity are essential to the ideas of "open-endedness and long-term engagement." "Ease of learning in interfaces tends to go along with being locked-down, inflexible, inexpressive, or non-amenable to creative, unexpected use. It trades off against the long-term power of the interface." (ibid.)

3 Methodology

Professionals are *expert subjects* [21] with detailed and nuanced subjectivity, knowledge, and opinions derived from extensive experience. An ethnographic methodology was decided as the most appropriate for the study. Methods included direct observation sessions and semi-structured interviews, which allowed us to best understand these professionals in their direct context. This study treats the ethnographer as the imperfect participant, where inside knowledge of their own community frames the context of inquiry [22]. This perspective exemplifies a modern ethnographical framework, compared to the traditional anthropological approach, which frames the ethnographer as the *objective observer* [22] in the process of studying a foreign culture or society. This perspective can be problematised for a lack of integral internal understanding of the culture being studied, and a history of treating Western knowledge-systems as objective truth.

This modern ethnographic framework, where the ethnographer is seen as intersubjective rather than an objective observer, enabled us to engage in a meaningful dialogue with industry professionals as the first author is a recording engineer and producer. The music recording industry in Aotearoa New Zealand is social, but relatively isolated from other communities. Thus, the first author was able to utilise his position as a member of the community to gain close access to studio practices.

3.1 Procedure

Five industry professionals were selected to take part in the study.[1] As previously stated, many professionals in the music recording industry now have portfolio careers, working across the production process rather than specialising in a certain element (e.g., as a specialist mix engineer, or a producer). The participants had diverse backgrounds across songwriting, production, and engineering, but are all currently portfolio professionals practising as producers and engineers.

Three of the participants were observed in person while they worked on a track, with each observation lasting approximately two hours. The other two participants, who were not available for in-person sessions, were interviewed over a video call, which lasted approximately 30 minutes. Interviews were audio and video recorded.

The audio from the recordings was transcribed, and inductive thematic analysis [23] was conducted specifically to identify 1) participant's sentiments on AI and automation tools; 2) their personal workflow. Statements were coded to reflect common or contrasting themes. Recurring tensions in participants' workflows emerged from this analysis, which are discussed in Section 4.

3.2 Participant Overview and Sentiment on AI

P1 works as a songwriter, producer, and engineer, although his background is originally as a musician and artist, stating that "I never really considered myself a mixer until very recently." A lot of his workflow is based on "just getting things sounding interesting fast... maybe it's because I have the luxury of holding on to the sessions that I know I can go back and tidy it up later."

P2 is a is a freelance recording engineer, producer, and mixer. He stated that, currently, approximately 60% of his workload is spent on mixing. Despite being from a traditional audio engineering background, he uses an assortment of automated and AI tools, stating that he is "just trying to stay relatively current and on top of it all."

P3 is another hybrid producer/engineer/mixer, trained in audio engineering. Currently, he spends about 80%

[1] The study was approved by the Ethics Review Board of the University of Auckland | Waipapa Taumata Rau, #26643

of his time mixing. Often taking the approach of recording "really aggressively" via analog gear in order to give the sound its principal character, then "tweaking to elevate, as opposed to having to shape something sonically" in the mixing stage. P3 uses an assortment of automated tools, however, he states that the main area AI is of "great interest" is in the tedious, non-creative tasks associated with the profession: "AI should be there to do the things that are time-consuming for you, so you can mix more, you can record more, you can do more."

P4, another freelance engineer/producer/mixer, uses very few automated tools. He has a very static workflow, and considers knowing his tools to be an important part of workflow: "Going through that process of relearning [a new tool], it actually slows you down". He agrees that AI could potentially be useful for "the menial stuff" but believes that a good mixer can add something "outside the box," and that "the more you veer into automating those processes, the more cookie-cutter it will get."

P5 works primarily as a producer and recording engineer. By her own admission, she is "definitely looking for [new] plugins all the time," Since she also works with artists a lot, she believes that it is important to have a session sounding good from the beginning at the expense of technical perfection. "They [the artist] can kind of see the vision, but you need them to like what they're hearing straight out of the speakers." She also suggests that AI could help with productions by "showing you an automatic collection of plugins that could work well with that sound."

3.3 Common Automated Tools

This section discusses automated tools that emerged as commonly used or appropriated by participants. We note participants' reasoning and sentiment for using these tools, as well as their impact on workflow. Specific information about each tool can be found in the Appendix.

3.3.1 The God Particle

Automated tools with a "black box" [2] design approach are not universally seen as problematic. P1 uses The God Particle "because it's just so easy," and states that lack of transparency does not worry him, as long as the output sounds good. He notes that he felt more comfortable using the plugin after watching a video series where designer and mixing engineer Jaycen Joshua describes how he uses The God Particle. "You don't necessarily know what's going on, but you should know how he's using it. I feel like if I didn't know that, I wouldn't use it." The God Particle has a simple UI, containing few details about internal processing. Despite its opacity, The God Particle remains useful to professionals as a fast and easy-to-use mix bus processor.

The God Particle's success with this opaque design policy, however, does not replace the desire for transparency as a part of good user experience. P5, says that they would find it more useful if the processing was less ambiguous. "I can't trust it, I just wish there was some button to go into the back end of it, so you can see what's actually happening... Just seeing it not even being able to change it." There are green target windows on this gain reduction metering, as well as on the input metering section, which Cradle say "are where Jaycen likes to have the inputs of his mixes landing." P5 debates that "it doesn't really tell you much, because these parameters that it shows are not definitive. 'Oh, the highs are pretty low, let's turn up the highs,' but in relation to what?"

3.3.2 Soothe2

Soothe2 is notable as it is used by all participants. P1 and P3 are particularly vocal proponents, saying "Soothe is just so useful" and "I use Soothe in every mix, regardless of what I'm doing, because I know it's going to work." P3 in particular notes that Soothe has made an extremely difficult task easy, "you don't lose the will to live when something is wrong and it's so hard to fix," and in some cases it even achieves results that were not possible with manual processing. "Going back to when Soothe wasn't a thing, I don't think you could get the same results, period." However, participants note that it needs to be used carefully, with P3 noting that too much processing can make things "start to sound weird," while P1 describes an overly Soothed vocal as "fizzy."

The plugin offers a fully featured set of controls, but does not necessarily exclude amateur users. "They really went in with the [diversity of] presets and made it so that even somebody that has no idea what it does can flick between them and get lucky" [P3]. Soothe2's

customisability has allowed the tool to be useful in a variety of settings. P1 states that they often use the plugin instead of a dedicated de-esser, and as a frequency specific sidechain. "I use it as my main sidechain for kick and bass. It's nice to be able to just pick up those [low] frequencies." This example demonstrates that through controllability, Soothe2 is able to be useful outside its original design brief.

3.3.3 Ozone

P3 appropriates the mastering tool Izotope Ozone, using the plugin as a reference to overcome environmental issues. "Say you're mixing in a different room, you hit the mastering assistant and it tells you if you have too much low end or two much high end very quickly." Here, rather than using Ozone as a solution to balance frequencies, it may allow the professional to quickly diagnose issues when they cannot trust their ears. "I don't use it to turn it on and then leave it there… I then go back and fix it [the frequency spectrum] to where it's supposed to be." Conceptually, this appropriation relates to the primary concern of all professionals in the study: speed and efficiency. Ozone is able to supplement the mixer's ears, allowing them to be confident without worrying about the impossible task of learning the acoustic idiosyncrasies of every room they work in.

3.3.4 XO

The line between technical help and shared creative agency in some tools is blurred. XO's form of automation can be viewed in two different ways. Firstly, as a purely technical assistant, XO can be viewed as a tool which uses automation to make the user more efficient, while still allowing them to retain full control over creative decision-making.

Secondly, XO can be viewed as a case of shared creative agency between the user and the tool. Allowing the system to decide the similarity and grouping of samples may affect which sounds a user chooses, and therefore the creative output. XLN Audio, the manufacturers of XO, encourage users "when you hear something you like, have a look around. You may find something even better close by."[2] Using these "predictive, suggestive aspects" [P2] of the tool can be considered to sign some of the user's creative agency over to the automated sorting process.

[2] https://www.xlnaudio.com/products/xo

4 Discussion

In this section we discuss the results of our ethnographic study against current scholarly knowledge. We particularly focus on tensions created in four aspects: speed versus control, trust and comfort, sharing creative agency, and industry concerns.

4.1 Speed Versus Control

"It's just so much about speed these days" [P1]. One common theme expressed by all the professionals in the study is the pressure to work quickly. Time, and equivocally in the professional world, money, is at a premium. Participants were especially receptive to automation when a tool provided noticeable time savings, even at the expense of control. P3 cites the Gullfoss intelligent EQ:[3] "It's hitting the professional the same as it's hitting the customer [consumer market] because the customer is surprised and pleased by the fact that it's so easy to use, and there's not many controls, and the pro is pleased that there's not many controls because then they can be quicker."

This time pressure is undoubtedly a specific product of the professional sphere. "I would love to be somebody who wakes up in the morning and spends half an hour working on EQ. I don't think any of us could do that at the moment" [P3]. The movement towards embracing automation may represent professionals consolidating their time and focusing on aspects of their work that are more distinctive. P2 believes that "what people hire a mix engineer or producer for is their taste… everyone's got all these tools, and so if I don't keep up... there's some sonic things that might not quite match up to what everyone else is doing." These observations expand on those of Shneiderman [5] and Roy et al. [6], specifying that automation may be most useful when it removes barriers to technical precision, allowing producers to spend more time on creative tasks.

Despite the prolific nature of opaque automated tools such as The God Particle, combining this technical efficiency-oriented style of automation with tool design that prioritises ease of use provides the highest chance of easing workflow tensions caused by automation. P3 states that "Having [a tool] that is intuitive and straightforward is the winning game to me… I think even more than what it sounds like is the interface and how

[3] https://www.soundtheory.com/gullfoss

it works, and how easily you can get from A to B." In this instance, the separate user groups identified by Sai Vanka et al. [2] can be seen to have the same goal of spending more time on creative tasks, although the threshold for these tasks is different between amateur and professional contexts.

4.2 Trust and Comfort

Being highly trained, professionals do not have a trust-based relationship with automated tools in the same way that amateur users might. "Regardless of whether it's an analog compressor or an AI-based processor, I can't trust the unit... You can't just blindly trust things to solve stuff ... I know I have to work it" [P3]. This reliance on their own ears appears to distinguish the way that professionals think about automation. Where the amateur might be unconcerned about details they don't understand [2], the professionals' priorities are simply how easy a tool is to use, and how the end product sounds. Therefore, a tool that adds compression, saturation, or EQ to a sound source may not be considered to be infringing on creative agency. However, as sometimes granular control is necessary (ibid.), when a tool's output is not satisfactory, opacity can create tension by obfuscating a solution.

Crucially, however, whether a tool's transparency actually matters to users will depend on the context of its use. For example, The God Particle remains widely used despite its opaque processing model, contrasting the design approach proposed by Dix [16]. While the option to have more insight into its processing may improve user understanding and allow a wider range of use cases, the argument could be made that users looking for this level of control are outside the target demographic for the software anyway. This case contrasts the conclusion of Roy et al. [6] that users may prefer high controllability, demonstrating that a moderate level of control may be acceptable when the accuracy of the automation is high enough.

Professionals are constantly aware of their workflow. P2 states "I'm tweaking and changing these new toys, swapping out old ones, trying the latest thing." However, this pursuit of efficiency can create tension with the fact that "knowing your tools is really important, and I think actually makes you a lot faster" [P4]. P3 mentions award-winning mixers Manny Marroquin and Serban Ghenea:[4] "Of course they have 3-4 assistants, but they mix 300 songs a year or something, so the reliability of tools that can get you from A to B in a quick and efficient manner is paramount. It's the same guys that use the early Waves[5] plugins because they just got used to them and know how to get the sound." Each participant uses common automated tools, such as Soothe2, in different ways, supporting the argument of Riemer and Johnston [17] that a tool's applications are evaluated within the user's context. Expanding the philosophies of Dix [16], and Zappi and McPherson [18] to the field of automation, we can argue that designing these tools in a controllable, appropriable way can make them amenable to a wider range of workflows, mitigating the trade-off between comfort and efficiency.

4.3 Sharing Creative Agency

While AMSs such as those developed by Koo et al. [12], Perez-Gonzalez and Reiss [13], and Steinmetz et al. [11] are not yet close to creating fully automated, professional quality mixes, they may soon provide scope for partial automation of the process as they improve. These technologies have the potential to create tensions, as participants were weary of automated tools which were perceived as infringing too heavily on creative agency. P2, P3, and P4 all refer to the importance of subjectivity and taste in their work: "What people hire a mix engineer or producer for is their taste" [P2], "I don't want somebody to take the creative approach instead of me because I mean, what's the point?" [P3] and, "You get to a point where pretty much everything you do is of an acceptable level, but the tricky part is what can you contribute that's going to take it somewhere else" [P4]. Additionally, for tools with a higher degree of automation, the assertion of Roy et al. [6] that high controllability may prevent frustrations with tools becomes even more important, especially in a subjective field such as music.

Although professionals believe that AI and automation are likely "going to change the [music] world" [P3], they hope that these changes will "lower the barrier between the idea in your head and the thing that comes out of the speakers" [P2], leaving room for the important human aspects of the art form. All participants

[4] https://www.mannymarroquin.com/ https://serbanghenea.com/

[5] https://www.waves.com/

were receptive to the idea of AI tools being able to automate menial tasks such as session prep. Tools such as Forte,[6] released since this research was conducted, have already made this a reality. As evidenced by the ubiquity of controllable tools such as Soothe2 in participants' workflows, appropriable, controllable design promoted by authors such as Dix [16], Roy et al. [6], and Riemer and Johnston [17] may allow for the greatest possible range of tool usability. However, tools such as The God Particle and XO remind us that ultimately, the threshold for where useful tools begin to infringe on natural creativity will differ between users.

4.4 Industry Concerns

"The whole shape of the industry five years from now is going to be completely different" [P2]. All participants believe that automation and especially AI will significantly impact the recording industry. Perhaps the most obvious example of this is the existential threat to the role of the assistant engineer. "It will chop off the bottom of the industry" [P2]. A large portion of the assistant's role is technical tasks such as session prep, editing, and printing. The potential of increasingly accurate automated tools to perform these tasks will make much of this work redundant.

With a central tenet of the recording industry historically being the master and apprentice approach, the reduced demand for assistants represents a potential challenge for industry newcomers. Without lower-skilled technical tasks to do, assistants may lose the opportunity to work alongside more skilled engineers, and in doing so, develop their own careers. Equally, the ease of working entirely remote [3] may democratise jobs, but make it more difficult to become established in an industry so reliant on reputation. One potential outcome may be a more efficient configuration of the industry, with jobs reallocated elsewhere. "Maybe the workload per song goes lower, and then my rates get lower, but I get through more work" [P2].

There is recognition among professionals that some aspects of the industry are more susceptible to AI than others. In particular, participants noted links between the music and film industries. "Hollywood is interesting because with those big writer strikes, it's so union based. The music industry is I guess globally a little more spread out" [P2]. The lobbying power of the film industry for protection over issues such as generative AI affords professionals more certainty over the future, where the recording industry is more susceptible to the market created by music consumption. This fact is also apparent in the market for generative AI. "I think that [generative music] will fulfil sync and restaurant music and that kind of thing, but I don't think I'm worried about generative AI tools that will replace creativity, they will come, and they will have their place" [P3]. This sentiment is similar to the thesis of Shneiderman [5], that while technology continues to offer us increasingly powerful tools, a continued focus on human interests and needs is central to sustainable technological development.

5 Conclusion

Automation in music recording and production is a multi-faceted issue, especially in the professional industry. While professionals may share common expert knowledge and techniques, the subjective nature of music, combined with different context of work, ensures that no two experts share quite the same viewpoint.

However, recurring themes were found between all participants, the most apparent being a prioritisation of speed. While professionals use a variety of different tools to achieve an efficient workflow, they were less concerned with transparency of processing than hypothesised. The most important factors for any tool were an efficient user experience, correct level of control, and sound. In some instances, it appears that the goals of the professional and amateur are aligned, allowing products to cater to both demographics, while in other cases professionals desire a more granular level of control, or tools which require a higher level of base knowledge to operate. Participants were united in the desire for AI and automation to remove time-consuming, technical tasks from their work, leaving more time for the creative aspects of their jobs. They were disinterested in automated tools which were perceived as significantly infringing on their creative agency.

Despite the complex issues and affordances of automated tools, they continue to become an integral aspect of the recording industry. If designed in a user-centric way, these tools may be able to provide a platform for new forms of experimentation and appropriation, promoting the continued diversification of music production as a practice. Opportunities for technological appropriation and the impacts of creative intrusion by

[6] https://www.forte-ai.com/

software are both issues without linear solutions, and given their implications for real-world music creation, would be attractive options for further research in the field.

References

[1] Thorley, M., "The role of failure in developing creativity in professional music recording and production," *Thinking Skills and Creativity*, 30, pp. 160–170, 2018.

[2] Sai Vanka, S., Safi, M., Rolland, J.-B., and Fazekas, G., "Adoption of AI Technology in Music Mixing Workflow: An Investigation," in *Audio Engineering Society Convention 154*, Audio Engineering Society, 2023.

[3] Thorley, M., "The rise of the remote mix engineer: Technology, expertise, star," *Creative Industries Journal*, 12(3), pp. 301–313, 2019.

[4] Fu, Y., Newman, M., Going, L., Feng, Q., and Lee, J. H., "Exploring the Collaborative Co-Creation Process with AI: A Case Study in Novice Music Production," *arXiv preprint arXiv:2501.15276*, 2025.

[5] Shneiderman, B., "Beyond intelligent machines: just do it," *IEEE software*, 10(1), pp. 100–103, 1993.

[6] Roy, Q., Zhang, F., and Vogel, D., "Automation accuracy is good, but high controllability may be better," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pp. 1–8, 2019.

[7] McDermott, J., Gifford, T., Bouwer, A., and Wagy, M., "Should music interaction be easy?" *Music and human-computer interaction*, pp. 29–47, 2013.

[8] Moffat, D., "AI Music Mixing Systems," *Handbook of Artificial Intelligence for Music: Foundations, Advanced Approaches, and Developments for Creativity*, pp. 345–375, 2021.

[9] Izhaki, R., *Mixing audio: concepts, practices, and tools*, Routledge, 2017.

[10] Pestana, P. D., Reiss, J. D., et al., "Intelligent audio production strategies informed by best practices," 2014.

[11] Steinmetz, C. J., Pons, J., Pascual, S., and Serra, J., "Automatic multitrack mixing with a differentiable mixing console of neural audio effects," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 71–75, IEEE, 2021.

[12] Koo, J., Martínez-Ramírez, M. A., Liao, W.-H., Uhlich, S., Lee, K., and Mitsufuji, Y., "Music mixing style transfer: A contrastive learning approach to disentangle audio effects," in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5, IEEE, 2023.

[13] Perez_Gonzalez, E. and Reiss, J., "A real-time semiautonomous audio panning system for music mixing," *EURASIP Journal on Advances in Signal Processing*, 2010, pp. 1–10, 2010.

[14] Steinmetz, C. J., Vanka, S., Martínez-Ramírez, M., and Bromham, G., "Deep learning for automatic mixing," *ISMIR, Dec*, 2022.

[15] Loor Paredes, M., "Emerging paradigms in music technology: valuing mistakes, glitches and uncertainty in the age of generative AI and automation," *AI & SOCIETY*, pp. 1–12, 2025.

[16] Dix, A., "Designing for appropriation," in *Proceedings of HCI 2007 The 21st British HCI Group Annual Conference University of Lancaster, UK*, BCS Learning & Development, 2007.

[17] Riemer, K. and Johnston, R. B., "Place-making: A phenomenological theory of technology appropriation," 2012.

[18] Zappi, V. and McPherson, A., "Hackable instruments: supporting appropriation and modification in digital musical interaction," *Frontiers in ICT*, 5, p. 26, 2018.

[19] Fels, S. and Lyons, M., "Interaction and music technology," in *Human-Computer Interaction–INTERACT 2011: 13th IFIP TC 13 International Conference, Lisbon, Portugal, September 5-9, 2011, Proceedings, Part IV 13*, pp. 691–692, Springer, 2011.

[20] Rosso, B., "Creativity and constraint: Exploring the role of constraint in the creative processes of new product and technology development teams

(Doctoral dissertation, The University of Michigan)," 2011.

[21] Bardzell, J. and Bardzell, S., "Cover story: Humanistic HCI," *interactions*, 23(2), pp. 20–29, 2016, ISSN 10725520, doi:10.1145/2888576.

[22] Cohen, S., "Ethnography and popular music studies," *Popular music*, 12(2), pp. 123–138, 1993.

[23] Braun, V., Clarke, V., Hayfield, N., and Terry, G., "Thematic analysis," in *Handbook of research methods in health social sciences*, pp. 843–860, Springer, 2019.

Appendix

The God Particle

The God Particle is an intelligent mix bus processor developed by Cradle in collaboration with award-winning mix engineer Jaycen Joshua.[7] Its processing includes EQ, multi-band compression, and limiting, as well as unspecified harmonics and distortion processing.

Fig. 1: The God Particle

Soothe2

Soothe2 is a semi-automated dynamic resonance suppressor manufactured by Oeksound.[8] The plugin provides users a range of controls for which frequency range to focus on, and how aggressive and sensitive the filtering should be.

Fig. 2: Soothe2

Ozone

Ozone is an intelligent mastering tool made by Izotope.[9] It listens and applies a track-specific mastering chain with various user controls.

Fig. 3: Ozone

XO

XO is a drum sampler made by XLN Audio[10] that uses AI to automatically categorise and map drum samples, allowing the user to search based on sonic similarities and providing a more musical way to categorise files.

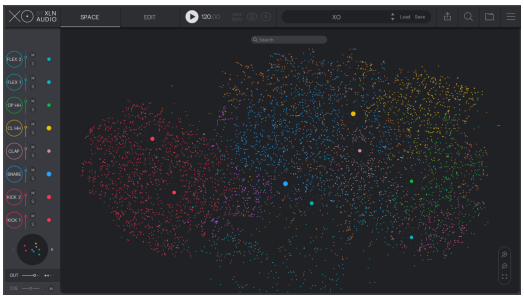


Fig. 4: XO's sample map

[7] https://cradle.app/products/the-god-particle

[8] https://oeksound.com/plugins/soothe2/

[9] https://www.izotope.com/en/products/ozone.html

[10] https://www.xlnaudio.com/products/xo

Audio Engineering Society

Conference Paper 21

Presented at the AES International Conference on Machine Learning and Artificial Intelligence for Audio
2025 September 8–10, London, UK

This paper was peer-reviewed as a complete manuscript for presentation at this conference. This paper is available in the AES E-Library (http://www.aes.org/e-lib), all rights reserved. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Journal of the Audio Engineering Society.

From CNN to Reservoir Computing: A New Perspective on Acoustic Scene Classification

Yuxuan He[1], Alireza Molla Ali Hosseini[1], Jakob Abeßer[2], Lina Jaurigue[1], Alexander Raake[1], and Kathy Lüdge[1]

[1]*Technische Universität Ilmenau, Ilmenau, Germany*
[2]*Fraunhofer IDMT, Ilmenau, Germany*

Correspondence should be addressed to Yuxuan He (yuxuan.he@tu-ilmenau.de)

ABSTRACT

Acoustic Scene Classification (ASC) is a fundamental task in machine listening, aiming to categorize the acoustic environment in which an audio recording was made. Although Convolutional Neural Networks (CNNs) are still the dominant approach in ASC today, increasing attention has been given to optimizing model efficiency, particularly for model deployment in resource-limited computational platforms such as hearing aids. In this study, we explore Reservoir Computing (RC) as an alternative lightweight approach for ASC and systematically compare its performance against multiple CNN architectures, including ResNet-34, MobileNet-V2, EfficientNet-B2, and TF-SepNet-20. Our results indicate that CNN models consistently outperform RC in classification accuracy, even when trained without data augmentation. However, RC exhibits significantly lower computational cost, requires less training time, and achieves the fastest inference speed among all models. This trade-off between accuracy and efficiency suggests that model selection should be guided by application requirements: CNNs are preferable when accuracy is paramount, while RC provides an alternative for low-power real-time applications such as hearing aids.

1 Introduction

Hearing loss is a growing global health concern, affecting more than 11% of the population in the European Union, with projections that more than 2.5 billion people worldwide will suffer from hearing impairment by 2050 [1]. Despite the availability of hearing aids, only 37-41% of individuals with hearing difficulties use these devices, mainly due to persistent issues with speech comprehension in noisy environments and the complexity of manual adjustment processes [2, 3].

Addressing these challenges, modern hearing aid technology incorporates adaptive signal processing systems to enhance speech perception in complex acoustic environments [4]. A key objective is to enable hearing aids to dynamically adjust to individual hearing profiles of users and daily auditory scenes, improve speech intelligibility in noisy conditions, and adapt to diverse listening environments [3]. A fundamental functionality in achieving this adaptability is Acoustic Scene Classification (ASC), which involves recognizing and categorizing environmental soundscapes into predefined classes, such as shopping malls, metro stations, or urban streets [5]. Incorporating ASC into hearing aids allows real-time adaptation to changing acoustic scenes, improving speech intelligibility and user comfort. For example, an ASC module can detect a

transition from a quiet office to a noisy street and automatically adjust the noise suppression and microphone directivity accordingly [3, 6].

However, the development of ASC models for hearing aids introduces a critical trade-off between accuracy and computational efficiency. Hearing aids operate under strict resource constraints, requiring models that achieve robust scene recognition while minimizing memory usage and power consumption [7, 8]. Therefore, the design and optimization of learning architectures play a pivotal role in determining the overall performance of ASC systems. Convolutional Neural Networks (CNNs) have proven to be highly effective in developing ASC systems [9]. Given the increasing demand for efficient audio classification models, recent research has focused on exploiting the powerful feature extraction capabilities of CNNs to improve both the accuracy and computational efficiency of ASC systems [7]. However, while CNNs remain the most popular choice, recent research suggests that Reservoir Computing (RC) offers a lightweight alternative [10], especially suitable for neuromorphic sensor integration.

RC is an efficient framework for processing time-dependent data, utilizing the transient dynamics of a fixed, nonlinear recurrent network (known as the reservoir). Near bifurcation points, the inherent nonlinearity of the system enables rich transformations of input signals into high-dimensional representation. Only the output layer is trained, significantly reducing training time and computational cost compared to traditional recurrent neural networks [10, 11]. The concept was developed to overcome the challenges of training complex neural networks using a fixed and randomly connected reservoir that captures rich temporal dynamics with minimal resource use [12, 13]. Here in this work, we want to exploit the dynamic response of the sensor for ASC in a RC setup as an energy efficient alternative to the neural networks models, leveraging the temporal correlations naturally present in audio signals.

In ASC, the task is to identify the environment from an audio recording, which requires capturing both temporal and non-linear sound characteristics. RC is well suited for this purpose because it naturally processes sequences and highlights dynamic features such as amplitude changes, frequency shifts, and temporal patterns, improving the distinction between scenes such as busy streets, parks, and bustling airports [14]. Moreover, recent advances allow the reservoir to be implemented with physical devices such as lasers or memristors. These hardware reservoirs can perform the high-dimensional projection mechanically, further reducing computational costs. Optimizing hyperparameters directly on such devices offers a promising path toward faster, more energy-efficient, and potentially more accurate audio scene classification systems [15, 16, 17, 18, 19].

Selecting the optimal learning architecture is crucial for achieving accurate and efficient ASC performance in hearing aids. This study makes the following contributions: (1) we introduce a novel RC model for ASC, offering an alternative to conventional deep learning approaches; (2) we systematically compare the RC model with four CNN architectures—ResNet-34, EfficientNet-B2, MobileNet-V2, and TF-SepNet-20—evaluating their performance; (3) we provide a detailed analysis of the trade-offs between classification accuracy, computational complexity, and efficiency, highlighting how different architectures align with the constraints of real-world hearing aid applications.

2 Methods

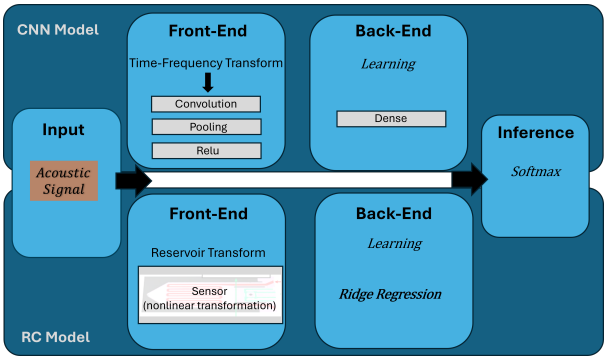


Fig. 1: Overview of the ASC processing pipeline.

To systematically compare CNN-based and RC-based approaches for ASC, we adopt a structured framework that decomposes each model into two key components: a front-end for feature extraction and a back-end for classification, as illustrated in Fig. 1.

In the front-end of an ASC model, raw audio input is transformed into a structured representation suitable for machine learning. In the case of common CNN-based models, this process involves applying the Short-Time Fourier Transform (STFT) followed by a Mel-scaled

filter bank transformation and a logarithmic magnitude compression. Such Mel spectrograms are then processed by a series of convolutional layers, pooling operations, and non-linear activations, progressively extracting relevant acoustic features [20, 21]. Conversely, in the RC-based approach, the front-end consists of a reservoir transformation where the audio waveform is fed into a nonlinear dynamic system (the sensor). A Fast Fourier Transform (FFT) is then applied to extract frequency-domain features, forming the representation used for classification.

The back-end of an ASC system is responsible for learning a mapping from the extracted features to predefined scene categories. For CNNs, this stage consists of fully-connected dense layers followed by a Softmax classifier, which assigns probabilities to different scene classes [6]. In contrast, the RC model employs ridge regression as a lightweight classification method, using the transformed reservoir responses for the final scene classification.

The following sections provide a detailed description of both CNN-based and RC-based ASC systems. Section 2.1 outlines the architecture and processing pipeline of the CNN approach, while Section 2.2 details the implementation of RC for ASC.

2.1 Convolutional Neural Network

Convolutional Layers. The convolutional layer is a core building block of CNNs. The convolution operation can be expressed mathematically as:

$$S_{i,j} = (\mathbf{K} * \mathbf{X})_{i,j} = \sum_{u=1}^{k} \sum_{v=1}^{k} K_{u,v} \cdot X_{i+u,j+v} \qquad (1)$$

where $\mathbf{K} \in R^{k \times k}$ is the convolutional kernel, and i, j denote the spatial position in the output feature map.

Several important parameters govern the behavior of the convolution operation, shaping the characteristics of the resulting feature maps. One such parameter is the **stride** (s_k), which determines the step size at which the kernel moves across the input [22]. Another parameter is **zero-padding** (p_k), which involves symmetrically adding rows and columns of zeros around the input to control the output size [23]. Additionally, the **dilation rate** (d_k) significantly influences the receptive field by introducing gaps between kernel elements [24].

Receptive Field. The receptive field of a neuron in a CNN refers to the region in the input space that contributes to its activation [25]. It determines the context in which a neuron can "see" at a specific layer.

Pooling Layers. Pooling layers downsample the feature maps, reducing their spatial dimensions while retaining essential information [26]. Two common pooling techniques are **Max Pooling** and **Average Pooling**. Max pooling retains the maximum value within each pooling window, emphasizing the most salient features [27]. Average pooling computes the average value within the pooling window, smoothing the feature map:

$$S_{i,j} = \frac{1}{k^2} \sum_{u=1}^{k} \sum_{v=1}^{k} X_{i+u,j+v} \qquad (2)$$

2.1.1 Feature Extraction for CNN

To effectively process audio inputs for our CNN models, as illustrated in Figure 1, we first apply a preprocessing pipeline to extract time-frequency representations. The input audio is provided in a single-channel format with a sampling rate of 44.1 kHz. Specifically, the STFT is performed using a window size of 3072 samples (64 ms) and a hop size of 500 samples (10.42 ms). Log-Mel spectrograms are computed using a Mel-scaled filter bank with 256 frequency bands and 4096 FFT points. Each spectrogram is stored as a three-dimensional tensor $\mathbf{X} \in R^{256 \times 84 \times 1}$, capturing 256 frequency bins, 84 time frames, and one channel which is processed by the CNN model.

2.1.2 CNN Architectures

As a performance comparison for the RC model in ASC, we evaluated four commonly used CNN architectures: ResNet-34 [28], EfficientNet-B2 [29], MobileNet-V2 [30], and TF-SepNet-20 [31].

ResNet-34 ResNet [28] is a family of deep CNNs designed to overcome the vanishing gradient problem through residual connections, which facilitate stable training in deep architectures. Among its variants, ResNet-34 adopts a bottleneck-free residual block design, preserving spatial and temporal details while maintaining a moderate parameter size. In this study, ResNet-34 is selected for its strong feature extraction capabilities and effective hierarchical modeling of spectro-temporal patterns, making it a well-suited candidate for ASC tasks [6, 32].

MobileNetV2 MobileNet [33] is a family of lightweight CNNs designed for efficiency, particularly in resource-constrained environments. It employs depthwise separable convolutions to reduce computational complexity, while maintaining strong feature extraction capabilities. Among its variants, MobileNetV2 [30] introduces inverted residual blocks, which enhance representational power with minimal computational cost. In this study, we adopt MobileNetV2 for its balance between accuracy and efficiency [34, 35].

EfficientNet-B2 EfficientNet [29] is a family of CNNs designed for optimal model scaling through compound scaling, which balances depth, width, and resolution to improve accuracy while minimizing computational costs. Among its variants, EfficientNet-B2 offers an trade-off between efficiency and performance, making it particularly suitable for ASC tasks that require spectro-temporal pattern recognition with low computational overhead. In this study, EfficientNet-B2 is selected for its lightweight architecture, swish activation for improved gradient flow, and mobile inverted bottleneck convolution (MBConv) layers for enhanced efficiency [35, 36].

TF-SepNet-20 TFSepNet [31] is an efficient CNN architecture designed to enhance feature extraction while minimizing computational cost. Unlike conventional CNNs, it employs time-frequency separable convolution (TF-SepConv), where 1D convolutional kernels process spectral and temporal features independently before merging them, optimizing ASC-related feature extraction. The architecture is inspired by BC-ResNet [37] but replaces 2D convolutions with 1D kernels along time and frequency axes, reducing parameter counts and MACs consumption. Additionally, depthwise separable convolutions and adaptive residual normalization improve efficiency and generalization. The global average pooling layer ensures a compact final representation while maintaining classification performance [6, 31].

2.1.3 Training Setup

The training process involves optimizing the model for 100 epochs using the Adam optimizer with default hyperparameters within the PyTorch framework. The cross-entropy loss function is used, as it is standard for multi-class classification tasks. A batch size of 32 is used. To facilitate faster convergence and to improve training stability, a warm-up learning rate strategy is adopted [38]. Specifically, the learning rate is gradually increased from 0 to 0.01 over the initial five epochs. After the warm-up phase, the learning rate is progressively reduced to 0 following a cosine annealing schedule [39]. Furthermore, early stopping is used to prevent overfitting, using 20% of the training data as validation data, with a patience setting of 15 epochs. Training is stopped if validation performance does not improve for 15 consecutive epochs [40]. Moreover, it is important to note that the CNN architectures in this study were trained without data augmentation.

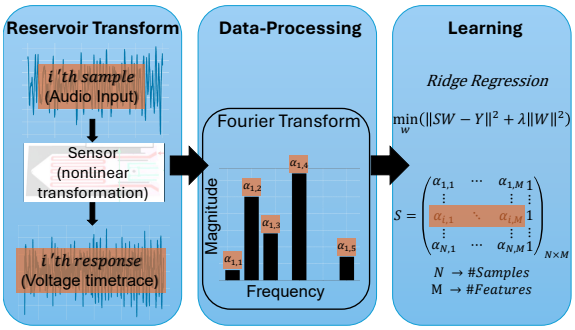


Fig. 2: The overall pipeline for the sensor based RC method using a MEMS-sensor for ASC.

2.2 Reservoir Computing Model Design

The overall pipeline of the RC method is shown in Figure 2. In the following we explain each component in detail.

Sensor-Reservoir Transform. We simulate the dynamics of a microelectromechanical (MEMS) sensor using a physics-based model derived from the Euler-Bernoulli beam theory, as detailed in [15]. This analytical model serves as the reservoir in our approach, transforming input signals through the nonlinear, auditory-like dynamic responce of the MEMS device. The MEMS sensor is described by the following set of equations:

$$\ddot{x}(t) + \frac{\omega_0}{Q_0}\dot{x}(t) + \omega_0^2 x(t) = \alpha\,\theta(t) + F_{\text{ext}}(t) \tag{3}$$

$$\dot{\theta}(t) + \beta\,\theta(t) = \frac{\gamma \cdot \min((a\,u_{\text{ac}}(t) + u_{\text{dc}})^2, u_{\text{max}}^2)}{R^2} \tag{4}$$

$$\dot{u}_{\text{ac}}(t) = -\frac{u_{\text{ac}}(t)}{\tau} + k\dot{x}(t) \tag{5}$$

Here $x(t)$, $\theta(t)$, and $u_{ac}(t)$ are beam deflection, temperature change and actuation voltage and describe the time dependent oscillation state of the sensor. The sensor is designed to mimic the behavior of the cochlea by converting external forces (i.e., sounds modelled by F_{ext}) into voltage time series $u_{ac}(t)$ using piezoresistive elements and an in-time feedback loop (feedback strenth a in Eq.(4)) for improved adaptation to its surroundings. The parameters $\alpha, \omega_0, Q_0, \gamma, \beta, \tau, k, R, u_{dc}$, $u_{\max}$ are conversion factor (deflection $x(t)$ to sensing voltages), angular resonance frequency, quality factor, conversion factor (actuation voltage u_{ac} to temperature θ), time constant (temperature changes), time constant for high pass filter, calibration factor (deflection $x(t)$ to sensing voltage), heater resistance, bias voltage, and cutoff voltage, respectively, and chosen to model the actual device from [15].

We implement the parameters for an existing sensor [15] and derive guideline for experimental implementation within a RC setup. We focus on the effect of the feedback strength a and the dc voltage u_{dc}.

To fully understand the sensor's behavior, its dynamics are first analyzed in the absence of external forces. This analysis includes investigating resonance frequencies, bifurcations, and overall sensor responses, which are crucial for effective hyperparameter optimization. It has already been shown that the sensor dynamics exhibits a Hopf bifurcation in (u_{dc},a)-parameter space, beyond which self sustained oscillations occur[41]. In Fig.3a, the steady state amplitude x of the sensor response is color coded with grey while the number of maxima observed in x in the self oscillatory regimes is depicted with blue/pink colors. The Hopf bifurcation is visualized via white and yellow dashed lines as determined numerically from our simulations. The yellow and white dashed lines indicate the dynamic transitions from steady state to the self-oscillatory behavior in the low-frequency and high-frequency regimes, respectively, resulting from a Hopf bifurcation. Exemplary time series of the sensor deflection $x(t)$ are shown in Fig.3b,c.

Comparing positive and negative feedback strength, we find that the sensor exhibits two distinct dynamic regimes which are: 1) A regime showing high resonance frequency near 14 kHz for negative a (timeseries in Fig. 3c), and 2) A regime with a complex dynamic response consisting of low frequencies (from 40 Hz to 250 Hz) and the 14 kHz component (timeseries in Fig. 3b).

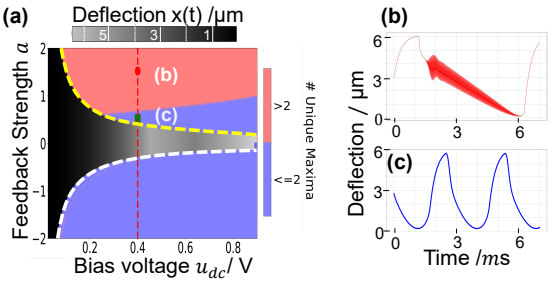


Fig. 3: Sensor dynamics (a) Unique maxima found in the sensor deflection $x(t)$ color coded in blue/red as a function of bias voltage $u_{\rm dc}$ and feedback strength a. Grey colors depict deflection value for the case without oscillations. White and yellow dashed lines represent the Hopf-bifurcation. (b) Time series of $x(t)$ with both high and low frequency components at $u_{\rm dc} = 0.4$, $a = 0.52$, (c) $x(t)$ with only low-frequency components at $u_{\rm dc} = 0.4$, $a = 1.52$.

Simulation and Data-Processing. We simulate the RC performance of the sensor using an audio scene dataset composed of one-second mono recordings at a 44,100 Hz sample rate. These recordings serve as the external force $F_{ext}(t)$ that acts on the sensor. The dynamic response of the sensor is given by the ODE system presented in Eqs.(3)-(5) which we simulate using a fourth-order Runge-Kutta method with a time step of 10^{-6} which is the largest step size that fully captures the system's dynamics without loss of accuracy. The audio samples are first interpolated, passed through a low-pass filter to eliminate aliasing effects, and then fed into the ODE system in order to obtain the sensor response $u_{\rm ac}$. Subsequently we post-process the response $u_{\rm ac}$ by applying a Fast Fourier Transform (FFT) to time series where each has a length of 1×10^6. Each sequence i of a Fourier transformed time series is one sample (number of samples is N). The FFT is given by

$$X(k) = \sum_{n=0}^{N'-1} x(n)\, e^{-j\frac{2\pi kn}{N'}}, \quad k = 0, 1, \ldots, N'-1, \quad (6)$$

where $N' = 10^6$ is the length of each time series and n is the time index (the time between n and n+1 ,which is one microsecond). We then compute the magnitude $|X(k)|$ via $|X|^2 = (\mathrm{Re}X)^2 + (\mathrm{Im}X)^2$ for each frequency k, where each k represents one feature of the sensor

response. The total number of features M will be subject to change as we can select different numbers of relevant features for classification.

For the classification task, we select M frequencies (features) that are most informative for distinguishing between classes. The M magnitudes $|X(k)|$ (denoted as α_i for sample i in Fig.2) form the feature vector for each sample. The resulting state matrix S has M columns (features) and N rows (samples).

Learning via Ridge Regression. Once the state matrix S is filled with the simulated and Fourier transformed sensor responses, we normalize the obtained data so that each feature (each column in S) has a zero mean and unit variance. We then append an additional column of ones to the state matrix S to serve as a bias term[42]. We use RC for the training and thus we will need 10 ridge regression with Tikhonov regularization (parameter λ) to classify the data into ten distinct classes. The trained weights that determine the output of the reservoir computer are called weight matrix W and are determined from solving the optimization problem $\min_W \|Y - SW\|_2^2 + \lambda \|W\|_2^2$ with Y being the target vector. The closed-form solution is given by

$$\hat{W} = \left(S^T S + \lambda I\right)^{-1} S^T Y. \tag{7}$$

a winner-takes-all strategy is then applied to determine the final predicted class for each sample. Ridge regression is chosen for its computational simplicity and efficiency, as it avoids backpropagation and iterative training, making it suitable for fast and low-power applications.

Hyperparameter Optimization. In the RC approach, two key groups of hyperparameters are optimized: the sensor parameters which are bias voltage u_{dc} and feedback strength a, and the model parameters which are ridge regression regularization parameter λ and the number of features M. In order to find the optimal sensor parameters u_{dc} and a, we perform a grid search. Fig. 4 shows the performance values achieved when scanning the u_{dc}-a-parameter plane. Within the steady state regions (grey colors in Fig.3) we find the best performance (red colors in Fig.4) . Also, slight improvements are seen when the operation point is closer to the transition point from steady state to low-frequency regime (yellow line in Fig.4).

Then, with the optimized sensor parameters, we optimized the model parameters. We first tune λ to maximize the classification accuracy using ridge regression

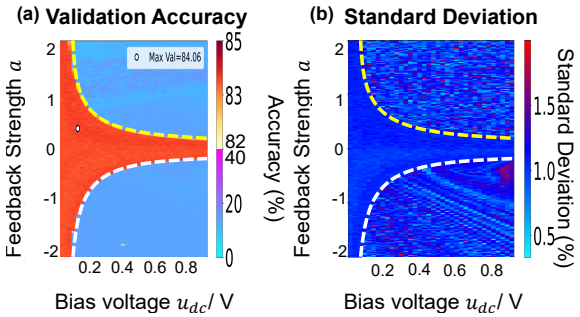


Fig. 4: Sensor Parameters (a) Average validation accuracy (from a 10-fold cross-validation) of the RC model as a function of sensor settings—bias voltage u_{dc} and feedback strength a. The white and yellow dashed lines mark the Hopf bifurcation, and the model uses $M = 24000$ features. (b) The corresponding standard deviation of these accuracies; note that the high-accuracy (red) region in (a) has a standard deviation of 1.

on the full feature set. Then, for the selected λ, the top M features are selected by analyzing the trained weights per class: weights are split into positive and negative groups, their medians are computed, and features exceeding a scaled median threshold are retained. This targeted pruning isolates the most influential features, yielding a compact representation without compromising accuracy.

3 Evaluation and Result

3.1 Dataset

In this work, we use the TAU Urban Acoustic Scenes 2022 Mobile development dataset [14], which is a well-established benchmark to assess the performance of low-complexity ASC systems. The dataset contains audio recordings from four different devices. However, for our experiments, we exclusively use the recordings from device A. This device captures the highest-quality audio using a Soundman OKM II Klassik/studio A3 binaural microphone and a Zoom F8 recorder. The decision to limit our analysis to a single device was made to minimize variability caused by different recording setups, ensuring that any observed performance gains are driven solely by the proposed model architectures. This approach allows us to isolate the impact of the model design on classification accuracy without introducing

domain shifts or device-specific effects. We follow the official 7:3 training/test split for our experiments.

3.2 Experimental Design

To systematically evaluate the performance of the RC model in comparison with CNN architectures under different data conditions, we design two experiments. The first experiment investigates model performance under limited data diversity by restricting the training set to a single city, specifically Barcelona, from the dataset. The training data is split into 80% for training and 20% for validation, and the same split is applied across all models to ensure a fair comparison. This setting allows us to assess the performance of different models when data is scarce. The second experiment examines the effect of increasing data diversity. We expand the training data to include recordings from three and five different cities, enabling us to analyze how increasing the number of cities in the training data impacts the performance of the RC model compared to CNN architectures. Both experiments are conducted using the four CNN models ResNet-34, EfficientNet-B2, MobileNetV2, and TF-SepNet, as described in Section 2.1.2. This setup allows us to explore the relationship between model size and classification ability across different data conditions and to better understand whether the RC model is particularly advantageous when training data is limited.

3.3 Computational Complexity

When comparing the computational complexity of RC and CNN models, it is important to acknowledge that the methods for calculating the number of parameters and Multiply-Accumulate operations (MACs) differ significantly due to their distinct architectural structures.

Parameter and MACs Calculation in CNNs. In CNN, the total parameter count is obtained by summing up the parameters from all convolutional and fully connected layers [43]. The total MACs is defined as the sum of preprocessing MACs and CNN MACs. The preprocessing MACs account for the operations required to transform raw audio waveforms into log-Mel spectrograms, which serve as inputs to the CNN. Moreover, to facilitate the calculation of MACs in our CNN models, we employed the open-source tool *NeSsi* [43].

Parameter and MACs Calculation in RC. In the ridge regression classifier for a 10-class problem in RC for this work, the state matrix S is composed of N rows (samples) and M columns (features). To account for the bias term, an additional column of ones is appended to S, resulting in an effective feature dimension of $M+1$. Consequently, the weight matrix W that maps the augmented feature space to the 10 output classes has dimensions $(M+1)\times 10$. Therefore, the total number of parameters in the model is given by $(M+1)\times 10$. The total MACs in RC is also defined as the sum of data-processing MACs and RC model MACs. The data-processing MACs account for the operations required to apply the real FFT on sensor responses [44]. Moreover, in order to have similar measure as the CNN models, the calculation of MACs in RC model also employed the open-source tool *NeSsi* [43].

3.4 Results

The results of **Experiment 1** are summarized in Table 1. Regarding validation accuracy, all three CNN architectures exhibit consistently high performance, with values exceeding 95%. In contrast, the RC model shows a lower validation accuracy of 84.49% using the optimal configuration ($u_{dc}=0.1, a=-1.08, \lambda=1e4, M=1260$). In terms of test accuracy, the CNN models also demonstrate stable performance, with all three basic architectures achieving results above 60%. TF-SepNet-20, while having a lower test accuracy than the other CNNs, still achieves 56.00%. The RC model, on the other hand, reaches a test accuracy of 44.50%. Regarding model complexity, both TF-SepNet-20 (0.016M) and the RC model (0.012M) exhibit significantly lower parameter counts compared to the other three CNN architectures. In terms of inference time (IT), among the CNN-based models, ResNet-34 achieves the fastest inference, requiring only 3.93 ms. This is faster than MobileNet-V2 (7.15 ms) and EfficientNet-B2 (10.87 ms). However, the most efficient model overall is RC, with an IT of only 1.70 ms, highlighting its suitability for resource-constrained scenarios like hearing aids.

The results of **Experiment 2**, illustrated in Figure 5. For validation accuracy, the CNN models show consistently high performance, maintaining values around 90% or higher under different training data conditions. In contrast, the RC model shows a noticeable decline in validation accuracy as the diversity of training data increases. Starting from 84.49% with 1 city, its validation accuracy drops to 72.38% with 3 cities and further decreases to 66.06% with 5 cities. In terms of test accuracy, all four CNN models remain relatively stable,

Architecture	Params (M)	MACs (G)	TT (s)	IT (ms)	Valid. Acc. (%)	Test Acc. (%)
ResNet-34	21.283	29.81	266.44	3.93	95.39	62.75
EfficientNet-B2	9.214	5.82	993.13	10.87	98.10	64.75
MobileNet-V2	2.236	2.54	446.67	7.15	97.19	64.50
TF-SepNet-20	0.016	0.27	557.89	4.95	93.96	56.00
Reservoir Computing	0.012	0.01	0.21	1.70	84.49	44.50

Table 1: Experiment 1: Model Performance on a Single City Dataset. The table presents a comparative analysis of different architectures in terms of model complexity (Params, MACs), computational efficiency, and classification accuracy (Validation Accuracy and Test Accuracy). TT refers to Training Time, and IT refers to Inference Time.

fluctuating around 60%, with minor variations across different training conditions. The RC model also exhibits relatively stable test accuracy, maintaining values around 45%. However, unlike its validation accuracy, the RC test accuracy shows a slight upward trend as the diversity of training data increases, improving from 44.50% with 1 city to 49.75% with 5 cities.

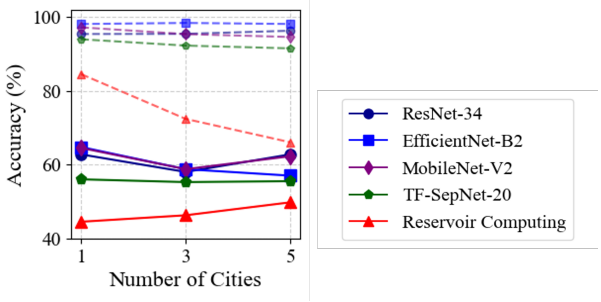


Fig. 5: Experiment 2: Impact of Increasing Data Diversity on Model Performance. Test accuracy (%) and validation accuracy (%) trends across different city data splits. Solid lines represent test accuracy, while dashed lines indicate validation accuracy.

4 Conclusion

The experimental results highlight the trade-off between classification performance and computational efficiency in ASC models. The CNN models consistently outperformed RC in classification accuracy, with all four CNN variants achieving test accuracy around 60%, while the RC model maintaining test accuracy around 45–50%. However, the RC model demonstrated remarkable efficiency in both TT and IT. Unlike CNNs, which require extensive training, the RC model required less than 1s for training across all experimental settings. Its IT also remained lower than that of CNNs, making it an attractive option for real-time applications with strict computational constraints. Such low inference latency, combined with its lightweight nature, makes the RC model particularly suitable for deployment in hearing aids, where energy efficiency and real-time responsiveness are critical.

Rather than presenting a best model, our findings suggest two viable solutions depending on system requirements: CNNs for accuracy-driven applications and RC for efficiency-critical scenarios. This flexibility can be beneficial, as it allows system designers to tailor ASC implementations according to specific needs. The ability to select between a high-accuracy model and an ultra-efficient model provides a pathway for optimizing ASC deployment in practical applications, particularly for hearing assistive technologies where balancing accuracy and computational cost is crucial.

This work presents an initial and straightforward step towards developing an RC model for ASC. While our results are promising, there is significant potential for further improvement. Moreover, adapting RC to other tasks may require careful reconfiguration of the sensor setup, as the reservoir behavior is tied to the physical sensor's dynamics. Therefore, task-specific tuning and modeling are essential. Future research may focus on optimizing the sensor setup and incorporating multiple sensors in an optimal configuration. Such advancements could lead to enhanced performance, potentially allowing the system to operate directly on the raw reservoir time series responses without the need for extensive intermediate data processing.

References

[1] W. H. Organization, "World report on hearing," *World Report on Hearing*, 2021.

[2] S. Kochkin, "Marketrak vii: Obstacles to adult non-user adoption of hearing aids," *The Hearing Journal*, vol. 60, p. 24–51, 2007.

[3] R. Vanitha Devi and Vasundhara, "Review on recent advances in hearing aids: A signal processing perspective," in *Proceedings of the International Conference on Paradigms of Computing, Communication and Data Sciences* (R. P. Yadav, S. J. Nanda, P. S. Rana, and M. H. Lim, eds.), Algorithms for Intelligent Systems, Springer, Singapore, 2023.

[4] C. Lenk, K. Ved, S. Durstewitz, T. Ivanov, M. Ziegler, and P. Hövel, "Bio-inspired, neuromorphic acoustic sensing," in *Bio-Inspired Information Pathways* (M. Ziegler, T. Mussenbrock, and H. Kohlstedt, eds.), vol. 16 of *Springer Series on Bio- and Neurosystems*, Springer, Cham, 2024.

[5] D. Barchiesi, D. Giannoulis, D. Stowell, and M. D. Plumbley, "Acoustic scene classification: Classifying environments from the sounds they produce," *IEEE Signal Processing Magazine*, vol. 32, pp. 16–34, May 2015.

[6] B. Ding, T. Zhang, C. Wang, G. Liu, J. Liang, R. Hu, Y. Wu, and D. Guo, "Acoustic scene classification: A comprehensive survey," *Expert Systems with Applications*, vol. 238, p. 121902, 2024.

[7] Y. Li, W. Cao, W. Xie, Q. Huang, W. Pang, and Q. He, "Low-complexity acoustic scene classification using data augmentation and lightweight resnet," in *2022 16th IEEE International Conference on Signal Processing (ICSP)*, vol. 1, pp. 41–45, October 2022.

[8] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.

[9] K. J. Piczak, "Environmental sound classification with convolutional neural networks," in *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6, September 2015.

[10] W. Maass, T. Natschläger, and H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," *Neural. Comput.*, vol. 14, p. 2531, Nov. 2002.

[11] H. Jaeger, "The "echo state" approach to analysing and training recurrent neural networks," GMD Report 148, GMD - German National Research Institute for Computer Science, 2001.

[12] L. Larger, A. Baylón-Fuentes, R. Martinenghi, V. S. Udaltsov, Y. K. Chembo, and M. Jacquot, "High-speed photonic reservoir computing using a time-delay-based architecture: Million words per second classification," *Phys. Rev. X*, vol. 7, p. 011015, February 2017.

[13] M. Nakajima, K. Tanaka, and T. Hashimoto, "Scalable reservoir computing on coherent linear photonic processor," *Commun. Phys.*, vol. 4, no. 20, p. 2399, 2021.

[14] T. Heittola, A. Mesaros, and T. Virtanen, "Tau urban acoustic scenes 2022 mobile, evaluation dataset," June 2022.

[15] C. Lenk, P. Hövel, K. Ved, S. Durstewitz, T. Meurer, T. Fritsch, A. Männchen, J. Küller, D. Beer, T. Ivanov, and M. Ziegler, "Neuromorphic acoustic sensing using an adaptive microelectromechanical cochlea with integrated feedback," *Nat. Electron.*, vol. 6, pp. 370–380, May 2023.

[16] P. Antonik, F. Duport, M. Hermans, A. Smerieri, M. Haelterman, and S. Massar, "Online training of an opto-electronic reservoir computer applied to real-time channel equalization," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, p. 2686, August 2016.

[17] L. Larger, M. C. Soriano, D. Brunner, L. Appeltant, J. M. Gutierrez, L. Pesquera, C. R. Mirasso, and I. Fischer, "Photonic information processing beyond turing: an optoelectronic implementation of reservoir computing," *Opt. Express*, vol. 20, no. 3, pp. 3241–3249, 2012.

[18] K. Vandoorne, P. Mechet, T. van Vaerenbergh, M. Fiers, G. Morthier, D. Verstraeten, B. Schrauwen, J. Dambre, and P. Bienstman, "Experimental demonstration of reservoir computing on a silicon photonics chip," *Nat. Commun.*, vol. 5, p. 3541, Mar. 2014.

[19] B. Barazani, G. Dion, J.-F. Morissette, L. Beaudoin, and J. Sylvestre, "Microfabricated neuroaccelerometer: Integrating sensing and reservoir computing in mems," *J. Microelectromech. Syst.*, vol. 29, no. 3, pp. 338–347, 2020.

[20] V. K. Singh, K. Sharma, and S. N. Sur, "A survey on preprocessing and classification techniques for acoustic scene," *Expert Systems with Applications*, vol. 229, p. 120520, November 2023.

[21] J. Abesser, "A review of deep learning based methods for acoustic scene classification," *Applied Sciences*, vol. 10, p. 2020, March 2020.

[22] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," *arXiv preprint arXiv:1603.07285*, 2016.

[23] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, and T. Chen, "Recent advances in convolutional neural networks," *Pattern Recognition*, vol. 77, pp. 354–377, May 2018.

[24] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *CoRR*, vol. abs/1511.07122, 2015.

[25] W. Luo, Y. Li, R. Urtasun, and R. S. Zemel, "Understanding the effective receptive field in deep convolutional neural networks," *ArXiv*, vol. abs/1701.04128, 2016.

[26] M. Lin, Q. Chen, and S. Yan, "Network in network," *CoRR*, vol. abs/1312.4400, 2013.

[27] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer Vision – ECCV 2014* (D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, eds.), vol. 8689 of *Lecture Notes in Computer Science*, pp. 818–833, Springer, Cham, 2014.

[28] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2015.

[29] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," *ArXiv*, vol. abs/1905.11946, 2019.

[30] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520, 2018.

[31] Y. Cai, P. Zhang, and S. Li, "Tf-sepnet: An efficient 1d kernel design in cnns for low-complexity acoustic scene classification," in *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 821–825, 2024.

[32] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. J. Weiss, and K. Wilson, "Cnn architectures for large-scale audio classification," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 131–135, 2017.

[33] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *ArXiv*, vol. abs/1704.04861, 2017.

[34] A. G. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, "Searching for mobilenetv3," *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1314–1324, 2019.

[35] X. Zhang, W. Xie, and Y. Gong, "Lightweight acoustic scene classification using knowledge distillation and efficient cnns," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 30, pp. 2055–2067, 2022.

[36] I. J. Ananya, S. Suad, S. H. Choudhury, and M. A. Khan, "A comparative study on approaches to acoustic scene classification using cnns," *ArXiv*, vol. abs/2204.12177, 2022.

[37] B. Kim, S. Chang, J. Lee, and D. Sung, "Broadcasted residual learning for efficient keyword spotting," in *Interspeech*, 2021.

[38] P. Goyal, P. Dollár, R. B. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, "Accurate, large minibatch sgd: Training imagenet in 1 hour," *ArXiv*, vol. abs/1706.02677, 2017.

[39] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," *arXiv: Learning*, 2016.

[40] L. Prechelt, "Early stopping-but when?," in *Neural Networks*, 1996.

[41] S. H. Strogatz, *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering*. Boca Raton: CRC Press, 2nd edition ed., 2015. eBook Published 23 May 2019.

[42] T. Hülser, F. Köster, L. C. Jaurigue, and K. Lüdge, "Role of delay-times in delay-based photonic reservoir computing," *Opt. Mater. Express*, vol. 12, no. 3, p. 1214, 2022.

[43] A. Ancilotto, "Nessi: Neural network simple statistics interface." `https://github.com/AlbertoAncilotto/NeSsi`, 2023. Accessed: 2025-01-31.

[44] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex fourier series," *Mathematics of computation*, vol. 19, no. 90, pp. 297–301, 1965.

Audio Engineering Society

Conference Paper 22

Presented at the AES International Conference on Machine Learning and Artificial Intelligence for Audio
2025 September 8–10, London, UK

This paper was peer-reviewed as a complete manuscript for presentation at this conference. This paper is available in the AES E-Library (http://www.aes.org/e-lib), all rights reserved. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Journal of the Audio Engineering Society.

Improvement and Cross-Domain Evaluation of SlowFast Networks

Ravi Kumar[1], Sascha Grollmisch[1], and Jakob Abeßer[1]

[1] *Fraunhofer Institute for Digital Media Technology IDMT, Ilmenau, Germany*

Correspondence should be addressed to Ravi Kumar (ravi.kumar@idmt.fraunhofer.de)

ABSTRACT

Audio classification requires robust models which can capture both transient events and long-duration patterns. This poses a challenge for common single-stream convolutional neural networks that process short- and long-term events in a uniform fashion using the same kernels. Inspired by the dual-stream processing of the human auditory system, SlowFast networks separate temporal and spectral analysis into parallel pathways. In this paper, we propose several enhancements to the SlowFast network, implementing uniform separable convolutions on both the slow and fast pathways to streamline the architecture and improve efficiency, while also introducing a novel lightweight model variant with a 92 % parameter reduction. We perform a comprehensive cross-domain evaluation using eight datasets that cover speech, environmental sounds, industrial sounds, and bioacoustic sounds. The enhanced SlowFast network surpasses the original SlowFast network as well as the MobileNetV3 single-stream baseline, especially for single-label tasks, while remaining competitive in multi-label tasks. The study highlights the potential of dual-stream architectures and underscores the importance of architectural design for audio classification.

1 Introduction

Audio classification is a fundamental component in many modern applications, including speech recognition, environmental sound monitoring, and industrial quality control. In real-world scenarios, sound sources often overlap and events can occur over different time scales. For example, there may be short transient sounds, such as mechanical clicks in a factory, as well as long-lasting sounds, such as the continuous hum of machinery. Conventional single-stream convolutional neural networks (CNNs) usually use rectangular kernels consistently across the spectrogram. This restricts their ability to effectively capture patterns of different durations.

Humans, on the contrary, navigate seamlessly through such complex auditory scenes. Neuroscience research identified two primary pathways in the human auditory cortex, often referred to as the dorsal (the "where") and ventral (the "what") pathways, which allow efficient processing of temporal dynamics and semantic content [1, 2]. Inspired by this dual-stream structure, SlowFast networks [3] have been proposed to handle multiple time scales within a single model. In the context of audio, one stream (the "slow" pathway) processes lower temporal resolution and focuses on spectral content, while the other stream (the "fast" pathway) operates at a higher temporal resolution to emphasize transient features.

Despite advances in deep learning, audio classification still faces several key challenges. The temporal variability of sounds with rapid fluctuations makes pattern recognition on a single time-scale difficult. Sounds in different domains vary significantly in frequency content, duration, and structure, further complicating their recognition. Audio recordings of real-world soundscapes contain overlapping sound events and background noises, which are often affected by room-acoustic effects such as reverberation. Additionally, the spatial context of a recording plays a crucial role, as many sounds require interpretation based on their surrounding environment. For instance, a tonal beep sound may indicate an alarm in one setting and a simple button press in another. Previous work [4] has demonstrated the potential of SFNets for datasets such as EPIC-Kitchens [5] or VGG-Sound [6], but their general effectiveness across multiple audio domains ranging from speech to industrial sounds remains unexplored. Moreover, the computational demands of the proposed SFNets restrict their use on devices with limited resources.

To address the aforementioned challenges, we propose as the main contribution of this work architectural refinements and a systematic cross-domain evaluation of SFNets for audio classification. The original SFNet applies spatially separable convolutions in all layers of the fast pathway but only in certain layers of the slow pathway without detailed reasoning. Therefore, we extend the use of spatially separable convolutions across all blocks of both pathways to simplify the model architecture resulting in our proposed variant `SF-SC`. Furthermore, we introduce a novel lightweight model variant of the SFNet, the `SF-SC-small` shown in Table 1, which additionally employs depthwise separable convolutions to reduce the parameter count while maintaining classification accuracy even with less number of kernels. We validate these modifications through an extensive cross-domain evaluation using a diverse set of public datasets that span speech commands, environmental scenes, industrial sounds, and bioacoustic sounds. This broad evaluation allows us to assess how well a dual-stream network generalizes across disparate audio domains. To isolate the benefits of the dual-stream architecture, we benchmark our approach against the single-stream MobileNetV3 architecture with model variants being trained from scratch or after an initial pre-training.

The remainder of this paper is organized as follows. Section 2 reviews related work for audio classification using multi-stream strategies. Section 3 details the modifications proposed to SFNets. Section 4 describes our experimental setup and datasets. Section 5 reports and analyzes the results. Finally, Section 6 concludes the paper and outlines directions for future research.

2 Related Work

Previous work in audio classification has focused mainly on single-stream convolutional neural networks (CNNs) operating on time-frequency spectrogram input [7]. One common example that has been successfully applied to audio classification [8] is the VGG-style architecture [9], which consists of up to 19 convolutional layers with a kernel size of 3×3 with pooling and fully connected classification layers. Deeper CNN variants such as ResNet [10] introduced residual connections to alleviate vanishing gradients in training, allowing networks to grow in depth. ResNet-based architectures have been evaluated for various audio tasks, such as music instrument classification, keyword spotting, and industrial sound analysis [11, 12]. Typically, these architectures use two-dimensional convolutions with square kernels, treating both time and frequency dimensions in a uniform way. However, this symmetry may be suboptimal because temporal and frequency information in audio has inherently different characteristics, such as rapid temporal transients or stable tones.

One direction of improvement has been to modify CNN kernel shapes and factorization to handle the non-uniform time-frequency patterns. For example, rectangular $k \times m$ kernels (where $k \neq m$) have been explored as a way to separately tune frequency and temporal resolution within a single-stream model [13]. Previous studies found that the use of vertical or horizontal elongated kernels can capture frequency-specific or temporal motifs more effectively than square kernels [7]. Another related strategy is the use of spatially separable convolutions that split a 2D convolution into consecutive $1 \times k$ and $k \times 1$ convolutions. This factorization reduces computational costs while still enabling the network to learn frequency- and time-specific features. Depthwise separable convolution [14] first performs a $k \times k$ depthwise convolution independently on each channel and then applies a 1×1 pointwise convolution to fuse channel information. Because it isolates spatial and channel mixing, the depthwise variant achieves greater reductions in parameters than the spatially separable convolution, making it especially attractive for

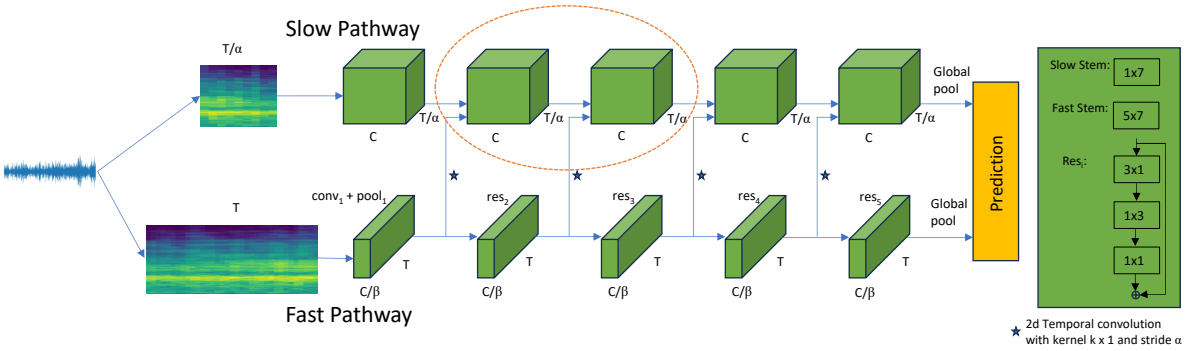


Fig. 1: Audio data processing flow in the SFNet architecture: Audio data is converted into a spectrogram, which is temporally downsampled using a striding operation in the Slow pathway (top) to process low temporal-resolution and input with the full-resolution into the Fast pathway (bottom) operating at high temporal resolution. Compared to the original SFNet architecture, which applies spatially separable convolutions (3×1 followed by 1×3 kernels) only in deeper blocks (`res4`, `res5`) of the Slow pathway, the proposed `SF-SC` uses spatially separable convolutions throughout all residual stages (modifications in dashed circle). Lateral connections (arrows) fuse Fast stream features into the Slow stream at multiple stages. The star symbol ($\star$) indicates the 2D temporal convolution (kernel $k \times 1$, stride α).

resource-constrained neural network designs, such as the MobileNetV3 [15]. However, all of these convolution variants and single-stream models do not explicitly treat temporal dynamics differently from spectral information.

To address this limitation, multi-stream architectures process the same audio input in parallel streams, each specializing in different aspects of the signal. As one prominent example, the SlowFast network (SFNet) is a dual-stream architecture originally introduced for video action recognition [3] and later adapted to audio [4]. SFNet for audio employs dual convolutional streams operating on the same log-Mel spectrogram input. The "Slow" stream focuses on spectral details using a larger number of channels, processes the input at a reduced frame rate using strided sampling, and thereby prioritizes detailed frequency analysis. In contrast, the "Fast" stream targets temporal patterns, as it operates at the full temporal resolution to capture rapidly changing audio events using fewer channels for efficiency. This dual-pathway architecture disentangles the processing of time and frequency components and allows each stream to specialize on a separate task. Both streams are fused at multiple stages using lateral connections, where intermediate outputs of the Fast stream are periodically merged into the Slow stream after scaling to match the temporal strides as shown in Figure 1. This multi-level fusion approach ensures that the Slow stream (frequency-focused) is enriched with temporal cues from the Fast stream before the final classification layer combines their outputs. In particular, the original SlowFast design uses separable convolutions in the Fast stream (splitting each 3×3 kernel into 3×1 followed by 1×3) to better isolate temporal features, while the Slow stream mostly uses regular 2D convolutions without further reasoning.

The SlowFast model has demonstrated state-of-the-art performance on the VGG-Sound dataset [4] and outperformed the Temporal Segment Networks (TSN) [16] baseline on the EPIC-KITCHENS-100 audio set [5]. Ablation studies verified a clear complementary specialization of both streams. The Slow stream alone yielded around 45% accuracy, focusing on classes with steady or slowly evolving spectral content such as "mosquito buzzing" or "sea waves", while the Fast stream alone achieved approximately 41% accuracy, excelling at rapidly changing or percussive sounds such as "woodpecker pecking" or "playing drum kit". These results underscore the benefit of separate spectral and temporal processing of audio data. However, the original SlowFast network (`SFNet`) was evaluated only on those two datasets, leaving its generalizability to other audio domains an open question.

Table 1: Comparison of SFNet variants.

Model	Spatially separable Conv2D	Depthwise separable Conv2D
`SFNet(orig)`	Last 2 blocks of slow pathway	No
`SF-SC`	All blocks	No
`SF-SC-small`	All blocks	Yes (if kernel > 1×1)

3 Proposed Modifications

We propose two modifications to the SlowFast network to simplify its architecture without sacrificing performance, which are shown in Table 1. In the first modification entitled SFNet-SeparableConv (`SF-SC`), spatially separable convolutions are applied uniformly across all layers of both the Slow and Fast streams, which streamlines the model architecture.

As a second new model variant, `SF-SC-small` extends `SF-SC` by combining spatially and depthwise separable convolutions at different stages of the network, the former (`SF-SC`) in all residual blocks to specialize along time and frequency axes, and the latter in all layers with a kernel size greater than 1×1 to improve efficiency. The `SF-SC-small` employs a [2,2,3,2] residual block configuration, denoting the number of blocks per network stage, substantially reducing the total number of parameters compared to the original `SFNet` model by 92%. The goal of this modification is to evaluate a more compact model while retaining the advantages of the SlowFast dual-stream architecture.

4 Experiments

4.1 Datasets

In order to evaluate the effectiveness of the proposed modifications to `SFNet` on a wide variety of audio signals, we use public datasets that cover everyday sounds, music, speech, and industrial sounds. Table 2 provides an overview of the key characteristics and processing setup for all datasets, including the number of classes, the audio pre-processing parameters (FFT size, hop length, sampling rate), the patch length (PL) used for the model input, the feature type and scaling (e.g., Mel spectrogram with PCEN or logarithmic scaling, MFCC), and the data augmentation techniques applied during training. For all datasets, we use the published splits into training, validation, and test sets. If no validation data was provided, we use a random 10% subset of the training data. Furthermore, we specifically analyze the performance for both single-label and multi-label classification datasets. Single-label classification assumes that each audio segment contains a single class, thus assigning one label to each instance. Conversely, multi-label classification better represents real-world scenarios with overlapping sound events, requiring the prediction of multi-hot labels for a single audio clip.

Acoustic scene classification (TUT2017) [17, 18] The TUT Acoustic scenes 2017 dataset contains recordings of 15 different acoustic scene classes (e.g., café, park, metro station). This is a single-label environmental sound classification task in which each 10-second recording is labeled with exactly one scene class. The dataset comprises 13 hours of stereo audio, with 4,680 segments for training and 1,620 for testing.

Google Speech Commands v2 (GSCv2) [19] The Google Speech Commands v2 dataset is a collection of spoken commands designed for voice interface applications, featuring 35 distinct classes that include common keywords (e.g., up, down, and stop). This is a single-label classification task and the complete dataset contains 105,829 utterances recorded from 2,618 different speakers.

Free Music Archive (FMA)[20] The Free Music Archive dataset offers a single-label classification challenge where each recording is assigned to one musical genre. We focus on the "FMA Small" subset containing 25,000 30-second segments with eight genres such as Pop, Folk, and Rock.

IDMT ISA Metal Balls Dataset (MB) [21] The IDMT ISA Metal Balls dataset contains sounds of metal balls with three types of coatings (coated, eloxed, and broken) rolling down a metal slide. This single-label dataset simulates wear inside ball bearings, comprising 2,832 files (19 minutes total).

IDMT-ISA-PUCKS Dataset (Pucks) [22] The IDMT-ISA-PUCKS dataset features recordings for industrial material analysis, which contain sounds of air

Table 2: Overview of the eight audio datasets used for cross-domain evaluation.

Dataset	Classes	SR (kHz)	FFT	Hop	PL (s)	Features	Augmentation
TUT2017	15	44.1	2048	1024	3.00	Mel spec (PCEN)	Mixup, RA, Cutout
GSCv2	35	16.0	640	320	1.00	MFCC	Mixup, RA, Cutout
FMA	8	44.1	2048	710	3.00	Magn. spec	Mixup, RA, Cutout
MB	3	32.0	2048	1024	1.00	Magn. spec (log)	Mixup, RA, Cutout
Pucks	5	32.0	2048	1024	1.00	Magn. spec (log)	BG Noise, Cutout
FSD50K	200	44.1	1024	441	1.28	Mel spec (log)	Mixup, RA, Cutout
USMv2	26	44.1	1024	441	5.00	Mel spec (log)	RA
DCASE21 T5	18	16.0	2048	1024	60.00	Mel spec (log)	Mixup, RA, Cutout

hockey pucks manufactured from four materials (plus no pucks) recorded under three different background noise levels. We train on the lowest noise level and test on the highest noise level to evaluate the performance against unknown background sounds – a common scenario in industrial sound analysis. This dataset consists of 260 one-minute recordings.

FSD50K Freesound Dataset (FSD50K) [23] The FSD50K Freesound dataset represents a comprehensive collection of 51,197 audio clips that encompass 200 diverse sound classes (e.g., urban sounds such as car horn, musical instruments such as violin, and climate sounds such as rainfall), where each clip can contain multiple sound events. This curated dataset contains over 100 hours of audio divided into 40,966 training and 10,231 test clips with variable durations (0.3–30 seconds), all verified by human annotators.

Urban Sound Monitoring v2 (USMv2) [24] The multi-label Urban Sound Monitoring v2 dataset focuses on urban environmental sound event detection, featuring 26 distinct sound categories that encompass urban infrastructure (such as cars, buses, and trucks) and environmental events (for example, rain, thunderstorm, and wind). The dataset contains 24,000 labeled sound clips totaling approximately 12 hours of audio, with recordings 5 seconds in duration.

Bioacoustics (DCASE21 T5) [25] This bioacoustics dataset from the Detection and Classification of Acoustic Scenes and Events (DCASE) Challenge 2021 Task 5 comprises several datasets of bird vocalizations in complex soundscapes. It is divided into training (14 hours 20 minutes), validation (5 hours), and evaluation sets, comprising a total of 4,996 labeled events across 19 different bird species, forming a multi-label classification problem with significant class imbalance. The evaluation set consists of 46 audio files acquired from different bioacoustic sources with varying sampling rates from 6 to 44.1 kHz, all resampled to 16 kHz by us.

4.2 Evaluation metrics

We assess model performance using file-wise accuracy (acc.) and Mean Average Precision (mAP). For single-label tasks, file-wise accuracy is the proportion of correctly classified instances, offering a clear measure of classification effectiveness.

In contrast, mAP is used in multi-label scenarios where instances may belong to multiple categories. It evaluates a model's ranking of correct detections by calculating the Average Precision for each class and averaging these across all classes, yielding a score from 0 to 100 %. A higher mAP suggests more accurate and complete detections overall.

4.3 Processing Pipeline

For all datasets, we use the same processing pipeline to ensure comparability. This includes feature extraction, as detailed in Table 2, as well as normalization using zero mean and unit variance per feature. Normalization values are calculated on the training set and applied to the validation and test set.

During training, in previous work various data augmentation strategies have been applied [22, 11, 26, 24]. These are listed in Table 2. RandAugment (RA), as described in [11], applies two randomly selected image augmentations with random magnitude, such as brightness/contrast adjustments and affine distortions, to each sample. We also incorporate Mixup [27], which blends between random pairs of samples, and Cutout

Table 3: Single-label datasets: Mean file-wise accuracy (%) and training time.

Model	TUT2017	GCSv2	FMA	MB	PUCKS	**Mean acc.**	**Mean Time**
SFNet	69.00±2.1	88.97±0.5	50.29±1.2	61.21±21.4	87.56±9.3	71.4	5 h 42 m
SF-SC	**70.29**±1.4	89.19±0.3	51.42±0.8	**76.48**±2.4	**95.22**±2.4	**76.5**	5 h 41 m
SF-SC-small	61.09±0.6	75.17±3.2	49.92±1.5	70.37±6.0	84.59±9.6	68.2	2 h 11 m
MV3(U)	13.75±12.3	90.52±0.6	51.12±2.7	33.33±0.0	20.00±0.0	41.7	3 h 13 m
MV3(P)	67.47±1.0	**91.90**±0.0	**53.12**±0.6	61.56±7.1	69.00±5.7	68.6	3 h 15 m

Table 4: Multi-label datasets: Mean mAP (%), parameter count, and training time.

Model	FSD50K	USMv2	DCASE21 T5	**Mean mAP**	**Params**	**Mean Training Time**
SFNet	44.1±0.6	38.40±0.3	43.1±2.6	41.9	26 M	8 h 37 m
SF-SC	43.3±0.8	**39.27**±0.7	42.3±0.2	41.6	26 M	8 h 11 m
SF-SC-small	37.6±0.5	37.67±0.7	43.3±0.4	39.5	2 M	3 h 55 m
MV3U	44.9±0.6	15.37±1.7	15.6±4.0	25.3	3 M	3 h 22 m
MV3P	**48.97**±0.8	35.80±0.8	**44.8**±2.3	**43.2**	3 M	3 h 03 m

[28], which masks rectangular regions in the input feature. All models are trained until convergence on the validation with a batch size of 32 and Adam optimizer with a learning rate of 0.001 on Nvidia RTX 6000 GPU hardware. It is important to note that different feature extraction parameters and augmentation strategies might lead to different results. However, the focus of this work is to compare architectures and not obtain the best results possible for each dataset.

5 Results

All results can be found in Table 3 and Table 4. One main finding is that the performance of the single-stream baseline model in form of the MobileNetV3 (MV3) strongly depends on the applied pre-training strategy. The version trained from scratch (MV3U) consistently underperformed the pre-trained version (MV3P), even though it was trained only on image data.[1] This validates that pre-training can be beneficial in general and is in line with findings for other neural network architectures [29]. Due to the large performance difference and easy availability, we decide to use the MV3P as the baseline, but need to emphasize that all SFNets are trained from scratch and pre-training could have a beneficial effect as well. Since there exist various pre-training strategies and our focus is on changes in the SFNet architecture and cross-domain performance, we keep this investigation for future work.

Although SFNet performed better than MV3U on all datasets, it is on par with MV3P on single-label and multi-label tasks, and the advantage is dataset dependent. This shows that the SFNet can perform well across different audio tasks and domains. However, it is not necessarily superior to single-stream architectures, and there is no clear pattern on which kind of data benefits from the multi-stream approach.

The proposed modification of the SFNet (SF-SC) outperforms SFNet on all single-labels tasks by 5.1 percentage points on average. Especially the relatively small industrial sound datasets (MB and Pucks) benefit from this simplified model architecture. For both datasets, we observed a lower validation loss compared to the original SFNet, indicating that separable convolutions improve generalization. Furthermore, SF-SC performs better than the MV3P baseline on the single-label tasks on average showing the benefit of dual-stream architecture even without pre-training.

On multi-label datasets, both SFNet and SF-SC perform comparably and slightly below the MV3P baseline on average. Without pre-training, there is no clear advantage in using dual-stream architectures, although all SFNets outperform the MV3U on USMV2

[1] We use the version provided in keras, which was pre-trained on ImageNet: https://www.tensorflow.org/api_docs/python/tf/keras/applications/MobileNetV3Large, last accessed 2025-03-06.

and DCASE21-T5. The general lower performance compared to single-label tasks is caused by the higher difficulty of multi-label tasks combined with the class imbalance. The majority classes had a higher mAP than the minority classes. This could be addressed using techniques such as PSLA [30] which aims to improve performance on underrepresented classes using mixup.

The `SF-SC-small` contains only about 2 million parameters, a 92% reduction in size compared to SFNet. Although the average accuracy is reduced by 8.3 percentage points compared to `SF-SC`, it still performs better than the larger `MV3U` trained from scratch, while remaining competitive with the pre-trained `MV3P` on single-label tasks. This indicates that the network design can be more important than the number of trainable parameters. For the multi-label tasks, `SF-SC-small` performs slightly better than the `SF-SC` on the DCASE21 T5 dataset, however, the mAP on FSD50k is reduced from 43.3% to 37.6% which leads to a slight decrease of 2.1 percentage points in the average mAP over all multi-label datasets. Overall, `SF-SC-small` is an efficient alternative to the original SFNet, the `SF-SC`, and the MobileNetV3 since it uses far fewer parameters while maintaining most of the initial classification performance. Furthermore, these results show that the current results cannot be solely attributed to the number of trainable parameters but rather to the architectural choices made.

6 Conclusion

Neural network architectures for automatic audio analysis are a key element of current research. Multi-stream architectures model the temporal variability of audio content using separate streams. In this paper, we simplified the dual-stream SlowFast network (SFNet) by applying uniform separable convolutions throughout both streams in the SFNet-SeparableConv (`SF-SC`). Furthermore, we investigated the effect of the model size on the performance by reducing the trainable parameters to 8% of the initial size. Experiments on eight datasets from multiple audio domains showed that all SFNet versions perform better than the single-stream MobileNetV3 trained from scratch. `SF-SC` serves as a valid and useful simplification of the original SFNet design, while the smaller SFNet version demonstrated the feasibility of multi-stream architectures for on-the-edge audio classification.

Nevertheless, pre-training the MobileNetV3 closed the gap to the SFNets, suggesting that good initial weights may be as crucial as network design itself. Thus, our future work will address pre-training SFNet versions to unlock their full potential, for example, by applying self-supervised learning approaches on large audio corpora [31]. By combining strong initial network weights with a multi-stream architecture, we aim to further enhance generalization and efficiency in diverse audio classification scenarios.

References

[1] Santoro, R., Moerel, M., Di Martino, F., Goebel, R., Ugurbil, K., Yacoub, E., and Formisano, E., "Encoding of natural sounds at multiple spectral and temporal resolutions in the human auditory cortex," *PLOS Computational Biology*, 10(1), pp. 1–14, 2014, doi:10.1371/journal.pcbi.1003412.

[2] Zulfiqar, I., Moerel, M., and Formisano, E., "Spectro-Temporal Processing in a Two-Stream Computational Model of Auditory Cortex," *Frontiers in Computational Neuroscience*, 13, 2020.

[3] Feichtenhofer, C., Fan, H., Malik, J., and He, K., "SlowFast Networks for Video Recognition," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6201–6210, 2019, doi:10.1109/ICCV.2019.00630.

[4] Kazakos, E., Nagrani, A., Zisserman, A., and Damen, D., "Slow-Fast Auditory Streams for Audio Recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 855–859, 2021, doi:10.1109/ICASSP39728.2021.9413376.

[5] Damen, D., Doughty, H., Farinella, G. M., Furnari, A., Kazakos, E., Ma, J., Moltisanti, D., Munro, J., Perrett, T., Price, W., and Wray, M., "Rescaling Egocentric Vision," *CoRR*, abs/2006.13256, 2020.

[6] Chen, H., Xie, W., Vedaldi, A., and Zisserman, A., "Vggsound: A Large-Scale Audio-Visual Dataset," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 721–725, 2020, doi:10.1109/ICASSP40776.2020.9053174.

[7] Purwins, H., Li, B., Virtanen, T., Schlüter, J., Chang, S.-Y., and Sainath, T., "Deep Learning for Audio Signal Processing," *IEEE Journal of Selected Topics in Signal Processing*, 13(2), pp. 206–219, 2019, doi:10.1109/JSTSP.2019.2908700.

[8] Hershey, S., Chaudhuri, S., Ellis, D. P. W., Gemmeke, J. F., Jansen, A., Moore, R. C., Plakal, M., Platt, D., Saurous, R. A., Seybold, B., Slaney, M., Weiss, R. J., and Wilson, K., "CNN architectures for large-scale audio classification," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 131–135, 2017, doi:10.1109/ICASSP.2017.7952132.

[9] Simonyan, K. and Zisserman, A., "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *Proceedings of the International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, 2015.

[10] He, K., Zhang, X., Ren, S., and Sun, J., "Deep Residual Learning for Image Recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, Las Vegas, NV, USA, 2016, doi:10.1109/CVPR.2016.90.

[11] Grollmisch, S. and Cano, E., "Improving Semi-Supervised Learning for Audio Classification with FixMatch," *Electronics*, 10(15), 2021, ISSN 2079-9292, doi:10.3390/electronics10151807.

[12] Kim, B., Chang, S., Lee, J., and Sung, D., "Broadcasted Residual Learning for Efficient Keyword Spotting," *arXiv preprint arXiv:2106.04140*, 2021.

[13] Huzaifah, M., "Comparison of Time-Frequency Representations for Environmental Sound Classification using Convolutional Neural Networks," *arXiv preprint arXiv:1706.07156*, 2017.

[14] Chollet, F., "Xception: Deep Learning with Depthwise Separable Convolutions," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1800–1807, 2016.

[15] Howard, A., Sandler, M., Chen, B., Wang, W., Chen, L.-C., Tan, M., Chu, G., Vasudevan, V., Zhu, Y., Pang, R., Adam, H., and Le, Q., "Searching for MobileNetV3," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1314–1324, Seoul, Korea, 2019, doi:10.1109/ICCV.2019.00140.

[16] Wang, L., Xiong, Y., Wang, Z., Qiao, Y., Lin, D., Tang, X., and Van Gool, L., *Temporal Segment Networks: Towards Good Practices for Deep Action Recognition*, p. 20–36, Springer International Publishing, 2016, ISBN 9783319464848, doi:10.1007/978-3-319-46484-8_2.

[17] Mesaros, A., Heittola, T., and Virtanen, T., "TUT Acoustic scenes 2017, Development dataset," Zenodo. `https://zenodo.org/record/400515`, last accessed 2025-03-06, 2017, doi:10.5281/zenodo.400515.

[18] Mesaros, A., Heittola, T., and Virtanen, T., "TUT Acoustic scenes 2017, Evaluation dataset," Zenodo. `https://zenodo.org/record/1040168`, last accessed 2025-03-06, 2017, doi:10.5281/zenodo.1040168.

[19] Warden, P., "Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition," *ArXiv*, abs/1804.03209, 2018.

[20] Defferrard, M., Benzi, K., Vandergheynst, P., and Bresson, X., "FMA: A Dataset for Music Analysis," in *International Society for Music Information Retrieval Conference*, 2016.

[21] Grollmisch, S., Abeßer, J., Liebetrau, J., and Lukashevich, H., "Sounding Industry: Challenges and Datasets for Industrial Sound Analysis," in *Proceedings of the European Signal Processing Conference (EUSIPCO)*, pp. 1–5, A Coruña, Spain, 2019, doi:10.23919/EUSIPCO.2019.8902941.

[22] Grollmisch, S., Johnson, D., Krüger, T., and Liebetrau, J., "Plastic Material Classification using Neural Network based Audio Signal Analysis," in *Sensor and Measurement Science International (SMSI)*, pp. 337–338, Online, 2020.

[23] Fonseca, E., Favory, X., Pons, J., Font, F., and Serra, X., "FSD50K: An Open Dataset of Human-Labeled Sound Events," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30, pp. 829–852, 2022, doi:10.1109/TASLP.2021.3133208.

[24] Abeßer, J., "Classifying Sounds in Polyphonic Urban Sound Scenes," in *Proceedings of the 152nd Audio Engineering Society (AES) Convention*, Online, 2022.

[25] Morfi, V., Stowell, D., Lostanlen, V., Strandburg-Peshkin, A., Gill, L., Pamula, H., Benvent, D., Nolasco, I., Singh, S., Sridhar, S., Duteil, M., and Farnsworth, A., "DCASE 2021 Task 5: Few-shot Bioacoustic Event Detection Development Set (2.0)," Data set, 2021, doi:10.5281/zenodo.5412896.

[26] Wallbott, P. C., Grollmisch, S., and Köllmer, T., "Examining Speaker and Keyword Uniqueness: Partitioning Keyword Spotting Datasets for Federated Learning with the Largest Differencing Method," in T. Calders, C. Vens, J. Lijffijt, and B. Goethals, editors, *Artificial Intelligence and Machine Learning*, pp. 167–177, Springer Nature Switzerland, Cham, 2023, ISBN 978-3-031-39144-6.

[27] Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D., "mixup: Beyond Empirical Risk Minimization," in *Proceedings of the International Conference on Learning Representations (ICLR)*, Vancouver, BC, Canada, 2018.

[28] DeVries, T. and Taylor, G. W., "Improved Regularization of Convolutional Neural Networks with Cutout," *arXiv preprint arXiv:1708.04552*, 2017.

[29] Koutini, K., Schlüter, J., Eghbal-zadeh, H., and Widmer, G., "Efficient Training of Audio Transformers with Patchout," in *Proceedings of the Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 2753–2757, Incheon, Korea, 2022, doi:10.21437/Interspeech.2022-227.

[30] Gong, Y., Chung, Y.-A., and Glass, J., "PSLA: Improving Audio Tagging with Pretraining, Sampling, Labeling, and Aggregation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2021, doi:10.1109/TASLP.2021.3120633.

[31] Liu, S., Mallol-Ragolta, A., Parada-Cabeleiro, E., Qian, K., Jing, X., Kathan, A., Hu, B., and Schuller, B. W., "Audio Self-supervised Learning: A Survey," *arXiv preprint arXiv:2203.01205*, 2022.

Audio Engineering Society

Conference Paper 23

Presented at the AES International Conference on Machine Learning and Artificial Intelligence for Audio
2025 September 8–10, London, UK

This paper was peer-reviewed as a complete manuscript for presentation at this conference. This paper is available in the AES E-Library (http://www.aes.org/e-lib), all rights reserved. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Journal of the Audio Engineering Society.

Improved Singing Voice Conversion with Frame-Level Content and Melody-Informed Speaker Embeddings Using Cross-Attention

Jih-Wei Yeh[1], Elaine M. Liu[1], and Yi-Wen Liu[1]

[1]*Department of Electrical Engineering, National Tsing Hua University, Taiwan*

Correspondence should be addressed to Jih-Wei Yeh (wilsun890126@gmail.com)

ABSTRACT

Singing voice conversion (SVC) converts audio sung by a source voice into the timbre of a target voice without changing the linguistic content or melody of the sung utterance. The popular SVC paradigm is to extract content features from the source and condition on timbre features extracted from the target to generate converted audio. In recent years, great strides have been made in SVC with advancements in content representations. However, what constitutes an accurate representation of timbre is still an emerging topic. Currently, most SVC systems use a single speaker embedding to represent one singer, but this disregards the large variations in timbre characteristics even for the same person when singing in different techniques and vocal ranges. In this paper, we propose a cross-attention mechanism to generate frame-wise speaker timbre embeddings informed by linguistic content and melody; we aim to generate timbre features that more accurately reflect how the voice changes with different vocal ranges and enunciations at different times. Objective metrics and subjective listening tests of our converted singing audio show improvements in enunciation clarity and voice similarity, while expressiveness is a more elusive challenge.

1 Introduction

Singing voice conversion (SVC) involves transforming source audio that includes phonetic content, rhythm, melody, and singing techniques into output audio that retains the content while adopting the timbre of a target singer. Recent studies focus on non-parallel singing voice conversion, where a sung utterance audio is reconstructed during training, and the conversion is inferred by prompting a singer ID or embedding vector. SVC has also benefited from advancements in the extraction of content-related and timbre-related acoustic features.

In the realm of content-related acoustic features, numerous pre-trained models developed for automatic speech recognition (ASR), such as phonetic posteriorgrams (PPGs) [1], Wav2Vec2.0 [2], HuBERT [3], and ContentVec [4], have proven effective in capturing phonetic content from an utterance. Another important content feature, particularly for singing voice conversion, is the song melody. Established algorithms, such as DIO in the WORLD vocoder [5], as well as deep-learning-based algorithms such as CREPE [6] and RMVPE [7], are effective at extracting the fundamental frequency f_0 contour from singing audio. These content-related acoustic features can be generated frame-wise by employing a sliding window method, similar to the short-

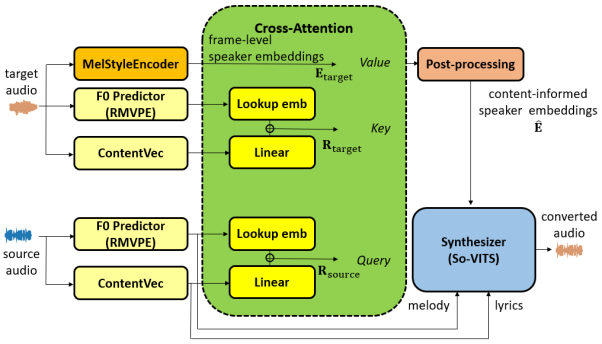


Fig. 1: Our proposed singing voice conversion framework using frame-wise speaker embeddings informed by ContentVec and f_0.

time Fourier transform, ensuring alignment across different features.

As for the timbre-related acoustic features, early work [8] represented a singer using a one-hot-like identity vector, but this approach has limitations on feature reusability and generalizability. First, one-hot-type vectors simply prompt the system to toggle between a fixed set of voices seen in training (any-to-many voice conversion). Second, such ID representations do not capture timbre information in a latent space where singers with similar voices tend to cluster with small distances. Such a latent space would allow for creativity with interpolation for voice morphing [9], or extrapolation for an unseen target voice (any-to-any voice conversion) [10].

Recent research has addressed this by replacing one-hot-type vectors with meaningful speaker embeddings such as ECAPA-TDNN speaker embeddings [11] or x-vectors [12] which are pre-trained for speaker recognition tasks in ASR. In our work, the term "speaker embedding" is borrowed from speech literature to refer to singer timbre representation. With the x-vector system, only one static vector is used to represent the timbre of an entire audio clip regardless of length, but this fails to capture timbre variability over time and results in a lack of adaptability to different enunciations and frequency ranges when singing.

In this paper, we propose a method to generate speaker embeddings from frame-level timbre information, in congruence with the frame-level approach for content features. Our approach shown in Figure 1 applies cross-attention to the lyrics and melody of the utterance and predicts the corresponding speaker embedding for each frame, thus enabling better adaptability to timbre variations over time. Additionally, we train our model on a more timbre-diverse dataset with 100 Mandarin singers, enabling better timbre generalization across a wide range of voices. Consequently, our model surpasses previous approaches in terms of voice similarity, vocal expressiveness, and enunciation clarity.

Inspired by the self-supervised learning paradigm, our training strategy resembles masked autoencoder frameworks such as HuBERT [3], in which parts of the input are intentionally masked and reconstructed to encourage the model to learn robust internal representations. Specifically, we mask the current and surrounding frames during training and learn to reconstruct the current frame's timbre embedding from other non-masked, content-related frames. This enables the model to discover meaningful frame-wise timbre information without relying on external labels.

2 Related work

2.1 So-VITS-SVC

So-VITS[1] is a conditional variational autoencoder (VAE) with a flow-based module that achieves state-of-the-art naturalness and timbral similarity in synthesizing singing voices of target singers [13]. Its core architecture includes one decoder and two encoders: a prior encoder that maps acoustic features into a latent space, and a posterior encoder that encodes mel-spectrograms of target utterances. A Kullback-Leibler (KL) divergence is applied to the two latent distributions to align them during training. The decoder reconstructs the singing voice from the latent representation, effectively serving as the generator within the VAE architecture.

To further enhance generation quality, So-VITS incorporates a generative adversarial network (GAN) component during training. Specifically, a multi-period discriminator is adopted, similar to that used in HiFi-GAN [14], to distinguish real from synthesized waveforms. This introduces an adversarial loss that guides the decoder to produce more natural-sounding audio.

Beyond So-VITS' architectural strengths, its success is also attributed to the use of ContentVec as a robust content feature extractor.

[1] https://github.com/svc-develop-team/so-vits-svc.git

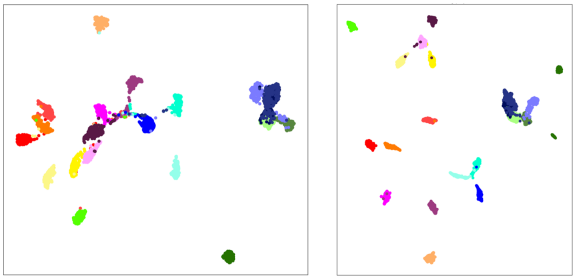
2.2 ContentVec

ContentVec [4], adapted from HuBERT, excels at preserving content information while removing speaker-related details. It is comprised of three components: a predictor and two speech representation networks, referred to as the *Student* and *Teacher*. *Teacher* uses a pre-trained, unsupervised model that generates speech representations, which are then clustered using K-means. These clusters serve as self-supervised learning labels for *Student*. *Student*, another speech representation network, is trained to focus on learning content-related information. The predictor classifies the categories of the speech representations generated by the *Student*. Both the *Student* and the predictor are trained to minimize cross-entropy loss between the clustered labels and the predicted labels. ContentVec outperforms other models with similar functionality by applying timbre transformations to both *Student* and *Teacher* networks, which allows *Student* and predictor to focus more effectively on content, thereby preserving more content information while disentangling speaker information.

2.3 Mel-Style Encoder

We choose to use the mel-style encoder proposed by Min et al [15] in place of the x-vector to generate speaker embeddings for timbre conditioning in So-VITS. This decision is based on our observation that the mel-style encoder demonstrates better voice distinction on the M4Singer [16] dataset, which consists of 20 singers across four voice types: soprano, alto, tenor, and bass. It produces tighter clustering compared to the x-vector in 2D visualizations, where the n-dimensional speaker embeddings are reduced using UMAP [17], as shown in Figure 2. In the visualization, the 20 singers are clustered in different colors, clearly reflecting inter-singer separability. Details of these embeddings are described as follows.

The mel-style encoder takes a mel-spectrogram as input and generates a vector that captures the speaker's timbre characteristics for a given utterance. This encoder consists of four sequential modules: spectral processing, temporal processing, multi-head self-attention, and a time-averaging module. The spectral and temporal processing modules transform spectral information from the frequency domain into a latent space, while capturing temporal patterns in the time domain. Following this, multi-head self-attention is applied to highlight global patterns across the entire utterance. Finally, the time-averaging module produces a speaker embedding of the entire utterance to represent the timbre information of the speaker. We adopt the first three modules of this encoder: spectral processing, temporal processing, and multi-head self-attention, while discarding the fourth time-averaging module, in order to generate frame-level speaker embeddings.



(a) X-vector **(b)** Mel-Style Embeds

Fig. 2: 2D Visualization of Speaker Embeddings.

2.4 X-Vector

The x-vector [12] is a powerful module pre-trained on thousands of speakers from the VoxCeleb [18] dataset. X-vectors have been well-established in forensics literature [19] as effective speaker representations with good accuracy in distinguishing different voices. The x-vector encoder takes audio as input and outputs a 512-dimensional speaker embedding. The encoder module was pre-trained on VoxCeleb for the purpose of automatic speaker recognition, and the output of the second-to-last hidden layer is used as a timbre feature vector, since the network has learned to distill information relevant to distinguishing one voice from another.

3 Methods

The overall architecture of our model for generating frame-wise speaker embeddings is shown in Figure 3. The model consists of a mel-style encoder, a multi-head cross-attention module informed by ContentVec and the f_0 contour, and a post-processing module. In this section, we provide a detailed explanation of each component.

3.1 Frame-wise Acoustic Feature Preparation

Consider a singing utterance $\boldsymbol{x} \in \mathbb{R}^{1\times L_a}$, where L_a represents the samples in the audio waveform. We processed

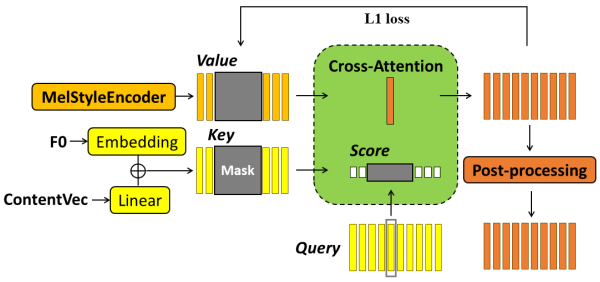


Fig. 3: Multi-head cross-attention on ContentVec and f_0.

audio clips of duration 5 to 10 seconds with a sampling rate of 44.1kHz, Hann window, frame length of 2048 samples, and hop size of 512.

Three crucial frame-wise acoustic features are generated from the utterance by the following modules: 1) the timbre-related Mel-Style Encoder $e(\cdot) : \mathbb{R}^{1\times L_a} \to \mathbb{R}^{C_e\times L_T}$, 2) the fundamental frequency predictor $f(\cdot) : \mathbb{R}^{1\times L_a} \to \mathbb{R}^{2\times L_T}$, and 3) the lyrics representation content-related encoder $c(\cdot) : \mathbb{R}^{1\times L_a} \to \mathbb{R}^{C_c\times L_T}$, where L_T represents the number of frames, and C_e=512, C_c=768 represent the embedding dimensions, respectively.

In order to generate frame-wise speaker embeddings for timbre information, we combine the mel-spectrogram and our revised mel-style encoder (where we kept the first three modules but removed the time-averaging module) as our timbre-related encoder. The output is denoted as:

$$\mathbf{E} = e(\boldsymbol{x}) \in \mathbb{R}^{C_e\times L_T} \tag{1}$$

Additionally, we utilize two pre-trained models: ContentVec and RMVPE, to extract content-related features. The ContentVec module leverages self-supervised learning techniques and demonstrates robust performance in downstream tasks such as ASR and voice conversion [4], and is used here to represent lyrics information:

$$\mathbf{C} = c(\boldsymbol{x}) \in \mathbb{R}^{C_c\times L_T} \tag{2}$$

We use the RMVPE module to represent melody, as it excels at generating reliable f_0 contours, outperforming traditional algorithms and earlier deep-learning-based models [7]. The output is denoted as:

$$\mathbf{F} = f(\boldsymbol{x}) = (\boldsymbol{f_0}, \boldsymbol{v}) \tag{3}$$

where $\boldsymbol{f_0} \in \mathbb{R}^{L_T}$ is the fundamental frequency contour and $\boldsymbol{v} \in \{0,1\}^{L_T}$ contains binary indicators of voiced or unvoiced in each frame.

3.2 Multi-head Cross-Attention

The multi-head cross-attention [20] module shown in Figure 3, aims to generate frame-wise speaker embeddings for timbre representations, informed by ContentVec (lyrics and enunciation information) and the f_0 contour (melody information).

The general equation for cross-attention is given by

$$\text{CrossAttention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \mathbf{V}\cdot\text{softmax}\left(\frac{\mathbf{K}^\top\mathbf{Q}}{\sqrt{d_k}}\right) \tag{4}$$

where queries are column vectors in $\mathbf{Q} \in \mathbb{R}^{d_k\times n_q}$, and similarity scores are calculated with keys that are column vectors in $\mathbf{K} \in \mathbb{R}^{d_k\times n_k}$. Values are column vectors in $\mathbf{V} \in \mathbb{R}^{d_v\times n_k}$, resulting in a $\mathbb{R}^{d_v\times n_q}$ matrix, where each column is a weighted sum of value column vectors corresponding to its query vector .

Figure 3 shows sets of frame-level vectors. In this process, the *query* $\mathbf{Q}$ and *key* $\mathbf{K}$ in the multi-head cross-attention module are acoustic features related to content, obtained by projecting ContentVec $\mathbf{C}$ and the pitch information $\mathbf{F}$ into a C_R=192-dimensional latent space and adding:

$$\mathbf{R} = \mathscr{B}_1(\boldsymbol{f_0}) + \mathscr{B}_2(\boldsymbol{v}) + \text{proj}(\mathbf{C}) \in \mathbb{R}^{C_R\times L_T} \tag{5}$$

where $\mathscr{B}_1$ and $\mathscr{B}_2$ are lookup tables that transform the 256 discrete frequency bins in $\boldsymbol{f_0}$ and the 2 categories of voiced/unvoiced in $\boldsymbol{v}$ into C_R-dimensional vectors, respectively.

We predict the ith frame's speaker embedding as follows:

$$\hat{\mathbf{E}}_i = \text{CrossAttention}(\mathbf{R}_i, \mathbf{R}_{\mathscr{T}\setminus\mathscr{M}_i}, \mathbf{E}_{\mathscr{T}\setminus\mathscr{M}_i}) \in \mathbb{R}^{C_e\times 1} \tag{6}$$

where $\mathbf{R}_i \in \mathbb{R}^{C_R\times 1}$ is the ith column of $\mathbf{R}$, representing the lyrics and melody content of the current ith frame. The set of all frame indices is denoted as $\mathscr{T} = \{1,\ldots,L_T\}$, and $\mathscr{M}_i = \{i-k,\ldots,i+k\}$ defines a symmetric mask of approximately 0.5 seconds centered on the ith frame. The matrices $\mathbf{R}_{\mathscr{T}\setminus\mathscr{M}_i} \in \mathbb{R}^{C_R\times N}$ and $\mathbf{E}_{\mathscr{T}\setminus\mathscr{M}_i} \in \mathbb{R}^{C_e\times N}$ denote the masked-out content and speaker embedding sequences, respectively, where $N = L_T - (2k+1)$ in the general case. At the boundaries, the

mask is clipped to avoid invalid indices. Specifically, when $i = 1$, the mask becomes $\mathscr{M}_1 = \{1, \ldots, 1+k\}$; when $i = L_T$, we set $\mathscr{M}_{L_T} = \{L_T - k, \ldots, L_T\}$. In such cases, N is adjusted accordingly, for instance, to $N = L_T - (k+1)$ when only one side of the mask is clipped.

The frame-wise *key-value* pairs consist of content-related acoustic features and speaker embeddings, respectively. Our concept is to utilize the current ith frame's acoustic content information $\mathbf{R}_i$ to query and find other frames that have similar content keys in $\mathbf{R}_{\mathscr{T} \setminus \mathscr{M}_i}$, in order to combine those frames' corresponding speaker embedding values in $\mathbf{E}_{\mathscr{T} \setminus \mathscr{M}_i}$, as a representation of the current frame's timbre.

During training, we encourage the cross-attention module to reconstruct the current frame's speaker embedding from the speaker embeddings of other frames with relevant content keys. To achieve this, both the *query* and the *key* are set to the same clip's content-related acoustic features. Close-proximity frames have near-identical content features, and we want our system to learn how to query for similar but *different* content. Therefore, we apply a mask $\mathscr{M}_i$ that spans approximately 0.5 seconds centered around the ith frame to ignore the current and neighboring frames' *key-value* pairs during training. For each frame's *query*, we obtain a relevance score with all the non-masked $\mathscr{T} \setminus \mathscr{M}_i$ frames' *keys*. High relevance means the current frame and another frame have similar content (word, melody) acoustic features, so intuitively these frames should have relevant speaker embeddings. These scores are then used as weights to predict the current frame's speaker embedding $\hat{\mathbf{E}}_i$ using a linear combination of the non-masked frames' speaker embeddings.

During training, we can apply the L_1 loss of $L_1(\hat{\mathbf{E}}_i, \mathbf{E}_i)$ between the predicted $\hat{\mathbf{E}}_i$ and original $\mathbf{E}_i$ frame-wise speaker embeddings, with the original serving as the ground truth.

During inference, no mask is needed since *query* and *key* originate from different audio, as shown in Figure 1. Here, the *query* is the content of the source audio, and the *key-value* pairs are the content and speaker embeddings from the target audio.

$$\hat{\mathbf{E}}_i = \text{CrossAttention}(\mathbf{R}_{\text{source},i}, \mathbf{R}_{\text{target}}, \mathbf{E}_{\text{target}}) \quad (7)$$

Thus, when the source content has changes in singing enunciations and pitch range, we can find relevant target timbre embeddings that reflect those enunciations and ranges.

3.3 Post-processing on Frame-wise Speaker Embeddings

Through the cross-attention module, we obtain frame-wise speaker embeddings informed by ContentVec and the f_0 contour, which can be directly used to condition the variational autoencoder in the So-VITS SVC model. However, before these embeddings are input into SoVits, we introduce a post-processing module. Since each frame is predicted independently by the cross-attention mechanism, the resulting frame-wise speaker embeddings are unaware of the embeddings of other frames, even those of neighboring ones, which may lead to perceptible discontinuities.

To address this, we apply 1D convolution to refine the frame-wise speaker embeddings generated by the cross-attention module. Each row in $\hat{\mathbf{E}} \in \mathbb{R}^{C_e \times L_T}$ is treated as a time series of length L_T, and a sliding kernel is applied so that neighboring values can be used to get a smoother result. In particular, our post-processing module contains one layer of 1D convolution, each layer with C_e learnable sliding kernels of size 5, shift 1, and zero-padding. We denote our post-processing smoothing operation by $s(\cdot) : \mathbb{R}^{C_e \times L_T} \to \mathbb{R}^{C_e \times L_T}$.

Thus, we modify the loss function used in training to $L_1(s(\hat{\mathbf{E}}), \mathbf{E})$.

4 Experiments

In this section, we first describe the composition of the dataset and conduct an ablation study to compare our proposed model with the original So-VITS. Section 5 discusses our subjective and objective assessments.

Table 1: Details of the Datasets

Dataset	Singers (F/M)	Duration	Sampling rate
Mpop600	4 (2/2)	12.3 hrs	48 kHz
M4Singer	20 (10/10)	29.7 hrs	48 kHz, 96 kHz
OpenSinger	76 (48/28)	51.9 hrs	44.1 kHz
Combined	100 (60/40)	14.8 hrs	44.1 kHz

Audio used for training and inference was taken from three Mandarin singing corpora: our lab's internal MPop600 [21] dataset, M4Singer [16], and OpenSinger [22]. The combined dataset includes 100 singers, with 60 females and 40 males. Details about

the singers, sampling rate, and duration are provided in Table 1. We segmented all audio into 5- to 10-second clips.

The original So-VITS model relies on a one-hot-like speaker identity representation, which limits the model to learning only a few singers at a time and requires tens of minutes to a few hours of training audio per singer (any-to-many conversion). In our experiment, we replace their approach of few singers with many samples each, to our approach of many singers with few samples each: we use comparatively many singers (60 females, 40 males) with smaller training audio duration per singer (9 minutes each). This allows us to investigate how well the model can generalize across a diverse range of singer timbres, despite the limited duration of data per singer (toward any-to-any conversion).

Using the combined dataset, we trained the singing voice conversion model with three different types of speaker embeddings, as shown in the tables in section 5.

1. **So-VITS** The baseline model is the original So-VITS conditioned on x-vectors.
2. **MelStyle-SVC** The second model is So-VITS conditioned on clip-wise mel-style embeddings, which are obtained by taking the temporal average of frame-wise mel-style embeddings from the target timbre audio.
3. **CrossAtt-SVC** Our proposed model, however, is conditioned on frame-wise mel-style embeddings, which are informed by content-related acoustic features, using multi-head cross-attention and post-processed by a 1D convolution.

We trained each system in an end-to-end manner. All models were trained with the AdamW optimizer on two RTX-3090 GPUs, using a mini-batch size of 20 for 120,840 steps. The learning rate was set to 10^{-4}.

5 Results and Discussion

In this section, we evaluate our results objectively and subjectively. Our objective metrics include Mel-Cepstral Distortion (MCD), Modulation Spectra Distance (MSD) [23], and cosine similarity of speaker embeddings (SpkCS) via x-vector and d-vector [2] [24, 9].

Because singing is highly subjective, we also rely on human perception in subjective listening tests and conduct mean opinion score (MOS) assessments on a 1-to-5 scale (5 being best) for the following three criteria:

- **Clarity of Articulation** refers to how clear the pronunciation of each word is in the audio file.
- **Timbre Similarity** measures how closely the timbre of the audio matches that of the real singer.
- **Vocal Expressiveness** assesses whether the singing demonstrates strength and control over the voice for more expressiveness throughout the performance.

We assess the systems in two parts. Part 1 focuses on singing voice reconstruction, where a sung utterance is used as both the source and target audio, and its own speaker embedding is employed to reconstruct the singing voice with the same content. Part 2 involves singing voice conversion, where the source singing audio is transformed into another timbre using a different target's speaker embedding. A demo page[3] is available.

Table 2: Part 1 Reconstructions Objective Metrics

	MCD ↓	MSD ↓	SpkCS ↑
So-VITS	7.802	0.4945	0.9785/0.9233
MelStyle-SVC	**7.614**	0.4863	0.9885/0.9413
CrossAtt-SVC	7.755	**0.4412**	**0.9909**/**0.9482**

We begin by evaluating the upper-bound performance of each model in the part 1 reconstruction task. Objective metrics are shown in Table 2. We can observe improvements across all three objective metrics after replacing the x-vector speaker embedding with the mel-style embedding. Our proposed model shows progress in cosine similarity of speaker embeddings and particularly in Modulation Spectra Distance, with only a slight degradation in MCD. This suggests that the cross-attention mechanism enhances speaker similarity while allowing the reconstruction of more detailed varying spectro-temporal components.

In our subjective assessment, we collected data from 23 listeners via an online questionnaire. In the Part 1

[2] https://github.com/resemble-ai/Resemblyzer/tree/master

[3] https://m105.nthu.edu.tw/~s105061807/wilsun/xattn.html

Table 3: Part 1 Reconstructions
Subjective MOS Scores

	Clarity	Similarity	Expressiveness
Ground Truth	4.48 ± 0.28	4.03 ± 0.19	4.14 ± 0.18
So-VITS	3.77 ± 0.37	3.74 ± 0.31	3.77 ± 0.32
MelStyle-SVC	4.01 ± 0.26	3.73 ± 0.25	3.73 ± 0.27
CrossAtt-SVC	$\mathbf{4.17 \pm 0.27}$	$\mathbf{3.80 \pm 0.16}$	$\mathbf{3.81 \pm 0.30}$

reconstruction task, 10 audio examples were each reconstructed by our 3 systems, along with the ground truth audio sung by real singers. In the Part 2 voice conversion task, 10 audio examples were also converted into another timbre by our 3 systems. These 70 audio examples were MOS-evaluated by listeners for clarity, similarity, and expressiveness. The audio examples used in evaluation had been held out from the training set. During the blind listening tests, the real singers' audio files and the synthesized outputs from three models were randomly ordered.

Table 3 further demonstrates that our model achieves the best performance across all three subjective MOS metrics, especially in clarity of articulation.

Next, we apply objective and subjective metrics to our part 2 singing voice conversion task. The results for these evaluations are presented in Table 4. For objective metrics, we use the x- and d-vector cosine similarities, since MCD and MSD would require unavailable ground truths. Subjective MOS scores for clarity, similarity, and expressiveness are also shown. We see that our proposed model still outperforms in cosine similarity of speaker embeddings, clarity and similarity with a slight drop in expressiveness.

Table 4: Part 2 Voice Conversions
Objective SpkCS and Subjective MOS

	SpkCS	Clarity	Similarity	Expressiveness
So-VITS	0.9615/0.8491	3.89 ± 0.24	3.87 ± 0.21	$\mathbf{3.93 \pm 0.22}$
MelStyle-SVC	0.9701/0.8691	4.11 ± 0.20	4.03 ± 0.23	3.84 ± 0.27
CrossAtt-SVC	**0.9738/0.8693**	$\mathbf{4.14 \pm 0.33}$	$\mathbf{4.04 \pm 0.13}$	3.88 ± 0.27

We also obtained informal feedback from in-house listeners in non-blind tests, and many reported that they prefer our proposed CrossAtt-SVC model's timbre compared to the original So-VITS conversions, because the originals have a muffled quality with singing that is not well-enunciated and sounds a bit like humming. This implies that our system has potential for good expressiveness, though the slight drop in MOS in Table 4 may be due to artifacts such as pitch fluctuations and background noise. Feedback indicates that our proposed model excels at reconstructing the high-frequency components of the singing voice, leading to improved clarity of articulation and enhanced brightness and fullness of the singer's timbre. This also contributes to a higher similarity score compared to baseline So-VITS. However, the overall quality of our model's conversion is not as robust as the original So-VITS, with occasional artifacts due to discontinuities introduced by our high-resolution frame-by-frame speaker embeddings.

In summary, using mel-style embeddings and the further step of cross-attention result in improvements in similarity and clarity. Similarity is improved because the frame-wise speaker embeddings are informed by the melody and capture the singer's varying vocals; clarity is improved because the frame-wise embeddings are informed by the content (lyrics and melody) and thus improve pronunciation and distinguishing spectro-temporal details of singing techniques.

6 Conclusion

Improvements in content representations have fueled rapid advances in singing voice conversion systems, but more work needs to be done in finding accurate timbre representations. This paper presents a cross-attention mechanism to generate frame-wise speaker embeddings informed by linguistic and melody information to better reflect how one singer's timbre can vary over time with different singing techniques and vocal ranges. Subjective and objective evaluations show encouraging results, with our model improving on clarity and voice similarity.

Acknowledgment

This research was supported by the National Science and Technology Council of Taiwan under Grant No. 109-2221-E-007-094-MY3 and 111-2634-F-002-023.

References

[1] Z. Li, B. Tang, X. Yin, Y. Wan, L. Xu, C. Shen, and Z. Ma, "PPG-based singing voice conversion with adversarial representation learning,"

Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 7073–7077, 2021.

[2] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 12449–12460, 2020.

[3] W.-N. Hsu, B. Bolte, Y.-H. H. Tsai, K. Lakhotia, R. Salakhutdinov, and A. Mohamed, "HuBERT: Self-supervised speech representation learning by masked prediction of hidden units," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 3451–3460, 2021.

[4] K. Qian, Y. Zhang, H. Gao, J. Ni, C.-I. Lai, D. Cox, M. Hasegawa-Johnson, and S. Chang, "ContentVec: An improved self-supervised speech representation by disentangling speakers," *Proceedings of the International Conference on Machine Learning (ICML)*, vol. 162, pp. 18003–18017, 2022.

[5] M. Morise, F. Yokomori, and K. Ozawa, "WORLD: A vocoder-based high-quality speech synthesis system for real-time applications," *IEICE Transactions on Information and Systems*, vol. E99-D, no. 7, pp. 1877–1884, 2016.

[6] J. W. Kim, J. Salamon, P. Li, and J. P. Bello, "CREPE: A convolutional representation for pitch estimation," *Proc. ICASSP*, pp. 161–165, 2018.

[7] H. Wei, X. Cao, T. Dan, and Y. Chen, "RMVPE: A robust model for vocal pitch estimation in polyphonic music," *Proceedings of Interspeech*, pp. 5421–5425, 2023.

[8] Y. Cho, Y. Tsao, H. Wang, and Y. Liu, "Mandarin singing voice synthesis with denoising diffusion probabilistic wasserstein gan," *Proceedings of the Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pp. 1956–1963, 2022.

[9] E. M. Liu, J.-W. Yeh, J.-H. Lu, and Y.-W. Liu, "Speaker embedding space cosine similarity comparisons of singing voice conversion models and voice morphing," *The Journal of the Acoustical Society of America*, vol. 154, p. A244, 2023.

[10] E. Casanova, J. Weber, C. D. Shulby, A. C. Junior, E. Gölge, and M. A. Ponti, "YourTTS: Towards zero-shot multi-speaker TTS and zero-shot voice conversion for everyone," *Proc. ICML*, vol. 162, pp. 2709–2720, 2022.

[11] B. Desplanques, J. Thienpondt, and K. Demuynck, "ECAPA-TDNN: Emphasized Channel Attention, propagation and aggregation in TDNN based speaker verification," *Proc. Interspeech*, pp. 3830–3834, 2020.

[12] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-Vectors: Robust DNN embeddings for speaker recognition," *Proc. ICASSP*, pp. 5329–5333, 2018.

[13] W.-C. Huang, L. P. Violeta, S. Liu, J. Shi, and T. Toda, "The singing voice conversion challenge 2023," *Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 1–8, 2023.

[14] J. Kong, J. Kim, and J. Bae, "HiFi-GAN: Generative adversarial networks for efficient and high fidelity speech synthesis," *Proc. NeurIPS*, vol. 33, pp. 17022–17033, 2020.

[15] D. Min, D. B. Lee, E. Yang, and S. J. Hwang, "Meta-StyleSpeech : Multi-speaker adaptive text-to-speech generation," *Proc. ICML*, vol. 139, pp. 7748–7759, 2021.

[16] L. Zhang, R. Li, S. Wang, L. Deng, J. Liu, Y. Ren, J. He, R. Huang, J. Zhu, X. Chen, and Z. Zhao, "M4Singer: A multi-style, multi-singer and musical score provided mandarin singing corpus," *Proc. NeurIPS, Datasets and Benchmarks Track*, 2022.

[17] L. McInnes, J. Healy, N. Saul, and L. Großberger, "UMAP: Uniform manifold approximation and projection," *Journal of Open Source Software*, vol. 3, no. 29, p. 861, 2018.

[18] A. Nagrani, J. S. Chung, W. Xie, and A. Zisserman, "Voxceleb: Large-scale speaker verification in the wild," *Computer Speech Language*, vol. 60, p. 101027, 2020.

[19] D. van der Vloed, F. Kelly, and A. Alexander, "Exploring the effects of device variability on forensic speaker comparison using vocalise and nfi-

frida, a forensically realistic database," *Proceedings of The Speaker and Language Recognition Workshop*, pp. 402–407, 2020.

[20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Proc. NeurIPS*, vol. 30, pp. 5998–6008, 2017.

[21] C.-C. Chu, F.-R. Yang, Y.-J. Lee, Y.-W. Liu, and S.-H. Wu, "MPop600: A mandarin popular song database with aligned audio, lyrics, and musical scores for singing voice synthesis," *Proc. APSIPA ASC*, pp. 1647–1652, 2020.

[22] R. Huang, F. Chen, Y. Ren, J. Liu, C. Cui, and Z. Zhao, "Multi-Singer: Fast Multi-Singer singing voice vocoder with a large-scale corpus," *Proceedings of the ACM International Conference on Multimedia*, pp. 3945–3954, 2021.

[23] M. T. Akhter, P. Banerjee, S. Dhar, and N. D. Jana, "An analysis of performance evaluation metrics for voice conversion models," *Proceedings of the IEEE India Council International Conference*, pp. 1–6, 2022.

[24] L. Wan, Q. Wang, A. Papir, and I. L. Moreno, "Generalized end-to-end loss for speaker verification," *Proc. ICASSP*, pp. 4879–4883, 2018.

Audio Engineering Society

Conference Paper 24

Presented at the AES International Conference on Machine Learning and Artificial Intelligence for Audio
2025 September 8-10, London, UK

This paper was peer-reviewed as a complete manuscript for presentation at this conference. This paper is available in the AES E-Library (http://www.aes.org/e-lib), all rights reserved. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Journal of the Audio Engineering Society.

Procedural Music Generation Systems in Games

Shangxuan Luo[1], Joshua Reiss[1]

[1]Queen Mary University of London

Correspondence should be addressed to Shangxuan Luo (s.luo@qmul.ac.uk)

ABSTRACT

Procedural Music Generation (PMG) is an emerging field that algorithmically creates music content for video games. By leveraging techniques from simple rule-based approaches to advanced machine learning algorithms, PMG has the potential to significantly improve development efficiency, provide richer musical experiences, and enhance player immersion. However, academic prototypes often diverge from applications due to differences in priorities such as novelty, reliability, and allocated resources. This paper bridges the gap between research and applications by presenting a systematic overview of current PMG techniques in both fields, offering a two-aspect taxonomy. Through a comparative analysis, this study identifies key research challenges in algorithm implementation, music quality and game integration. Finally, the paper outlines future research directions, emphasising task-oriented and context-aware design, more comprehensive quality evaluation methods, and improved research tool integration to provide actionable insights for developers, composers, and researchers seeking to advance PMG in game contexts.

1 Introduction

Procedural Music Generation (PMG) originates from the concept of Procedural Content Generation (PCG), which is the algorithmic creation of game content with limited or indirect user input [1]. The basic idea of PCG seems to first appear concurrently with the early development of computers around the 1980s [2]. One early example of using PCG was Rogue in 1978, which became the beginning of Rogue-like games. In Rogue, algorithms were used to generate randomised dungeons, with the map changing each time you played. Without storing massive static assets, PCG addressed hardware memory bottlenecks in the early days, but has since become more widely used to expand the game content and richness, including levels, characters, stories and sound.

In addition to significantly impacting the player's immersion by generating diverse content, PCG also dramatically reduces the development time and cost [1], [3], [4], particularly benefits the non-linearity characteristic of games that require dynamic, replayable, or expansive content, such as Rogue-likes, Sandboxes, Open-World Games, etc.

Music, as an important aspect of game content, provides unique functional support to enhance the game experience. For example, theme music aims to express the core tone or character identity of a work, whereas background music seeks to render the atmosphere of a scene. Moreover, music offers a unique means of enhancing storytelling through its ability to adapt dynamic environmental changes in games and facilitate smoother scene or state transitions.

However, most games comprise a soundtrack of between one and four hours of music, whereas the duration of game-play can vary considerably, from dozens to hundreds of hours [5]. Furthermore, Live Service Games are often designed to provide endless player experience by iterating the gameplay [6]. Yet very little music goes with it. This results in players hearing music repeated on many occasions. While repetition is a critical element of music, excess can disrupt immersion and become a source of frustration and fatigue for the player [7], [8]. Therefore, extending the concepts and methodologies of PCG to music is a highly effective strategy that alleviates this fatigue by generating a wide variety of unique content, thus improving the overall game quality.

This paper aims to examine the evolution of PMG systems in both research and application areas. It intends to compare current methodologies, highlight emerging trends, identify gaps and challenges, and provide constructive recommendations for future advancements in the field.

2 Background

2.1 Definition

An early definition of procedural music given by Collins is a composition that evolves in real-time according to a specific set of rules or control logics [9]. To further elucidate the evolution of the system in response to input data, this paper adopts a definition closer to procedural audio generation [10]: PMG is *the use of algorithms to dynamically generate music content in real time while adapting to changing inputs.*

Compared to similar concepts like algorithmic music or generative music, which also involve creating music through systemic automation, the critical characteristic of PMG is its real-time adaptability to dynamic game content. This requires systems capable of making instant adjustments and variations to the music in response to changes in the game state. As a result, PMG often relies on complex rule-based frameworks in larger games to determine how to interpret and respond to game data, ensuring the music remains cohesive and contextually appropriate throughout the player's experience.

A typical workflow of PMG systems is illustrated in Figure 1. The system receives two primary types of

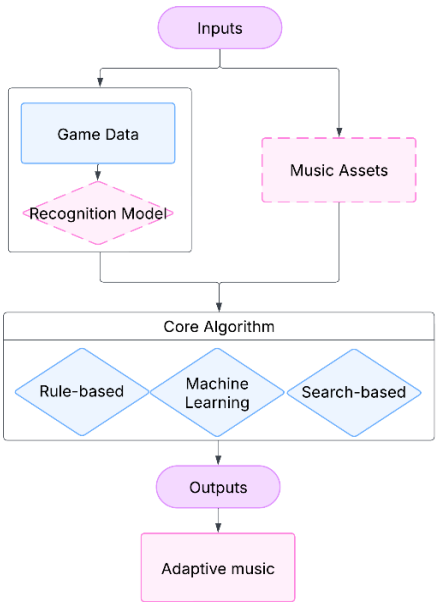


Figure 1. A Typical Workflow of PMG Systems.

input: in-game state data and music asset data. Based on these inputs, the system generates real-time adaptive music as its output, dynamically adjusting to reflect the evolving context and gameplay.

The state data usually refers to all the information used in a video game to describe the current condition of play. The following parameters are commonly used to define and control various elements of procedural music:

- *Character* parameters include vitality status (healthy or dangerous), behavioural patterns (e.g. sneaking or attacking), etc.
- *Environment* parameters include weather, time of day, terrain, etc.
- *Spatial* parameters include enemy distance, location coordinates, etc.
- *Game mechanics* parameters include quest progression, difficulty level, etc.
- *Interaction* parameters include player input (mouse/keyboard mapping) and the results of interactions with other events or objects.

In academic research, many systems introduce an intermediate model, which we tentatively call cognitive models, to interpret and analyse the above parameters to generate a reference that guides the subsequent music generation [11], [12], [13], [14]. The main references involve emotion labels and intensity levels. Essentially, this approach maps high-

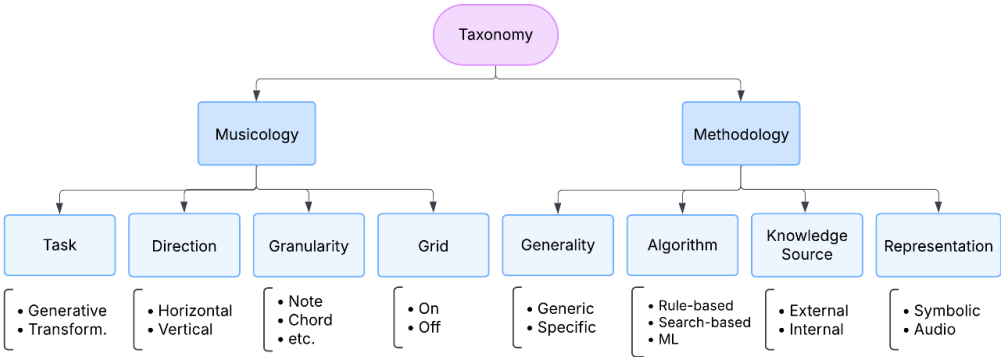


Figure 2. Hierarchy Taxonomy of Procedural Music Systems.

dimensional data into a low-dimensional space to reduce system complexity for prototyping.

Another type of input data, music assets, are audio files or resources that are used in the game development process, mainly including sound effects and soundtrack clips. There are two types of musical compositions in games that indicate whether music assets are involved in the generation progress. Generative composition refers to music created entirely by algorithms without using pre-existing music assets as inputs. In contrast, transformational composition involves rearranging or modifying existing fragments from music assets to generate new music [15].

Based on the input of game data and music assets, the system dynamically generates music through its core algorithms. These algorithms are classified into three main categories: rule-based, search-based and machine learning. However, a mixed approach is commonly used in practice. We will further discuss them in the subsequent sections.

2.2 Taxonomy

To systematically categorise and analyse PMG systems, we adopt a comprehensive taxonomy that inherits and adapts the work of [16], shown in Figure 2. For a detailed comparison of the systems studied, see Table 1 in Appendix. We investigated eight dimensions of these systems in two main aspects: Musicology and Methodology. Musicology contains four categories, including Task, Direction, Granularity and Grid. Methodology also contains four categories: Generality, Algorithm, Knowledge Source and Representation.

Tasks indicate whether the system is generative or transformational [15]. Transformational systems generate new music by reorganising existing music assets, whereas generative systems usually do not require music assets as input and can generate music from scratch using rules and algorithms. The boundary between the two is not entirely deterministic, and is mainly measured by the granularity of the music manipulated by the system.

Direction refers to whether the system can manipulate the music horizontally or vertically. Horizontal manipulation involves generating or rearranging music over time, such as adjusting the order of measures or phrases. In contrast, vertical manipulation involves changing the music in terms of layer at a single time point, such as instrument groups.

Granularity refers to the level of detail with which a music system generates musical elements or aspects. A generative system has higher levels of detail in terms of note compared to a transformational system, which often manipulates the music at a coarser level, such as measures or phrases. Other aspects of granularity include chords, timbre, tempo, velocity, etc.

Grid refers to whether the music generated by the system is aligned to the beat. In other words, whether the music has a relatively fixed groove. In Western classical music, the rhythm is mainly triple and quadruple meter, and the music's tempo, dynamics, and chords are usually based on the meter. Non-aligned music is common in some Eastern cultures, but in the context of game music, it is generally for ambient establishment.

Generality refers to the purpose for which a system is designed. Specific systems are designed for a particular game or style of play, whereas generic systems are designed to provide a more general solution for a wider range of game scenarios or categories.

Algorithms refer to the methodology used by the system and are categorised into rule-based, search-based and machine learning. Rule-based methods refer to techniques according to predefined rules or patterns. Some stochastic methods, such as Markov models, are often integrated into rule-based methods. Search-based methods explore a vast space of possible solutions to find the best or most desirable outcome based on specific criteria or fitness functions. Machine learning methods simulate neural networks and are trained on large-scale data to learn music patterns.

Knowledge Source (KS) is where information of the algorithm comes from. External knowledge comes from composers or system designers, especially in rule-based systems. They predefine the rules based on their experience and understanding of the music, which is often complex and done manually. Internal knowledge comes from the data itself, for example, the hidden state of a stochastic model or the weight of a trained deep-learning model.

Representation refers to the method that a system uses to store and organise its musical knowledge. In computing systems, the most widely used format for symbolic representation is MIDI (Musical Instrument Digital Interface), where each musical event is encoded using variables such as pitch, velocity, channel, and on/off states. Audio representation involves capturing analog signals of real-world music and converting them into a digital format for storage and processing.

3 Research

In academia, a variety of advanced methods are used. We broadly classify them into three categories, rule-based, search-based, and machine learning-based, according to the research in PCG [1].

3.1 Rule-based methods

Rule-based methods, sometimes referred to as constrictive methods, usually act as constraints by encoding rules from music theory. One example is the use of a feasibility equation to limit the content generated [17]. This equation encodes the following rules:

$$Feasibility = -\sum_{n=0}^{N-1} L(n, n+1) + S(n, n+1) + D(n, n+1)) \quad (1)$$

Where L encodes that it should not have leaps between notes bigger than a fifth, S encodes that it should contain at least a minimum number of leaps of a second and *D* encodes that each note pitch should differ from the preceding note pitch. All of the sub-equations in the equation are Boolean, and thus the whole function returns a range of values from -3 to 0. A 0 means that the music generated is reasonable, and a negative value does not.

Lopez uses Max/MSP, a visual programming environment, to implement a rule-based system that generates real-time game music and provides interesting generation results [18]. The rhythmic development algorithm is based on Lerdahl & Jackendoff's Generative theory [19]. Pitch Generation utilises 'drunk ' and "drunk-contour" modes for real-time pitch assignment, with parameters for step, range, and non-repetition. The drunk mode based on the idea of random-walk generates discrete sequences of notes, where each note is independent and random. The drunken contour pattern generates continuous melodic contours, with randomly generated key points connected by interpolated notes.

Rule-based methods are strongly interpretable and extendable. Researchers can easily control and shape the output music in terms of different elements without affecting other aspects. However, they require high maintenance and are limited to the designer's musical knowledge.

3.2 Search-based methods

The most common search-based method, Genetic Algorithm (GA), is adopted in many game music studies [17], [20], [21], [22]. Musical generation is represented as an optimisation problem, and the search space here is the set of all possible music pieces generated.

GA can be non-deterministic by providing a diverse set of solutions. It creates a varied population of candidate music to adapt creative tasks. However, this requires delicate fitness function design considering both the quality and novelty of a music piece. Three fitness equations can be used to evaluate the individual music piece of a population in GA:

- Rule-based fitness function - Rules extracted from the original material or external music theory rules.
- Human fitness function - based on direct feedback from the composer [23].
- Learned fitness function - Training a neural network to decide [24].

An example of a rule-based fitness equation is:

$$C_{\text{Step}} = \sum_{n=0}^{N-1} I(n, n+1)(P(n, n+1) + Q(n, n+1))/L \quad (2)$$

Where C_{Step} stands for Counter Step; P for Pre-Counter Step; Q for Post-Counter Step; and L for Leap. All sub-equations are Boolean, and the equation evaluates whether the melody has a nice smooth contour, specifically, whether it reverses motion after a big jump (greater than a second interval). It is a common technique in musical progressions to reverse the movement of the melody after a big jump in order to make the musical progression aurally unobtrusive. Other rule-based fitness functions include melodic novelty, note density, etc. [21].

Unlike pure rule-based systems, rule-based fitness functions provide a more indirect way to select desired music, leaving more space for interesting music candidates. However, it involves high-latency iterations, making real-time deployment challenging.

When more than one fitness equation needs to be satisfied at the same time, the problem is programmed as a multi-objective optimisation (MOO), which can be solved by a method like Non-dominated Sorting Genetic Algorithm II (NSGA-II) [25]. This algorithm is designed to find Pareto bounds - the set of optimal solutions.

However, there are exceptions that use only crossover and mutation techniques of GA due to the subjective nature of fitness equations [26].

3.3 Machine learning

There has been a growing trend of using machine learning methods in game music systems, especially transformer architectures in recent years [11], [12], [27]. This architecture shows surprising potential in commercial areas on large-scale training, such as sunoAI and Udio. However, the data hunger of machine learning and the difficulty in controllability have been research hindrances.

Traditional neural-network architectures have not been completely abandoned either. [20] uses a multi-agent RNN model to handle melody, harmony and other tasks separately. [28] also used RNN to implement a simple adaptive music generation system. [11] used CNN to identify the mood of the game screen, combined with a transformer to generate music. Research using simple probabilistic models such as Markov chains for music generation dwindled.

3.4 Hybrid Method

When involving varied tasks in a complex system, it is common to use a hybrid approach to tackle different challenges [12], [18], [20], [29]. Normally, researchers adopt distinct methods to handle the generation and recognition tasks, for example, interpreting game status as emotion or intensity using a recognition model (Figure 1).

In [20], Hutchings proposed a novel Adaptive Music System (AMS) that combines cognitive models with a multi-agent composition system. The cognitive model is responsible for identifying the emotion of the game content, while three agents cooperate on melody, harmony and percussive rhythm tasks to adapt the pre-composed music to match the emotion in real-time. He incorporated a variant of genetic algorithms, Wilson's eXtended learning Classifier System [30], into his melody agent, while Recurrent Neural Networks were the backend in other agents.

4 Applications

In commercial games, most existing procedural music generation systems are rule-based. These rules are designed to be based on the composer or sound designer's musical knowledge and are game-specific. Compared to the advancement of PCG in the design of levels, terrain, characters, etc., game music is less well regarded and easier to adhere to traditional methods. However, many great games show the potential of applying PMG technology.

Spore is a successful early example of implementing PMG. The prototype was completed in Max/MSP, and adapted to a customised version of PureData [31]. In Spore, the music is stochastically generated in real time from many music samples through a set of rules [9], [31]. Leonard Paul was inspired by Spore and also used PureData to construct his procedural music system for Sim Cell [32]. Graphical music tools like Max/MSP and PureData have also been widely used

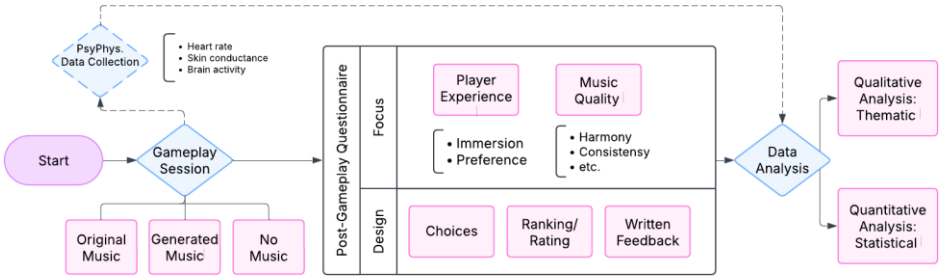


Figure 3. A Flowchart of Evaluation Design.

for implementing PMG in other games, for example, The Legend of Zelda: Tears of the Kingdom [33].

Proteus [34], DOOM [35] and Ape Out [36] all use timbre manipulation as a way of creating variations in the content of the music, expanding somewhat into the realm of procedural audio generation. Proteus is creative in its use of various effects to change the timbre and arrange the music in a non-aligned manner. DOOM creates sequences of varying sounds by stacking many different effects on top of each other, and then samples from a different sequence position each time using a simple rhythmic pattern to achieve a constantly changing timbre. Ape Out generates varied musical content by associating in-game action with percussion sounds.

Both Ape Out [36] and Rez Infinite [37] are highly adaptive music systems, which can also be referred to as reactive music systems, where changes in music or sound are triggered by actions/key presses. Reactive music is a special case of adaptive music, more like an interactive synthesiser, commonly found in rhythm games [16]. Here, the input is mainly the player's interaction data. Unlike Ape Out's timbre mapping, music and sound in Rez Infinite are primarily pitch mapped and are generated procedurally by responding to the player's shooting action with a constant rhythmic beat that synchronises with the character's vibration.

Genesis Noir [38] creatively takes advantage of Jazz improvisation in its music system and brings a unique solution for PMG systems. Recorded music based on preset chord progression provides the overall music ambience, while the player's reaction generates the leading melody in real time. There are two ways music adapts to the gameplay, one of which is by responding to a series of players' inputs through dialogue. For example, the player clicks several buttons to generate a melody sequence, and the system re-arranges it in terms of rhythm and decoration; another way is by instantaneously responding to the interaction. For example, the mouse's motion speed controls the rhythm, and its y-axis coordinates on the screen control the pitch.

Whilst audio middleware is a versatile solution for deploying adaptive music for small to medium-sized games, large games require more complex and flexible solutions to arrange music. As a consequence, they develop independent systems that can integrate seamlessly with the game engine and handle all the interaction logic on how the music adapts to changes in the game. For example, Music Manager in Ara: History Untold [39] is a data-driven system combining both a weighting scheme and stochastic rules. It collects data within a turn and makes a decision about music generation at the end of each turn. Other applications include Pulse in NMS [40], and a server-based music system in The Outlast Trials [41].

5 Evaluation Design

One crucial aspect that differentiates research from application is that researchers typically employ a rigorous assessment to evaluate the effectiveness of PMG systems, which involves multiple gameplay sessions under different music playback conditions [11], [18], [20], followed by post-gameplay questionnaires and optional physiological data collection, shown in Figure 3.

For music systems without real game integration, a purely listening test is often used instead of a game session [12], [29]. Where available, physiological and behavioural data, such as heart rate and skin conductance, are collected during gameplay [13]. This objective data complements the subjective feedback gathered through questionnaires. However, it is more difficult to obtain due to the experimental

conditions. After each gameplay session, participants complete a questionnaire assessing player experience and music quality. The questionnaire includes multiple-choice questions, ranking/rating scales, and sometimes written feedback [18] to capture both quantitative and qualitative insights.

The collected data is then analysed using thematic analysis for qualitative feedback and statistical methods for quantitative data. This dual approach ensures a comprehensive evaluation of the procedural music generation systems. However, most studies investigated in this paper are primarily subjective, focusing on player experience and music quality; few papers have investigated the user experience of composers or sound designers working with the generation tools.

6 Challenges

By analysing research and application practices, we identify three key gaps. First, many academic systems are too experimental or computationally expensive for game developers to adopt. As shown in Figure 4, researchers focus on cutting-edge algorithms—such as reinforcement learning, genetic algorithms, and other advanced methods—that often require prerequisites for practical application, including high-quality datasets, real-time integration, and sophisticated fine-tuning. However, game developers often have to consider the overall allocation of resources in the game. Sound is often not the highest priority [31]. Therefore, they are less willing to experiment with risky or unconventional methods and instead prefer reliable, scalable, and extensible approaches, which has led to rule-based systems dominating the industry.

Second, the quality of many existing academic music systems is currently not up to industry standards. Based on the evaluation results of the academic research investigated in this paper, there is no strong evidence to suggest they provide a more enjoyable or immersive experience for players. A study of interviews with sound designers also confirms that music quality is a major contributor to the lack of acceptance of academic systems [8]. To simplify experimental design, many systems rely on high-level or single parameters—such as “emotion”—to control overall musical output [12], [13], [29], which significantly limits expressivity. In contrast, sound designers employ a wider range of game parameters to fine-tune various musical elements in real-world applications, yielding a much richer listening experience. Yet music quality depends on more than

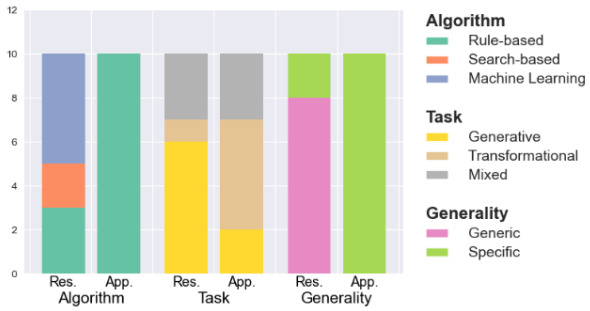


Figure 4. Comparison of Algorithm, Task and Generality in Research and Application

symbolic representation. Many sub-tasks of music generation—such as expressive rendering and modelling musical styles to match different gameplay scenarios—remain complex, unresolved challenges.

A third challenge is the absence of a standardised, user-friendly framework for integrating music systems into games[6]. Deploying these systems in a real game demands specialised expertise and considerable effort, since gameplay mechanics differ widely across styles and developers need solutions that plug directly into popular game engines. In practice, this often requires a dedicated team of audio engineers and access to proprietary assets, making it resource-intensive and sometimes legally complex. The fact that these music systems in academia are difficult to tailor to specific game contexts creates a gap that hampers the collection of in-game feedback and limits the real-world impact and evolution of academic work in this area.

7 Future Directions

Future research should focus on the following key areas to address the identified challenges.

A task-oriented approach that considers practical constraints such as available resources, deployment environments, and specific application requirements is advocated. When employing cutting-edge technologies like machine learning, focus on amplifying their benefits while thoughtfully addressing any limitations. By designing experiments with contextual awareness, researchers can create solutions that are both more extendable and industry-ready. At the same time, game developers should foster a spirit of experimentation with their music systems. Embracing a little risk can unlock richer, more immersive player experiences.

Beyond improving a music system's quality, it is even more important to establish evaluation criteria that accurately measure its contribution within the creative process. Such a framework must inclusively account for every participant involved in generating the music. While a long-term aim of Procedural Music Generation is to move beyond assisted creation toward fully autonomous composition, this shift must always serve to amplify human creativity. It is crucial to acknowledge the system's role as well as its users, ensuring we design tools that avoid stepping on the toes of our creatives [42], [43].

Integrating music into games involves a complex pipeline—DAWs, audio middleware, and the game engine—and many studios even build custom music and audio engines. To streamline procedural music system development, we need reusable, user-friendly research frameworks that plug seamlessly into popular game engines [16]. Studies like [13], [18] show how prototype games can serve as testbeds, giving researchers real-time player feedback to refine their systems and design more robust evaluations, despite the extra effort this requires. Ultimately, building a collaborative platform where researchers and developers share tools, datasets, and best practices will prove invaluable, accelerating innovation, fostering knowledge exchange, and bridging the gap between academic research and real-world game implementation.

8 Conclusion

This paper provides a comprehensive survey and analysis of PMG systems in both research and application domains. By employing a unified taxonomy, we compare these systems across various dimensions, identifying their development patterns, gaps, and challenges. Our analysis reveals that while advanced approaches and generative systems thrive in the research landscape, significant barriers must be conquered to achieve practical application, including deployment difficulties, music quality limitations, and a lack of seamless integration with games.

To address these challenges, we propose future directions that emphasise context-aware approaches tailored to specific tasks, music quality assessment considering human involvement in the creative process, and the development of enhanced research platforms and frameworks. We hope these efforts contribute to the long-term advancement of PMG, fostering both technological progress and practical application in the field.

References

[1] N. Shaker, J. Togelius, and M. J. Nelson, *Procedural Content Generation in Games*. in Computational Synthesis and Creative Systems. Springer International Publishing, 2016.

[2] M. Scirea, 'Affective Music Generation and its effect on player experience', IT-Universitetet i København, 2017.

[3] F. Amato and F. Moscato, 'Formal Procedural Content Generation in Games Driven by Social Analyses', in *2017 31st International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, Taipei, Taiwan: IEEE, Mar. 2017, pp. 674–679.

[4] A. Summerville *et al.*, 'Procedural Content Generation via Machine Learning (PCGML)', *IEEE Trans. Games*, vol. 10, no. 3, pp. 257–270, Sept. 2018.

[5] J. Cullimore, H. Hamilton, and D. Gerhard, 'Directed Transitional Composition for Gaming and Adaptive Music Using Q-Learning', *ICMC*, 2014.

[6] K. Worrall, Z. Yin, and T. Collins, 'Comparative Evaluation in the Wild: Systems for the Expressive Rendering of Music', *IEEE Trans. Artif. Intell.*, vol. 5, no. 10, pp. 5290–5303, 2024.

[7] S. Michael, *Writing interactive music for video games: a composer's guide*. Pearson Education, 2015.

[8] K. Worrall and T. Collins, 'Considerations and Concerns of Professional Game Composers Regarding Artificially Intelligent Music Technology', *IEEE Trans. Games*, pp. 1–13, 2023,

[9] K. Collins, 'An Introduction to Procedural Music in Video Games', *Contemp. Music Rev.*, vol. 28, no. 1, pp. 5–15, Feb. 2009.

[10] D. Menexopoulos, P. Pestana, and J. Reiss, 'The State of the Art in Procedural Audio', *J. Audio Eng. Soc.*, vol. 71, no. 12, pp. 825–847, Dec. 2023.

[11] F. Zumerle, L. Comanducci, M. Zanoni, A. Bernardini, F. Antonacci, and A. Sarti, 'Procedural music generation for videogames conditioned through video emotion recognition', in *2023 4th International Symposium on the Internet of Sounds*, Pisa, Italy: IEEE, Oct. 2023, pp. 1–8.

[12] L. Ferreira, L. Lelis, and J. Whitehead, 'Computer-Generated Music for Tabletop Role-Playing Games', *Proc. AAAI Conf. Artif. Intell. Interact. Digit. Entertain.*, vol. 16, no. 1, pp. 59–65, Oct. 2020.

[13] A. Prechtl, 'Adaptive Music Generation for Computer Games', Open University (United Kingdom), 2016.

[14] J. Tanabe, I. Khan, T. V. Nguyen, C. Nimpattanavong, and R. Thawonmas, 'Adaptive Background Music According to the Player's Arousal for DareFightingICE', in *Proceedings of the 13th International Conference on Advances in Information Technology*, Bangkok Thailand: ACM, Dec. 2023, pp. 1–6.

[15] R. Wooller, A. R. Brown, E. Miranda, R. Berry, and D. Joachim, 'A framework for comparison of process

in algorithmic music systems', *Gener. Arts Pract.*, 2005.
[16] C. Plut and P. Pasquier, 'Generative music in video games: State of the art, challenges, and prospects', *Entertain. Comput.*, vol. 33, p. 100337, Mar. 2020,
[17] M. Scirea, J. Togelius, P. Eklund, and S. Risi, 'Affective evolutionary music composition with MetaCompose', *Genet. Program. Evolvable Mach.*, vol. 18, no. 4, pp. 433–465, Dec. 2017.
[18] A. Lopez Duarte, 'A Progressive-Adaptive Music Generator for Videogames (PAMG): an Approach to Real-Time Algorithmic Composition', UC Riverside, 2023.
[19] F. Lerdahl and R. S. Jackendoff, *A Generative Theory of Tonal Music, reissue, with a new preface*. MIT Press, 1996.
[20] P. E. Hutchings and J. McCormack, 'Adaptive Music Composition for Games', *IEEE Trans. Games*, vol. 12, no. 3, pp. 270–280, Sept. 2020.
[21] D. Plans and D. Morelli, 'Experience-Driven Procedural Music Generation for Games', *IEEE Trans. Comput. Intell. AI Games*, vol. 4, no. 3, pp. 192–198, Sept. 2012.
[22] V. Arutyunov and A. Averkin, 'Genetic algorithms for music variation on genom platform', *Procedia Comput. Sci.*, vol. 120, pp. 317–324, Jan. 2017,
[23] J. A. Biles, 'Interactive GenJam: Integrating real-time performance with a genetic algorithm', *ICMC*, 1998.
[24] J. Biles, P. Anderson, and L. Loggi, 'Neural network fitness functions for a musical IGA', *Present. Scholarsh.*, Mar. 1996.
[25] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, 'A fast and elitist multiobjective genetic algorithm: NSGA-II', *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.
[26] J. A. Biles, 'Autonomous GenJam: Eliminating the Fitness Bottleneck by Eliminating Fitness', *Proc. 2001 Genet. Evol. Comput. Conf. Workshop Program*, 2001.
[27] G. Amaral, A. Baffa, J.-P. Briot, B. Feijó, and A. Furtado, 'An adaptive music generation architecture for games based on the deep learning Transformer model', in *21st Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*, Natal, Brazil: IEEE, Oct. 2022, pp. 1–6.
[28] M. Kopel, D. Antczak, and M. Walczyński, 'Generating Music for Video Games with Real-Time Adaptation to Gameplay Pace', in *Intelligent Information and Database Systems*, Singapore: Springer Nature, 2023, pp. 261–272.
[29] M. Scirea, J. Togelius, P. Eklund, and S. Risi, 'MetaCompose: A Compositional Evolutionary Music Composer', in *Evolutionary and Biologically Inspired Music, Sound, Art and Design*, C. Johnson, V. Ciesielski, J. Correia, and P. Machado, Eds, in Lecture Notes in Computer Science, vol. 9596. Cham: Springer International Publishing, 2016, pp. 202–217.
[30] S. W. Wilson, 'Classifier Fitness Based on Accuracy', *Evol. Comput.*, vol. 3, no. 2, pp. 149–175, June 1995.
[31] K. Jolly, 'Procedural Music in Spore', presented at the Game Developers Conference, 2008.
[32] L. Paul, 'The Generative Music and Procedural Sound Design of Sim Cell', YouTube, 2014.
[33] J. Osada, 'Tunes of the Kingdom: Evolving Physics and Sounds for "The Legend of Zelda: Tears of the Kingdom"', presented at the Game Developers Conference, 2024.
[34] P. Morton, 'The Sound and Music of Proteus - An Academic Case Study'.
[35] M. Gordon, 'DOOM: Behind the Music', presented at the Game Developers Conference, 2017.
[36] *Ape Out - Reactive Music System*, (2020).
[37] T. Mizuguchi, 'Classic Game Postmortem: Rez', presented at the Game Developers Conference, 2016.
[38] J. Abel, 'Genesis Noir: Bringing Jazz Improvisation To Gaming', presented at the Game Developers Conference, June 20, 2022.
[39] P. Klassen, M. Curran, and J. Peros, 'Technical Music Design in Ara History Untold', presented at the Game Sound Conference, CA, Oct. 30, 2024.
[40] P. Weir, 'The Sound of No Man's Sky', presented at the Game Developers Conference, 2017.
[41] T. Salta, 'Scoring and Implementation in a Multiplayer Horror Game', presented at the Game Sound Conference, 2024.
[42] K. Worrall, 'Crafting Better Procedural Music with Deep Learning', presented at the Game Sound Conference, 2024.
[43] B. L. T. Sturm, M. Iglesias, O. Ben-Tal, M. Miron, and E. Gómez, 'Artificial Intelligence and Music: Open Questions of Copyright Law and Engineering Praxis', *Arts*, vol. 8, no. 3, Art. no. 3, Sept. 2019.

Appendix

Author/System	Year	Field	Generality	Task	Algorithm	Direction	Granularity	Grid	KS	Representation
Spore [31]	2008	Application	Specific	Mixed	Rule-based	Horizontal	Note	On	External	Symbolic
Plans [21]	2012	Research	Generic	Generative	GA	Horizontal	Note, Chord	On	External	Symbolic
Proteus [34]	2013	Application	Specific	Mixed	Rule-based	Mixed	Timbre	Off	External	Audio
Sim Cell [32]	2014	Application	Specific	Mixed	Rule-based	Mixed	Note	On	External	Symbolic
Prechtl [13]	2016	Research	Generic	Mixed	Markov chain	Horizontal	Chord, Tonality, Tempo, Velocity	On	External	Symbolic
DOOM [35]	2016	Application	Specific	Transformational	Rule-based	Horizontal	Timbre	On	External	Audio
No Man's Sky [40]	2016	Application	Specific	Transformational	Rule-based	Mixed	Phrase	On	External	Audio
Genesis Noir [38]	2016	Application	Specific	Transformational	Rule-based	Horizontal	Note	On	External	Symbolic
Meta Compose [29]	2017	Research	Generic	Mixed	Stochastic, GA, Rule-based	Mixed	Chord, Melody, Accompaniment	On	External	Symbolic
Rez Infinite [37]	2017	Application	Specific	Generative	Rule-based	Horizontal	Note	Off	External	Audio
AMS [20]	2019	Research	Generic	Transformational	GA, RNN, Graph Model	Horizontal	Note	On	Both	Symbolic
Ape Out [36]	2019	Application	Specific	Generative	Rule-based	Horizontal	Timbre	On	External	Audio
Bardo Composer [12]	2020	Research	Generic	Generative	Transformer, Beam Search	Horizontal	Note	On	Internal	Symbolic
Amaral [27]	2022	Research	Generic	Generative	Transformer	Mixed	Note, Timbre	On	Both	Symbolic
PAMG [18]	2023	Research	Specific	Mixed	Rule-based, Stochastic	Mixed	Note, Timbre	On	External	Symbolic
Kopel [28]	2023	Research	Generic	Generative	RNN	Horizontal	Note	On	Both	Symbolic
Zumerle [11]	2023	Research	Generic	Generative	CNN, Transformer	Horizontal	Note	On	Internal	Symbolic
Tanabe [14]	2023	Research	Specific	Generative	Rule-based	Horizontal	Pitch, Tempo, Velocity	On	External	Symbolic
The Outlast trials [41]	2024	Application	Specific	Transformational	Rule-based	Mixed	Phrase	On	External	Audio
Ara: History Untold [39]	2024	Application	Specific	Transformational	Rule-based	Mixed	Phrase	On	External	Audio

Table 1. Key Procedural Music Systems in Research and Application

Audio Engineering Society

Conference Paper 25

Presented at the AES International Conference on Machine Learning and Artificial Intelligence for Audio
2025 September 8-10, London, UK

This paper was peer-reviewed as a complete manuscript for presentation at this conference. This paper is available in the AES E-Library (http://www.aes.org/e-lib), all rights reserved. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Journal of the Audio Engineering Society.

Establishing a Virtual Listener Panel for audio characterisation

Michelle Herlufsen[1], Niels Asp Fuglsang[1], Benjamin Pedersen[1], Søren Vase Legarth[1]

[1] *SenseLab, FORCE Technology, 2970 Hørsholm, Denmark*

Correspondence should be addressed to Michelle Herlufsen (MCHE@forcetechnology.com)

ABSTRACT

Listening tests provide valuable insight into the sound quality of audio products, but the barriers to conducting such tests are high. They require specialized knowledge, facilities, equipment, time, and access to a panel of trained assessors. The cost alone of maintaining a panel of expert assessors is significant. This study introduces an AI model, the "Virtual Listener Panel", trained using a Convolutional Neural Network (CNN) to overcome these challenges. The purpose of the model is to predict expert ratings for audio characterization. While this paper briefly describes the architecture of the proposed model, the primary focus is on the data collection approach developed to obtain premium quality training data. The performance of the Virtual Listener Panel is evaluated using unseen test data from a listening test in which 20 expert assessors characterized several different earbuds. Prediction performance of five audio characterization attributes from the timbre domain were evaluated: bass strength, bass depth, midrange strength, treble strength and brilliance. Despite the modest dataset size, the model achieved Pearson correlation coefficients between 0.80 and 0.97. The Virtual Listener Panel offers a scalable solution for fast and accurate audio characterization, reducing the need for extensive listening tests and providing valuable insights for all stages of product development.

1 Introduction

Perceptual audio evaluation is crucial for understanding how listeners perceive and appreciate audio products. However, conducting listening tests to collect perceptual ratings is both costly and time-consuming. As a result, objective metrics that predict these ratings are increasingly being used and evolving rapidly. For instance, an open dataset of audio signals with corresponding subjective quality scores is available to accelerate research on perceived audio quality predictions[11]. The International Telecommunication Union (ITU) has standardized ITU-R Rec. P.863 "POLQA" [2] and ITU-R Rec. BS.1387 "PEAQ" [1]. Additionally, several other perceptually driven audio metrics exist, such as 3QUEST [17], VISQOL [3], DPAM [4], CDPAM [5]. These metrics are intrusive methods, meaning they require a clean reference signal. However, this requirement is often impractical in real-world scenarios where reference recordings are unavailable or difficult to obtain. Consequently, non-intrusive metrics—which do not require a reference signal—have been developed using different deep learning architectures, such as Convolutional Neural Network (CNN) [12]-[14],[16],[20]. As a result, state-of-the-art approaches are increasingly relying on deep learning techniques.

All the objective metrics mentioned above focus on speech degradation and mean of opinion scores. While these metrics provides valuable information about consumer preference, they do not provide insight of which characteristics are the *drivers of liking*. This additional insight is crucial for improving

sound quality in a product and would typically require listening test data from trained assessors. In this study we utilize our panel of expert listeners to train a CNN model, the Virtual Listener Panel (VLP), to characterise audio. The process of sensory assessor development to obtain expert assessors [8] requires significant resources and is therefore less common within the audio field. Consequently, the use of expert data in deep learning modelling is limited. Experts are highly trained assessors who are usually used for perceptual objective tests. This is where the human serves as the measuring instrument. The characteristics of the perceived stimuli are evaluated objectively whereas preference and other affective measures are subjective [10]. As an example, we use a wide range of sensory descriptors from the Sound Wheel[7][15], such as clarity, brilliance, bass strength or different spatial cues to characterise audio products. At SenseLab, an expert panel consisting of more than 30 assessors, understands and is repeatedly trained in these attributes. This means that they contribute with high-quality data, and the panel is potentially large enough to deliver sufficient data for a CNN model in this unique area of audio characterisation. A neural network model does not give knowledge of the internal workings, often referred to as a "black box". Developing a model that offers insight to objective perceptual characterisation, which can be used for preference mapping[1], can mitigate the black box effect. Especially, timbre is found to be essential to audio preference. Rumsey et al. [23] found that approximately 70% of the perceptual differences between loudspeakers were related to timbre, for surround sound stimuli downmixed to other loudspeaker configurations. The high importance of timbre leading to preference is also observed for headphones [21][22].

While the collection of training data is the focus of this paper, the VLP model's training architecture will be briefly described first.
Finally, to validate the model predictions, results from a listening test with expert listeners will be compared to the VLP's output.

2 Model framework

Inspired by Manocha et al. [13], that proposed an approach called No-Reference Speech Quality Assessment via Pairwise Preference (SQAPP), we propose a CNN that learns its own prediction scale for audio attribute assessment. The model is trained using 4-second snippets of variable-Q transform power spectra from audio recordings. Essentially, we present a simple architecture that, during training, learns by comparing pairs of recordings, but during inference, it requires only single recordings to make predictions.

The VLP model consists of three main components: the *Backbone CNN*, *Attribute Encoder*, and *Rating Predictor*. The training architecture is illustrated in Figure 1.

Backbone CNN: To extract features from spectrograms of audio recordings, we use a pretrained ResNet-18 [24] as the model's backbone. The backbone choice is flexible and can be replaced to adjust the model's complexity.

Attribute Encoder: A multilayer perceptron (MLP) that converts a one-hot-encoded variable into a feature vector.

Rating Predictor: An MLP that maps the combined features from the backbone and attribute encoder to a scalar rating.

With the proposed architecture, we can train the model on various attributes, enabling it to discover non-linear relationships between them across different music genres - all without relying on a reference recording.

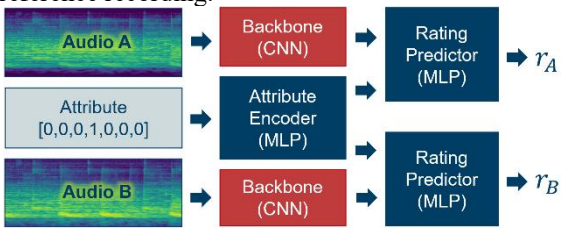


Figure 1) The training architecture of the VLP model using pairwise comparison.

3 Data collection

Data collection is a crucial component in constructing a reliable neural network model. The quantity and the quality of the data are balanced critical factors in generating accurate deep learning outputs. The high quality and volume of data were ensured using

[1] Preference mapping links consumer preference to the sensory characteristics of products. This approach provides a reliable basis for adapting or creating products to meet consumer expectations and optimize preference.

SenseLab's expert listener panel [8][9]. They undergo annually testing with a passing criterion for discrimination and reproducibility abilities to ensure they contribute with reliable data. Furthermore, they receive training in different test types, which includes audio characterization using the sensory descriptors from the Sound Wheel [7]. This results in agreement among the panel in understanding and rating the attributes. Every second year, high-frequency audiograms are measured to confirm normal hearing.

3.1 Experimental setup

Each expert listener was provided with a home-kit containing high-resolution, low-noise sound cards (FiiO K3) and high-fidelity headphones (Sennheiser HD650). Assessors were instructed to perform the listening tests only when feeling alert and when background noise at home was minimal. Each listening test was estimated to last approximately two hours, including breaks. A written test instruction was always included.

All participants rated attributes on a horizontal continues scale with verbal anchors in a multiple stimulus test method (cf. Figure 2), using SenseLabOnline – a web-based service for administering, performing and analyzing listening tests [6]. The corresponding attribute definition was present on the same screen.

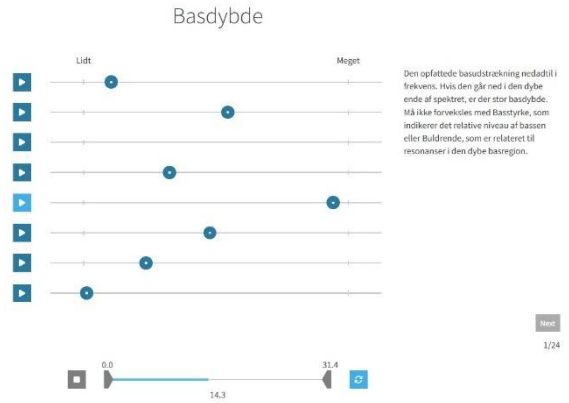


Figure 2) Screenshot example of the data collection process: Multiple stimuli test where experts provided their attribute ratings.

All stimuli in the listening test were loudness-aligned according to ITU-R BS.1770-4 [18]. Participants were instructed to adjust the level to the most comfortable using the sound card in the beginning of the session and to maintain that level throughout the test.

For the five attributes listed below, approximately 38,000 ratings comparing various hybrid and augmented stimuli were collected from experts. These ratings were subsequently transformed into approximately 159,000 pairwise comparisons for training the VLP model.

The Attributes and their corresponding definitions were selected from the existing Sound Wheel. The initial set of attributes used to train the model included:

1) Bass depth
2) Bass strength
3) Midrange strength
4) Treble strength
5) Brilliance

For each attribute, music samples were carefully selected to ensure suitability for evaluating the specific attribute. Consequently, the samples varied according to the specific attribute being assessed. However, pink noise was consistently included as one of the samples in all tests.

The samples used in the test were excerpts from original music tracks, typically 20-30 seconds in duration.

Although being classified as expert listeners, assessors were required to pass a gating test before participating in the actual data collection. To pass, they had to rate 4-6 systems in the correct order twice, with increasing difficulty. If they failed, they received immediate feedback and were given the opportunity to try again.

To ensure the highest data quality, additional evaluations were used for post-screening after the full dataset was collected. If the responses in these trials were incorrect (given in a wrong order), the corresponding dataset for that expert and attribute was excluded.
The test sequence followed this order:

- Gating (with immediate feedback)
- Post-screening
- Actual data collection for VLP
- Post-screening repeated

3.2 Stimuli generation

To optimize the variation in training material for the model, the variations were not entirely random. Measured impulse responses from real headphone

devices were convolved with the selected music excerpts. Additionally, various filters were applied to introduce a broad range of reasonable frequency response variations. Additional filter variations known to cause perceptual changes in the specific attribute were applied to achieve diverse ratings for the training material.
Based on our experience, having an assessor rate a single attribute for an extended period can lead to confusion and reduced rating accuracy. To mitigate this, anchors were frequently included throughout the test to stabilize scale usage.

Figure 3 shows an example of final frequency response variations used to generate stimuli for the data collection of *bass depth*. It is important to note that all experts had unique tests—no two assessors rated the same stimuli (except for the gating and post-screening). This approach maximized the variation observed by the model.

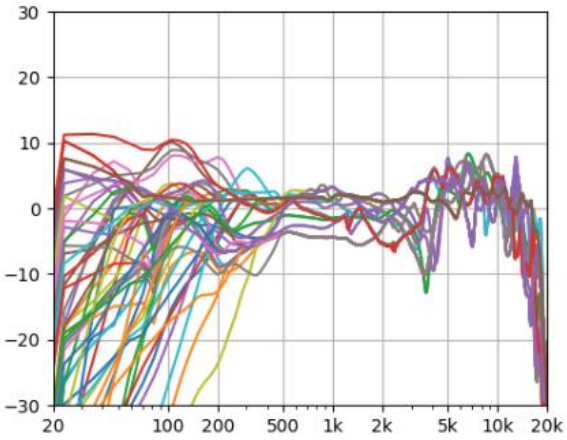


Figure 3) Example of filters that were used to process the music samples to generate stimuli for an assessor. The horizontal axis is frequency in Hz and the vertical axis is gain in dB.

4 Results

In this section, a separate listening test of earbuds is utilized as a validation dataset, to test the reliability of the VLP model.
The listening test was conducted by 20 assessors from SenseLab's expert panel in sound-insulated IAC listening booths using Sennheiser IE 900 in-ear headphones.
A total of seven midrange earbuds were recorded on a B&K 5128 HATS with five music samples of 20-30 sec length:

1. Argon Audio, IE20
2. boAt, Nirvana Ivy
3. Jlab, JBuds ANC 3
4. Samsung, Galaxy Buds FE
5. SkullCandy, Rall ANC
6. Soundcore, Liberty 4 NC
7. Urbanista, Palo Alto

None of the recordings have been used to train the VLP model. However, up to two of the five music excerpts were similar to those included in the VLP training data for certain attributes.
In the listening test a "reference" and "anchor" were included to encourage the experts to use most of the scale. The anchor was created by applying a bandpass filter with the passband frequency range of 100 to 7000 Hz. While the reference and anchor are included in the results section, it is important to note that the VLP remains a non-intrusive method. The model's predictions are unaffected by the presence or absence of a reference or anchor and yield identical results either way.

The results for each attribute are plotted, as shown in Figure 4-8. To ensure easy visual comparability, a min-max normalisation method is applied to both assessor ratings and VLP predictions. The output of the VLP is an arbitrary scale which enables relative comparisons (ranking and perceptual distance). While many models aim for prediction of the absolute rating, most or all data in perceptual audio evaluation is relative, although efforts to make scales used in a stable absolute sense has been made [25]. Because the output is relative, normalizing such that the ratings and prediction have the same min and max is needed.

The VLP generates scores in 4-second time slices at fixed intervals (time delta), e.g., every 1 second, resulting in multiple predictions across each sample. The error bars shown in the figures for the VLP predictions represent the standard deviation of the predictions across time and across all five music

excerpts, using a fixed interval (time delta) of 2 seconds.

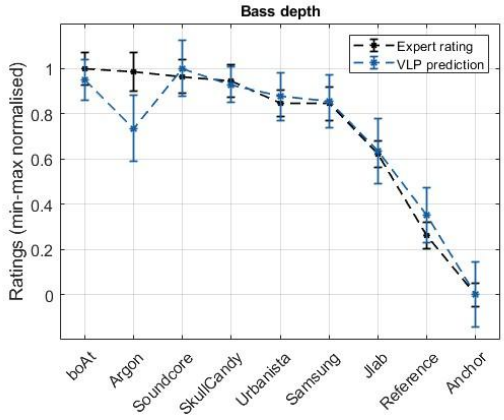


Figure 4) Bass depth evaluations. They are ordered according to the average expert ratings.

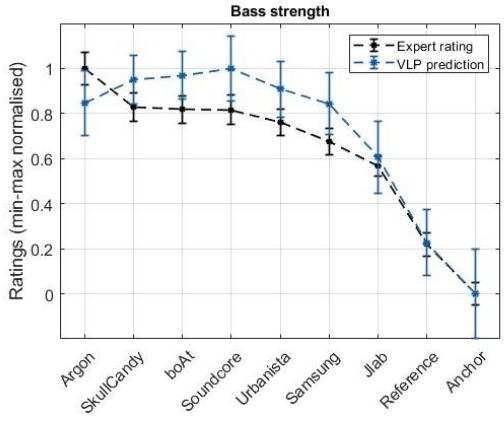


Figure 5) Bass strength evaluations. They are ordered according to the average expert ratings.

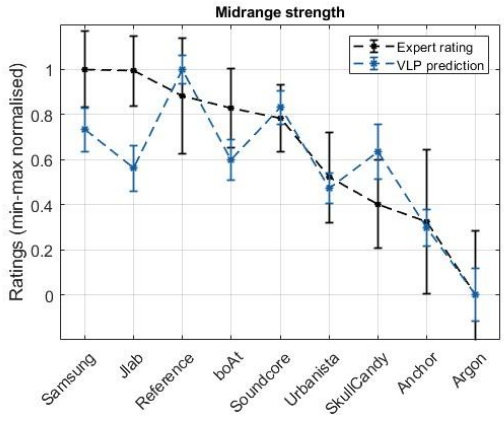


Figure 6) Midrange strength evaluations. They are ordered according to the average expert ratings.

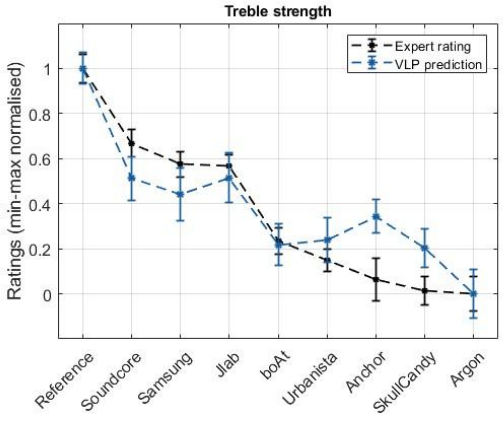


Figure 7) Treble strength evaluations. They are ordered according to the average expert ratings.

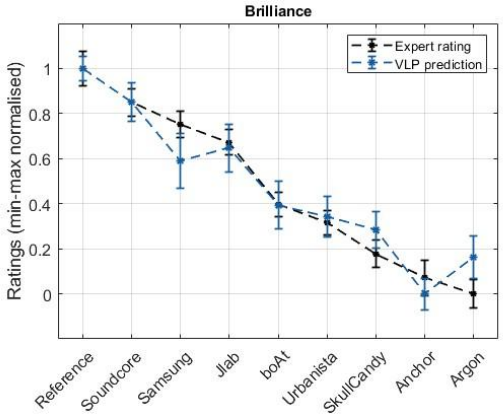


Figure 8) Brilliance strength evaluations. They are ordered according to the average expert ratings.

	Pearson	Spearman	RMSE	RMSE*
Bass depth	0.96	0.80	0.098	0.029
Bass strength	0.95	0.77	0.135	0.030
Brilliance	0.97	0.97	0.094	0.020
Treble strength	0.92	0.88	0.145	0.041
Midrange strength	0.80	0.63	0.219	0.042

Table 1. Results of the predictive VLP model with root mean square error (RMSE) and a modified version (RMSE*), Pearson-, and Spearman correlation coefficient.

In Table 1, the accuracy of the prediction model is evaluated using correlation coefficients and the root mean square (RMSE) evaluation metric. Furthermore, modified epsilon-insensitive rmse

(RMSE*) values, which take the uncertainty of the listening test scores into account [19], are included. To calculate the RMSE and RMSE* values, the rating and prediction scores are min-max normalised.

5 Discussion

The purpose of this study was to investigate whether it is possible to train a neural network model for the use of audio characterisation. Controlled laboratory listening test and maintenance of an expert panel for such characterisations are both costly and time consuming. In SenseLab we are in the unique position to have a trained listener panel. The CNN model's performance is illustrated in Figure 4-8 and summarized in Tabel 1: The comparative analysis demonstrates a high validity and performance of the prediction model. The prediction model's validity for midrange strength evaluations is inferior compared to the other attribute predictions. However, the experts' ratings from the listening test, also exhibit greater difficulty in reaching consensus on midrange strength ratings, as seen by the relatively wider confidence intervals compared to the other attribute evaluations. Hence, the RMSE* value reduces significantly compared to the traditional RMSE value, due to the uncertainty from the expert ratings itself. Midrange strength can be a difficult attribute to rate compared to bass- and treble strength. So, the larger disagreement with expert ratings could be due to difficultly in rating this attribute, both from the listening test itself but also from the training material collected for the VLP model.

While predictions of some attribute ratings can certainly be improved, the VLP is already useful at this performance level. One example is for selection of music samples for listening test; Here the VLP can predict which will give the largest differentiating between products and where the excerpt is stable enough to provide similar ratings across assessors focusing on different parts of the excerpts.

In machine learning in general, data augmentation is an important tool, because data collection can be expensive. However, within audio, even small changes to the stimuli might lead to audible differences that would lead to a different rating. The lack of published knowledge on Just Noticeable Differences of complex stimuli makes these decisions harder. Using the trick of presenting stimuli in a multiple stimulus method but converting them to paired comparison ratings is a way of multiplying the collected data with a significant factor. Although increasing the number of ratings rather than the number of stimuli, it is still very valuable. When combined with the pooled attribute data model architecture, it makes it possible to train an attribute prediction model based on one (large) test, rather than dozens or hundreds of tests, assuming excellent data quality. With each new attribute we collect data on, predictions for other attributes are likely to improve as well, as the model continuously refines its ability to identify non-linear relations – mirroring the auditory system. Currently, the model has been trained on three more attributes—timbral balance, nasal and boxy—which needs to be validated with expert listening test data.

6 Concluding remarks

In this paper, we propose a method for efficient collection of high-quality data for an audio characteristic prediction model. Starting with five timbral aspects, known to be an important domain in terms of drivers of liking. The Pearson correlation ranged from 0.80-0.97, when tested on a dataset from an independent expert listening test. To maximize the data quality, we implemented the following measures:

- Used trained listeners (experts) to provide the training material for the audio characterisation prediction model, VLP.
- Conducted data collection with known and identical hi-fi equipment.
- Required passing a gating test and post-screening.
- Included reference stimuli to anchor scale usage.
- Used carefully selected music excerpts matched to each attribute.
- Created relevant, audible and varied variations in stimuli.

The scope of the model in this study was tested only with expert data from recordings of earbuds. It is still unknown how well the model performs with recordings of other audio devices, such as loudspeakers.

References

[1] A. W. Rix, J. G. Beerends, et al., “Perceptual evaluation of speech quality (PESQ)-a new method for speech quality assessment of telephone networks and codecs,” in ICASSP, 2001, vol. 2, pp. 749–752.

[2] J. G. Beerends, C. Schmidmer, et al., “Perceptual objective listening quality assessment (POLQA), the third generation itut standard for end-to-end speech quality measurement part i—temporal alignment,” Journal of the AES, no. 6, 2013

[3] A. Hines, J. Skoglund, et al., “ViSQOL: An objective speech quality model,” EURASIP, vol. 2015, no. 1, 2015.

[4] Manocha, A. Finkelstein, et al., “A differentiable perceptual audio metric learned from just noticeable differences,” Interspeech, 2020.

[5] P. Manocha, Z. Jin, et al., “CDPAM: Contrastive learning for perceptual audio similarity,” ICASSP 2021.

[6] “SenseLabOnline” (version 5.0) [Online]. Available: https://forcetechnology.com/en/services/acoustics-noise-sound-quality/senselabonline

[7] SenseLab’s “Sound Wheel” [Online]. Available: https://forcetechnology.com/en/services/acoustics-noise-sound-quality/senselab-sound-wheel

[8] S. V. Legarth and D. N. Zacharov, "Assessor Selection Process for Multisensory applications," in *Audio Engineering Society*, München, Germany, 2009.

[9] ISO 8586-2, “Sensory analysis — General guidelines for the selection, training and monitoring of selected assessors – Part 2: Experts”. International Standard ISO

[10] T.H. Pedersen, “The Semantic Space of Sounds”, 2008.

[11] M. Torcoli et al., “ODAQ: Open Dataset Of Audio Quality,” Dec. 2023, doi: 10.5281/zenodo.10405774.

[12] Mittag, Gabriel & Chehadi, Assmaa & Möller, Sebastian. (2021). NISQA: A Deep CNN-Self-Attention Model for Multidimensional Speech Quality Prediction with Crowdsourced Datasets.2127-2131.10.21437/Interspeech.2021-299.

[13] P. Manocha, Z. Jin, and A. Finkelstein, “SQAPP: No-Reference Speech Quality Assessment Via Pairwise Preference.” [Online]. Available: https://pixl.cs.princeton.edu/pubs/Manocha

[14] A. H. Andersen, J. M. de Haan, Z. -H. Tan and J. Jensen, "Nonintrusive Speech Intelligibility Prediction Using Convolutional Neural Networks," in *IEEE/ACM*, vol. 26, no. 10, pp. 1925-1939, Oct. 2018.

[15] Report ITU-R B.2399-0, “Methods for selecting and describing attributes and terms, in the preparation of subjective tests”, ITU 2017.

[16] A. R. Avila, H. Gamper, C. Reddy, R. Cutler, I. Tashev, and J. Gehrke, “Non-intrusive Speech Quality Assessment Using Neural Networks,” in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, Institute of Electrical and Electronics Engineers Inc., May 2019, pp. 631–635.

[17] ETSI EG 202 396-3, “Speech and multimedia Transmission Quality (STQ); Speech Quality performance in the presence of background noise; Part 3: Background noise transmission – Objective test methods”, 2018.

[18] Report ITU-R B.1770-4, “Algorithms to measure audio programme loudness and true peak level”, ITU 2015.

[19] Recommendation ITU-T P.1401, “Methods, metrics and procedures for statistical evaluation, qualification and comparison of objective quality prediction models”, ITU 2020

[20] P. Manocha, B. Xu, and A. Kumar, “NORESQA: A Framework for Speech Quality Assessment using Non-Matching References,” Sep. 2021, [Online]. Available: http://arxiv.org/abs/2109.08125

[21] S. E. Olive and T. Welti, “The Relationship between perception and Measurement of Headphone Sound Quality,” 2012, [Online]. Available: www.aes.org

[22] G. Ravizza, J. Villegas, C. P. Volk and Tore Stegenborg-Andersen, “An Over-Ear Headphone Target Curve for Brüel & Kjær Head And Torso Simulator Type 5128 measurements” in 155th Convention Express Paper, 2023.

[23] F. Rumsey, S. Zieliński, R. Kassier, S. Bech, “On the relative importance of spatial and timbral fidelities in judgments of degraded multichannel audio quality”, 2005, The Journal of the Acoustical Society of America, vol. 118.

[24] K. He, X. Zhang, S. Ren & J. Sun, “Deep Residual Learning for Image Recognition”, Microsoft Research, 2015, [Online] Available: https://arxiv.org/abs/1512.03385

[25] Lim J, Wood A, Green BG, “Derivation and evaluation of a labeled hedonic scale”, 2019, Chem Senses, 34, 739-751.

Audio Engineering Society

Conference Paper 26

Presented at the AES International Conference on Artificial Intelligence and Machine Learning for Audio
2025 September 8-10, London, UK

This paper was peer-reviewed as a complete manuscript for presentation at this conference. This paper is available in the AES E-Library (http://www.aes.org/e-lib) all rights reserved. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Journal of the Audio Engineering Society.

A Machine Learning approach to modal control in small rooms

Carlo M. Bolla[1], Trevor J. Cox[1], and Bruno M. Fazenda[1]

[1]Acoustic Research Centre, University of Salford, Salford M5 4WT, United Kingdom

Correspondence should be addressed to Carlo M. Bolla (c.m.bolla@edu.salford.ac.uk)

ABSTRACT

Reducing the decay time of low-frequency resonances in small rooms remains a critical challenge in audio engineering, as these resonances can degrade sound quality, leading to a "boomy" or "muddy" audio experience. Traditional equalization methods, while effective in certain contexts, rely heavily on static equalization filters, making them suboptimal for complex scenarios. This paper explores a novel machine learning-based approach to suppress low-frequency resonances by leveraging a modified Wave-U-Net architecture. The model incorporates psychoacoustic principles, specifically perceptual modal thresholds, to ensure that resonance suppression is guided by human auditory perception while preserving the natural characteristics of the audio. A large dataset of synthetic Room Impulse Responses was generated using a physically informed parametric model for cuboid rooms with randomized dimensions and absorption profiles. The model was trained on amplitude-modulated white noise bursts to simulate broadband excitation. Evaluation on unseen synthetic data demonstrates effective suppression of low-frequency resonances below perceptual thresholds. Limitations in generalization are discussed, along with future directions that include retraining on musical stimuli and incorporating perceptual loss metrics.

1 Introduction

Low-frequency resonances are a challenge in audio engineering, can impact the quality and clarity of audio in both recording and playback environments. These resonances, result in excessive buildup of energy at certain frequencies, leading to a "boomy" or "muddy" sound [1]. Addressing these is critical to ensuring that audio recordings and mixes translate well across different listening systems [2] [3] [4].

Conventional strategies for suppressing low-frequency room resonances have historically centred on the application of static equalisation filters, subsequently refined through expert manual adjustment to achieve the desired tonal balance.

Traditional approaches to managing low-frequency resonances typically involve the use of static equalization (EQ) filters [5]. These methods can be time-consuming, particularly in complex audio scenarios. Moreover, overuse of EQ can introduce unwanted artifacts, further compromising the audio quality [6]. Automated tools, such as dynamic EQs and resonance suppressors, have made strides in streamlining this process but often lack the precision needed to account for perceptual factors critical to listening experience [5]. Recent advancements in deep learning have demonstrated significant potential for using machine learning to address complex audio processing tasks.

Among these is Wave-U-Net, [7] a convolutional neural network architecture originally designed for audio source separation This has shown promise due to its ability to model time-frequency relationships effectively. It has been applied to tasks such as source separation [7], speech enhancement [8], and de-

reverberation [9]. However, its application to the low-frequency resonance suppression is yet to be fully explored.

This paper therefore presents an adaptation of the Wave-U-Net architecture to filter low-frequency resonances in audio. Our approach integrates psychoacoustic principles to align the filtering process with auditory perception.

By considering Perceptual Modal Thresholds (PMTs) [10] which define the threshold beyond which modal resonances become perceptually intrusive, we aim to suppress resonances to below perceptual significance rather than complete elimination.

Integrating these principles into deep learning models offers a promising avenue for perceptually-aware audio processing. Audio samples are provided online[1].

2 Method

A modified Wave-U-Net architecture was trained using supervised learning to reduce resonant room modes in reverberant signals to perceptually acceptable levels.

Primary focus was placed on verifying whether the proposed system could automatically detect modal frequencies whose decay exceeds the PMTs and adjust their decay accordingly, all within an end-to-end architecture.

While the model's non-causal design inherently precludes real-time processing, this was an exploratory phase, and other deep learning architectures suitable for real time processing will be tested later.

Given the impracticality of capturing real-world Room Impulse Responses (RIRs) across diverse acoustic conditions, we employed a parametric modal decomposition model derived from the formal solution of the wave equation [11] to synthesize a comprehensive training dataset.

Such a model, validated in other works [12] for accurately reproducing frequency responses of enclosed spaces within our target low-frequency range (20-240 Hz), enabled generation of approximately 2.7 million synthetic RIRs.

The dataset encompassed varied room dimensions, absorption characteristics, and randomized source-receiver placements (defined in Section 2.2), with careful constraints to avoid corner positions that could introduce bias through exaggerated modal effects.

The 240 Hz frequency cap was selected to focus computational resources on the frequency range most relevant to room modal behaviour while maintaining practical training efficiency at the reduced sampling rate of 480 Hz.

This sampling rate, though limiting for full-bandwidth applications, provided sufficient

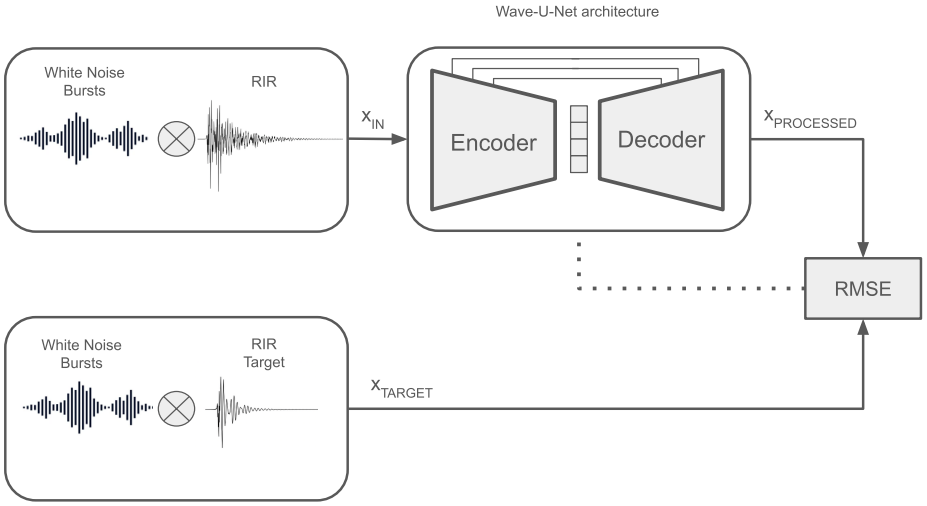


Fig. 1 - Overview of the training pipeline. The model processes an input signal (X_{IN}) and generates a predicted output ($X_{PROCESSED}$), which is compared to a target signal (X_{TARGET}) using time domain root mean square error (*RMSE*) as a performance metric.

[1] https://carlo-bolla.github.io/aes2025-ml-ai-audio/

resolution for the low-frequency modal control task while enabling reasonable computational requirements during training and final model size.
Such an architecture can be integrated as a component within a multi-rate filtering algorithm.

Fig. 1 illustrates the supervised learning framework employed for training the Wave-U-Net architecture. The input signals (X_{IN}) are synthesized by convolving modulated white noise burst (defined in Section 2.1) excitations with RIRs, while the corresponding target signals (X_{TARGET}) are generated through convolution of the same excitation signals with perceptually-optimized target RIRs. These target RIRs are derived from their reverberant counterparts by selectively reducing modal decay times to meet PMTs, ensuring that resonance suppression aligns with human auditory perception.
The framework leverages the commutative property of convolution operations, allowing the training process to focus on learning the mapping between resonant and perceptually-optimized acoustic responses while maintaining computational efficiency.

2.1 Excitation Signal

The excitation signals employed in the dataset consisted of amplitude-modulated white noise bursts, specifically designed to provide uniform spectral excitation across the entire target frequency range (20-240 Hz). White noise inherently provides full-spectrum stimulation with equal energy distribution across all frequencies.
To enhance dataset diversity and reduce the likelihood of the model learning fixed temporal patterns leading to potential overfitting [13], amplitude modulation was applied to the white noise bursts with randomized temporal parameters. These modulation parameters were systematically varied according to the following constraints:

$$\begin{cases} 0.3\ Seconds \leq L \leq 2.0\ Seconds \\ 0.01 \leq \dfrac{A}{L} \leq 0.20 \\ 0.10 \leq \dfrac{D}{L} \leq 0.80 \end{cases} \quad (1)$$

Where:

- L is the white noise burst length;
- A is the attack time;
- D is the decay time

The amplitude envelope transitions during attack and decay phases were shaped using a Blackman window function to eliminate abrupt discontinuities that could introduce spectral artifacts. This modulation approach generated a comprehensive range of excitation signals with varying temporal characteristics, potentially supporting improved model generalization to diverse resonant environments.
Multiple modulated bursts were concatenated to form extended sequences, further increasing dataset variability and enabling the model to process longer reverberant decay characteristics rather than isolated transient events.
To maintain consistency in the training pipeline, both input and target signals were standardized to 4-second durations. This temporal window was selected to ensure complete capture of modal decay characteristics within the target frequency range while balancing computational efficiency with the temporal resolution required for effective modal equalization analysis.
The 4-second duration provides sufficient temporal context for the model to analyse the full decay envelope of low-frequency room modes, which typically exhibit longer decay times compared to higher-frequency components.

2.2 Room Impulse Response

The synthetic RIRs were generated using modal decomposition formulated for cuboid rooms with rigid boundaries [11].
The modal accounts for modal pressure distributions, resonance frequencies, and damping effects based on the three-dimensional coordinates of both source and receiver positions. The source is modelled as an omnidirectional point source.
The room transfer function is formulated [12] as:

$$p_w(sr) = j\rho Q c^2 \sum_n \frac{p_n(s)p_n(r)}{K_n(\omega^2 - \omega_n^2 - 2j\delta_n\omega_n)} \quad (2)$$

Where:

- ρ is the air density $[kg/m^3]$;
- Q is the source strength $[m^3/s]$;
- c is the speed of sound $[m/s]$;
- n is the n^{th} order mode;
- $p_n(s)$ and $p_n(r)$ correspond to the modal pressure distributions at the receiver and source positions respectively;
- K_n is a mode-specific scaling factor;
- $\omega\ is$ the driving angular frequency of the source $[rad/s]$;
- ω_n defines the resonant angular frequency of the n^{th} mode $[rad/s]$;
- δ_n the modal damping coefficient,

The modal damping coefficient of a single resonant frequency, can also be expressed as a function of the modal decay as follows:

$$\delta_n = \frac{2.2\ \pi}{MD_{60,\mathrm{n}}} \cong \frac{6.91}{MD_{60,\mathrm{n}}} \tag{3}$$

Where $MD_{60,n}$ represents the modal decay time of the *n-th* resonant frequency ω_n.
It is computed as per [14] and depends on the room's absorption properties. The absorption coefficient α is assumed to be homogeneous across all room surfaces, meaning that all walls, floor, and ceiling have identical acoustic absorption properties.
The RIR target is derived with the same methodology except that $MD_{60,\mathrm{n}}$ is obtained from the refactored formulation of the PMTs [15]:

$$MD_{60,\mathrm{n}} = 0.15 + \frac{755}{f_n^2} \tag{4}$$

Where f_n is the resonant frequency of the n^{th} mode expressed in Hz.

The room configurations and placement rules were defined by the following parameters:

- Room dimensions: $\vec{L} = (L_x, L_y, L_z)$;
- Receiver position: $\vec{r} = (x_r, y_r, z_r)$;
- Source position: $\vec{s} = (x_s, y_s, z_s)$;
- Absorption coefficient α.

The room dimensions were randomly assigned based on uniform distributions, with the following constraint applied to prevent cubic or nearly cubic configurations:

$$\vec{L} = \begin{cases} 2.5\ m \le L_x < 6.1\ m \\ 2.5\ m \le L_y < 6.1\ m \\ 2.5\ m \le L_z < max(L_x, L_y) \end{cases} \tag{5}$$

Source constraints:

$$\vec{s} = \begin{cases} 0.15\ m \le x_s < L_x - 0.15\ m \\ 0.15\ m \le y_s < L_y - 0.15\ m \\ 0.30\ m \le z_s < 0.50\ m \end{cases} \tag{6}$$

Absorption coefficient:

$$0.05 < \alpha < 0.5 \tag{7}$$

2.3 Model Architecture

The proposed architecture (Fig. 2a) builds upon the Wave-U-Net framework [7], a fully convolutional neural network specifically designed for end-to-end time-domain audio processing.
The model employs an encoder-decoder structure with hierarchical downsampling and upsampling operations to efficiently extract multi-scale audio features while maintaining fine temporal resolution necessary for accurate modal frequency detection.
The encoder (Fig. 2b) consists of *M=5* convolutional blocks, each implementing a downsampling factor of 2 through stride-based 1D convolutions followed by PReLU activation functions. These stride-based convolutions progressively reduce the temporal resolution of the input signal, enabling the model to learn hierarchical representations of the audio.
The decoder structure (Fig. 2c), consists of N=5 symmetric blocks that systematically restore temporal resolution through learned upsampling operations. Each decoder block begins with feature map upsampling to match the corresponding encoder resolution, followed by multiple 1D convolutional

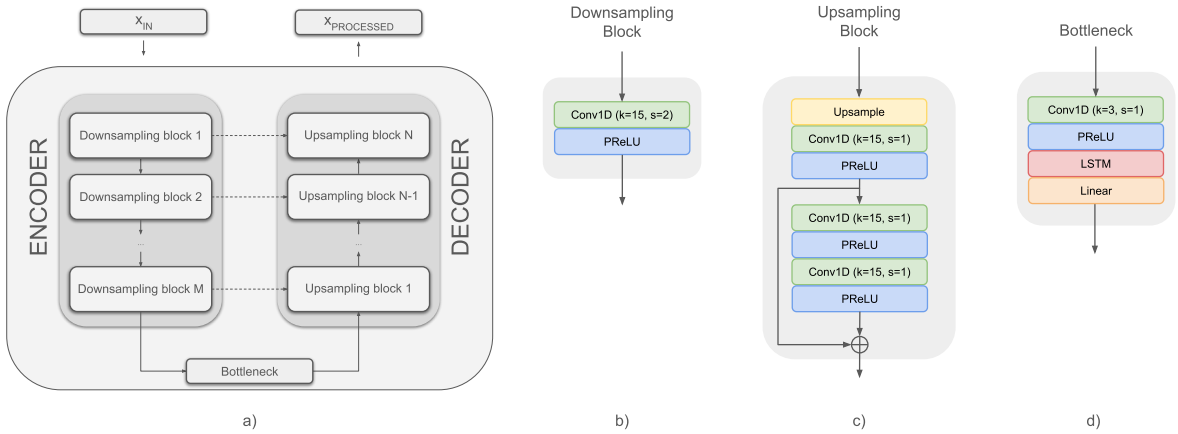


Fig. 2 - Model architecture schematics: a) Wave-U-Net, b) Encoder block, c) Decoder Block, d) Bottleneck.

layers with PReLU activations that progressively refine the reconstructed features. Skip connections merge decoder outputs with corresponding encoder features enhancing training stability.

At the bottleneck (Fig. 2d), a 1D convolutional layer with 144 channels further refines the learned representation, followed by a Long Short-Term Memory (LSTM*)* module with two layers, which enables the model to capture long-range dependencies in the signal to better accommodate low-frequency room mode equalization.

The addition of LSTM networks within the model addresses the need to capture long-range temporal dependencies inherent in processing the decay of low-frequency resonances. This integration has been explored in various studies, such as the work on low-latency live speech enhancement [16], where combining Wave-U-Net with LSTM modules improved the model's ability to handle temporal dynamics in speech.

Employing multiple convolutional layers per decoder block [17], enhances waveform reconstruction by progressively refining upsampled features, capturing temporal dependencies more effectively.

2.4 Training Process

The dataset was divided into training (80%), validation (10%), and test (10%) sets to ensure robust model generalization across the synthesized RIRs.

The data was fed into the model in batches of 64 samples, balancing computational efficiency with variability across different room conditions in each batch.

The model was optimized using the Root Mean Squared Error (RMSE) loss function computed in the time domain:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_{PROCESSED,i} - x_{TARGET,i})^2} \qquad (8)$$

where N represents the total number of samples in each 4-second training sequence. This loss function was selected to minimize the temporal deviation between processed and target signals while preserving the overall energy characteristics of the modal suppression.

AdamW [18] optimization was employed due to its ability to adaptively adjust learning rates, enhancing stability during training.

The initial learning rate was set to 0.0001 and progressively decayed every 50 epochs to ensure smooth convergence. The model was trained for 250 epochs, with training performance monitored using the validation set.

To prevent excessive training durations and mitigate the risk of overfitting, an early stopping criterion was implemented. Training was automatically halted if the validation *RMSE* did not improve for 20 consecutive epochs.

3 Results

The performance of the modal control model was evaluated using dataset samples from the test subset, which were not included in training or validation, ensuring an unbiased assessment.

Fig. 3, Fig. 4 and Fig. 5 present results for three different room volumes excited with three different white noise bursts randomly generated according to Section 2.1.

The analysis focuses on the time-domain amplitude response and the frequency-domain magnitude response within the 20–240 Hz range. In all cases, the excitation signal was randomized to maintain consistency across test conditions (Section 2.1).

The time-domain amplitude response plots show the envelopes of the broadband signal, where the processed signal exhibits a reduction in the decay portion compared to the input signal while maintaining the attack of the white noise bursts.

The extent of decay attenuation varies across room volumes, with the processed signal following the target response closely in most cases.

Fluctuations in the processed signal are greater at smaller amplitudes (< -40 dBFS), which should be perceptually negligible.

A similar behaviour is observed in the frequency-domain magnitude response plots.

The resonant room modes in the input signal exhibit higher peaks, whereas the processed signal presents smoother peaks that more closely match the target response. In the high frequency regions, where higher modal order is observed, the processed spectra resemble a gain-adjusted version of the input signal, exhibiting similar features to the target magnitude response.

Additionally, on average, the magnitude values of the processed spectra are lower than the target, indicating a potential underestimation in gain compensation across frequencies. This suggests that the model adapts its spectral shaping to match the target while preserving the overall structure of the frequency response.

4 Discussion

The proposed model is successful in identifying and attenuating room modes, effectively reducing their

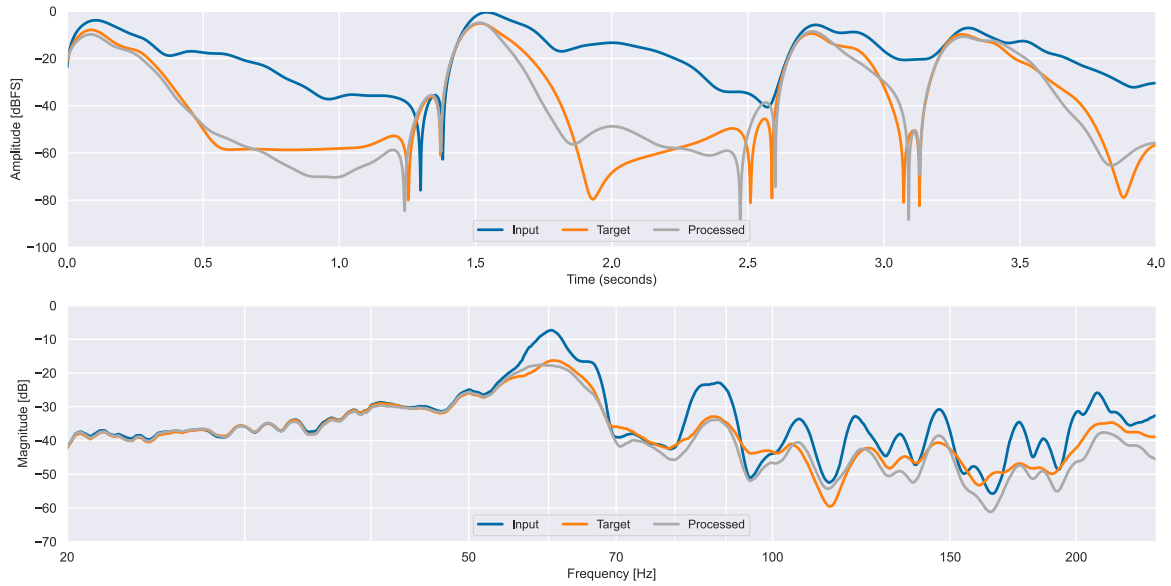


Fig. 4 – Room Volume 21 m^3 -Top: Time response (scaled in dBFS) of the Input (blue), Target (orange), and Processed (grey) envelopes.– Bottom: Frequency response of the Input (blue), Target (orange), and Processed (grey) signals.

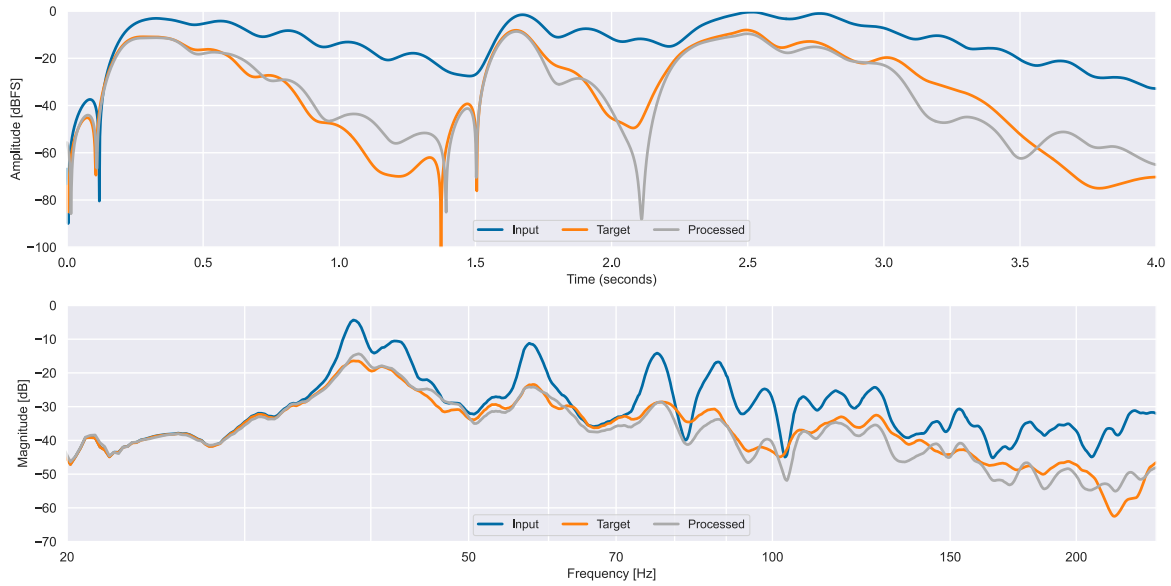


Fig. 3 – Room Volume 67 m^3 -Top: Time response (scaled in dBFS) of the Input (blue), Target (orange), and Processed (grey) envelopes.– Bottom: Frequency response of the Input (blue), Target (orange), and Processed (grey) signals.

decay to levels consistent with PMTs (orange curve in the time-domain plot). Traditionally, the detection and mitigation of room resonances rely on empirical methods, often requiring iterative adjustments. In contrast, this model provides a systematic and data-

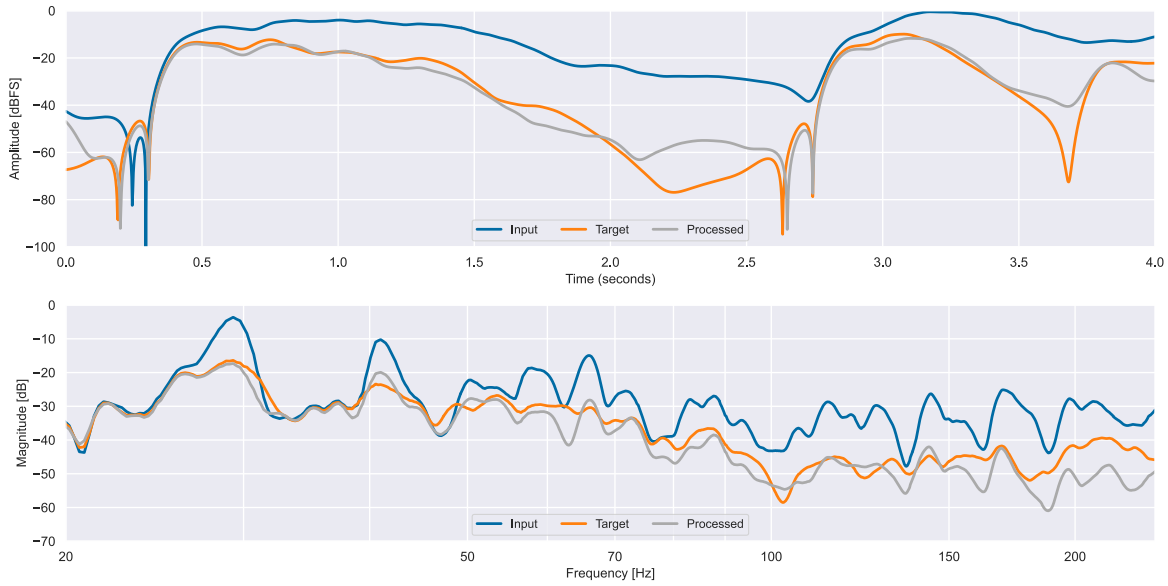


Fig. 5 – Room Volume 208 m^3 – Top: Time response (scaled in dBFS) of the Input (blue), Target (orange), and Processed (grey) envelopes.– Bottom: Frequency response of the Input (blue), Target (orange), and Processed (grey) signals.

driven approach to modal control, improving efficiency in room acoustics correction.

As shown in Fig. 3, Fig. 4 and Fig. 5, its performance remains consistent across the dataset, effectively adapting to different room volumes and reducing modal decay in accordance with the target response. The time domain correction is also noticeable in the frequency domain where spectral peaks attributable to room modes are reduced in magnitude and exhibit increased bandwidth.

Moreover, the envelopes of the processed white-noise signals, particularly the attack and decay segments, closely track those of the target, with only minor deviations.

Differences become apparent only at levels below −40 dBFS, where the processed output diverges slightly from the target. This behaviour arises from the use of RMSE as the loss criterion; the least-squares formulation minimises overall error and thus affords diminished weighting to very low-amplitude regions.

Consequently, while the overall spectral envelope is corrected, residual peaks persist at these frequencies. In contrast, for lower-order modal frequencies, the model effectively suppresses modal energy, leading to a smoother spectral response where the prominent peaks present in the input are no longer visible. This suggests that the model is particularly effective at low-frequency modal correction, while further refinements in the loss function could improve peak suppression in the higher-frequency range.

Although these peaks remain visible in the spectra, they may not be perceptually relevant, as room modes at these frequencies fall below the PMTs and the target spectral response.

Formal subjective listening tests may be required in future work to further investigate this aspect.

The generalization of the model to real-world signals, such as music and measured RIRs, remains an open research question. Since the model was trained exclusively on white noise bursts and synthesized RIRs, its performance on naturally occurring excitation signals is currently limited.

When applied to music or real RIRs, the model exhibits a notable reduction in accuracy, likely due to the spectral and temporal characteristics of the training data.

5 Conclusion

This study implemented a deep learning-based approach for low-frequency modal suppression in room acoustics using a modified Wave-U-Net architecture trained on synthetic RIRs.

Results show that the model effectively reduces modal decay times, aligning them with the target values as confirmed by the frequency-domain analysis, which indicates that low-order modal frequencies are successfully attenuated, resulting in a smoother spectral response compared to the input.

Higher-order modal frequencies exhibit residual spectral peaks, suggesting that the current model prioritizes overall spectral alignment but does not explicitly penalize localized deviations.
A refined loss function incorporating spectral smoothing constraints could improve spectral peak suppression while maintaining alignment with the target response.
However, their perceptual relevance remains uncertain and may require formal subjective listening tests for further assessment.
While the model performs as expected within the synthetic dataset, its generalization to real-world signals, such as music and measured RIRs, remains an open question.
Training was conducted using white noise bursts, ensuring uniform spectral excitation, but lacking the more idiosyncratic harmonic and temporal characteristics of musical signals.
When tested on non-synthetic signals, deviations in spectral alignment suggest that the model may not fully generalize to real-world excitation sources.
Future work will focus on retraining the model with musical excitation signals to improve its adaptability to real-world scenarios.
This transition introduces additional challenges due to the spectral sparsity, non-stationarity, and dynamic range variations of music, which differ significantly from white noise excitation. Data augmentation techniques and perceptually motivated loss functions will be investigated to enhance robustness.
Additionally, introducing causality into the model would improve its real-time applicability, making it more suitable for live audio processing and low-latency applications.

References

[1] A. Weisser and J. H. Rindel, "Evaluation of sound quality, boominess, and boxiness in small rooms.," *Journal of the Audio Engineering Society,* vol. 54, no. 6, pp. 495-511, 2006.

[2] F. A. Everest, Master Handbook of Acoustics, McGrawHill Education Tab, 2014.

[3] F. E. Toole, Sound Reproduction: The Acoustics and Psychoacoustics of Loudspeakers and Rooms, Routledge, 2017.

[4] T. Corbach, A. Von Dem Knesebeck, K. Dempwolf, M. Holters, P. Sorowka and U. Zölzer, "Automated Equalization For Room Resonance Suppression," in *Conference on Digital Audio Effects - DAFX*, Como, 2009.

[5] B. Owsinski , The Mixing Engineer's Handbook, Bobby Owsinski Media Group, 2022.

[6] M. Senior, Mixing Secrets for the Small Studio, Routledge, 2018.

[7] D. Stoller, S. Ewert and S. Dixon, "Wave-U-Net: A multi-scale neural network for end-to-end audio source separation.," in *Proceedings of the 19th International Society for Music Information Retrieval Conference*, Paris, France, 2018.

[8] C. Macartney and T. Weyde, "Improved Speech Enhancement with the Wave-U-Net. 2018," in *https://arxiv.org/abs/1811.11307*, arxiv.org, 2018.

[9] J. Donley and P. Calamia, "DARE-Net: Speech Dereverberation And Room Impulse Response Estimation," *Stanford University,* 2022.

[10] B. M. Fazenda, M. Stephenson and A. Goldberg, “Perceptual thresholds for the effects of room modes as a function of modal decay,” *The Journal of the Acoustical Society of America,* vol. 137, no. 3, pp. 1088-1098, 2015.

[11] H. Kuttruff, Room Acoustics, CRC Press, 2019.

[12] B. Fazenda, M. R. Avis and W. J. Davies, "Perception of Modal Distribution Metrics in Critical Listening Spaces-Dependence on Room Aspect Ratios," *Audio Eng. Soc,* vol. 53, no. 12, p. 1128–1141, 2005.

[13] E. Kharitonov, R. Morgane, S. Gabriel, W. Lior, P.-E. Mazaré, M. Douze and E. Dupoux, "Data Augmentation and Deep Learning Methods in Sound Classification: A Systematic Review," Shenzhen, China, 2020.

[14] J. H. Rindel, "Modal energy analysis of nearly rectangular rooms at low frequencies.," *Acta Acustica united with Acustica,* vol. 101, no. 6, pp. 1211-1221, 2015.

[15] J. Heddle, "Room acoustics for listening," in *2nd Australasian Acoustical Societies Conference*, 2016.

[16] B. Hanbin, P. Andreev, A. Saginbaev, N. Babaev, W.-J. Lee, H. Sung and H.-Y. Cho, "Speech Boosting: Low-Latency Live Speech Enhancement for TWS Earbuds," in *Interspeech*, Kos, Greece, 2004.

[17] M. T. Ho, J. Lee, B.-K. Lee, D. H. Yi and H.-G. Kang, "A Cross-Channel Attention-Based

Wave-U-Net for Multi-Channel Speech Enhancement," in *Interspeech*, Virtual conference, 2020.

[18] I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization," in *International Conference on Learning Representations*, New Orleans, LA, USA, 2019.

[19] B. Katz, Mastering Audio: The Art and the Science, Routledge, 2014.

[20] S. Cecchi, A. Carini and S. Spors, "Room Response Equalization—A Review," Applied Sciences, 2018.

[21] J. Pons, S. Pascual, G. Cengarle and J. Serrà, "Upsampling Artifacts In Neural Audio Synthesis," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Toronto, ON, Canada, 2021.

Audio Engineering Society

Conference Paper 27

Presented at the AES International Conference on Machine Learning and Artificial Intelligence for Audio
2025 September 8–10, London, UK

This paper was peer-reviewed as a complete manuscript for presentation at this conference. This paper is available in the AES E-Library (http://www.aes.org/e-lib), all rights reserved. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Journal of the Audio Engineering Society.

Perceiving AI in Music: Human Evaluation of AI-Generated Melodies and AI Detection Sensitivity

Michael Oehler[1], Jasper Oldach[1], and Florian Zwißler[2]

[1] *Osnabrück University, Music Technology and Digital Musicology Lab*
[2] *Cologne University of Music and Dance*

Correspondence should be addressed to Michael Oehler (michael.oehler@uos.de)

ABSTRACT

Recent advancements in generative artificial intelligence (AI) have led to increasingly sophisticated music composition systems, yet their perceived creative quality and detectability remain debated. This study examines human perception of AI-generated music across two paradigms: a melody continuation task, where AI- and human-generated melodies were rated on four aesthetic dimensions, and an AI detection task, where participants distinguished between AI-composed and human-composed songs. Results indicate that AI-generated melodies were consistently rated lower than human compositions, with no significant differences between AI models or the impact of expert feedback. In the detection task, participants identified AI-generated songs at a level above chance, though sensitivity remained low. Notably, musical sophistication, as measured by the Gold-MSI, did not significantly influence ratings or detection accuracy. These findings highlight persistent limitations in AI-generated music while raising questions about the perceptual boundaries between AI and human creativity.

1 Introduction

Recent advancements in generative artificial intelligence (AI) have significantly impacted creative domains, notably musical composition [1]. AI systems, including ChatGPT and Google Magenta, have demonstrated the ability to generate musical content; however, empirical evaluations of their creative capabilities relative to human composers remain sparse. Even recent comprehensive comparative evaluations of AI music-generation systems, such as Yin et al. [2], did not include general-purpose AI models like ChatGPT, focusing instead exclusively on specialized deep-learning-based music generation models. This gap underscores the necessity of systematically assessing general-purpose AI agents, particularly given their rapid development and broad availability.

Schreiber et al. recently conducted a seminal study assessing AI-generated melodies from ChatGPT and Google Magenta against human-created melodies by music students within a standardized melody continuation paradigm [3]. They found the AI compositions to be significantly inferior in aesthetic quality.

The present research builds on the methodology of [3] with updated AI agents, as the rapid evolution of AI technologies necessitates continuous empirical reassessment. Notably, Google Magenta has seen limited recent developments, whereas newer generative systems like Google Gemini and advanced reasoning

models from ChatGPT (e.g., GPT-4 and later variants) have emerged, offering potentially enhanced creative capabilities relevant for music generation.

1.1 Background and Related Work

A growing body of perceptual research has examined listeners' abilities to detect and evaluate AI-generated music. For instance, Frieler and Zaddach asked jazz experts to rate human-performed solos against algorithmically composed counterparts, finding a clear preference for human performances and identification accuracy around 65 % [4]. Extending this paradigm to a specialized sub-genre, Sarmento et al. showed that progressive-metal guitarists could reliably distinguish AI-generated guitar tablature from human-composed examples, again preferring the latter in subjective ratings [5]. Complementing these discrimination studies, White et al. and Shank et al. revealed a pronounced labeling bias: when identical excerpts are presented as "AI-generated," listeners rate them significantly lower in expressive and aesthetic quality than when labeled "human" [6, 7].

Parallel to these perceptual inquiries, the landscape of generative AI in music has bifurcated into symbolic-domain and audio-domain approaches. Symbolic systems—from early rule-based composition engines [8] to modern neural architectures such as PerformanceRNN [9] and MusicTransformer [10]—produce MIDI sequences that are subsequently rendered via synthesis engines or neural vocoders. By contrast, audio-domain models like Jukebox [11], MusicLM [12], MusicGen [13], MusicLDM [14], Mustango [15], Riffusion [16], and Suno AI [17] operate directly on waveform or spectrogram representations, often leveraging discrete audio tokens or diffusion in a learned latent space. Seed-Music has recently proposed a hybrid pipeline combining autoregressive language modeling with diffusion-based audio refinement, illustrating how high-level symbolic control and fine-grained timbral editing can be unified [18].

Comparative evaluations within the symbolic domain have largely focused on specialist architectures: Yin et al. systematically benchmarked models such as MusicTransformer and CoCoNet in a melody-continuation task [2]. More recently, Schreiber et al. extended this to general-purpose language models—ChatGPT and Google Magenta—finding that both lag behind human composers in aesthetic ratings, with ChatGPT's continuations rated marginally higher than those from Magenta [3].

Building on these insights, our study bridges the symbolic and audio domains. We first evaluate state-of-the-art autoregressive transformer models (GPT-4, Gemini) in a standardized melody-continuation paradigm. We then assess a prototypical audio-domain diffusion system (Suno AI), which synthesizes rich timbral textures via latent diffusion over spectrogram embeddings, in a signal-detection task to measure listeners' sensitivity to AI generation.

1.2 Aims and Hypotheses

This dual-paradigm evaluation addresses a critical gap by examining both symbolic and audio-based generative systems through the lens of human perception. Unlike prior technical detection approaches [19], our study emphasizes listeners' subjective experiences and qualitative judgments. Based on prior research on AI-generated music perception and human-AI comparisons in musical creativity, the present study formulates the following hypotheses:

H1: AI-generated melodies will receive significantly lower ratings than human-composed melodies across several aesthetic dimensions. This hypothesis aligns with previous studies showing that AI-composed music is often perceived as less expressive and coherent compared to human compositions [3].

H2: AI-generated melodies that incorporate expert feedback (reprompts) will receive higher ratings than those without feedback. Given that AI models are designed to optimize outputs based on iterative refinements, we expect that structured feedback from human experts will improve the perceived quality of AI compositions.

H3: Participants will identify AI-generated songs at a level above chance, indicating some level of sensitivity to distinguishing AI-composed from human-composed music. This expectation follows from previous research showing that listeners can differentiate between AI and human music, albeit with moderate accuracy [4, 7, 5].

H4: Musical sophistication will correlate with AI detection accuracy, such that individuals with higher musical sophistication will demonstrate greater sensitivity in distinguishing AI-generated from human-composed music.

H5: AI-generated songs containing vocals will be more easily identified as AI-generated compared to instrumental AI-generated music. This hypothesis is derived from research suggesting that listeners are particularly sensitive to unnatural renditions of human voice [20, 21, 22].

2 Methods

A listening experiment was conducted in two parts. In the first part, participants qualitatively evaluated stimuli generated using the melody continuation task. The second part utilized a Signal Detection Theory (SDT) paradigm [23, 24], in which listeners decided for each stimulus whether it was AI-generated or human-composed. Additionally, musical sophistication was assessed using the Gold-MSI [25] to examine whether participants' judgments were influenced by their level of musical sophistication.

2.1 Melody continuation task

Following [3], we selected a distinct seven-bar melodic fragment (3/4 meter, 80 bpm) as seed prompt. Eight graduate music students each produced one "human-composed" continuation of this fragment. The same students then prompted two general-purpose AI agents—GPT-4o and Gemini 2.0—to generate continuations under identical instructions, and finally they provided up to three iterative reprompts (expert feedback) to each AI output in order to refine it. Students provided feedback to the AI in the same manner as they would during a live co-composing session, offering guidance on musical shape rather than prescribing exact pitches. For instance, one might say: "Try strengthening the melody in bar 7 and resolve the phrase more harmonically at the end." Note that this feedback deliberately avoided specifying exact notes or sequences, focusing instead on the overall contour and expressive goals.

Unlike specialized music-generation systems (e.g., MuseNet, MusicLM, MusicGen, Seed-Music), which work on raw audio or opaque MIDI representations via hierarchical embeddings and diffusion/transformer architectures, both GPT-4o and Gemini accept symbolic pitch–duration sequences expressed in natural-language prompts and return continuations in the same format. This makes them ideally suited for a standardized, symbolic melody-continuation paradigm.

All continuations—whether human- or AI-generated—adhered to a common set of compositional guidelines: a film-music style, between 10 and 20 notes, varied note durations, a limited pitch range (D3–D5), and a clearly defined melodic climax. By comparing outputs with and without expert feedback, our factorial design yielded 40 stimuli in total:

- 8 human-composed stimuli (HUM)
- 8 GPT-4o stimuli without feedback (GPT_nF)
- 8 GPT-4o stimuli with feedback (GPT_wF)
- 8 Gemini stimuli without feedback (GEM_nF)
- 8 Gemini stimuli with feedback (GEM_wF)

All continuations were encoded as MusicXML, rendered using the oboe timbre from the MuseScore SF3 soundfont, and normalized for loudness to ensure consistency.

In the subsequent listening study, participants rated the resulting melodies on several aesthetic dimensions, specifically focusing on how musically logical and meaningful, interesting, convincing, and personally likable they were, using 5-point Likert scales.

2.2 AI Detection Task

This part of the study follows a more exploratory approach, as it is challenging to balance the use of commercially popular AI music generation systems with the need for strict control over generation parameters. Many widely used AI agents offer limited customization options, making it difficult to systematically manipulate and standardize key musical aspects across different conditions. At the same time, highly controllable research-based AI models often fail to match the naturalness and production quality of commercial systems. Given these constraints, we opted for a commercial AI platform that represents the current state-of-the-art in AI-generated music [26], ensuring the highest level of realism and fidelity while allowing us to study the impact of such technology in real-world usage scenarios.

Eight music students were tasked with using Suno's AI music generation platform [17] to create three musical pieces in a genre/style of their choice, aiming for a particularly natural sound. The specific requirement was to produce one piece featuring male vocals,

one with female vocals, and one instrumental. This approach stems from the understanding that listeners are particularly sensitive to unnatural renditions of human voice, as indicated by several studies [20, 21, 22]. Suno was selected for this task due to its recent Version 4 release, which has been noted for producing more natural-sounding music, especially in vocal performances.

To maximize the perceived naturalness of the AI-generated pieces, music students were allowed to refine their compositions iteratively, adjusting generation parameters until they closely matched their artistic expectations. Additionally, participants were asked to select lesser-known, non-AI-composed songs that closely matched their AI-generated pieces in terms of genre/style, instrumentation etc. This selection aimed to facilitate a comparative analysis between AI-generated and human-composed music.

From the total of 48 musical pieces, comparable 30-second excerpts were extracted, containing characteristic elements such as verse and chorus sections. All excerpts were loudness-normalized to ensure consistent listening conditions.

2.3 Participants

An a priori sample size calculation was conducted using G*Power [27] to ensure sufficient statistical power for the planned repeated-measures ANOVA. While Schreiber et al. [3] found a large effect size in their study, we opted for a more conservative approach due to advancements in AI agents. Specifically, we based our calculation on a small effect size (f = 0.2) to account for potential differences in perceptual sensitivity. With a power of 0.95 and a significance level of 0.05, this resulted in a required sample size of 48 participants. To compensate for potential exclusions due to incomplete data, we recruited 60 participants. After data screening, two participants had to be excluded due to missing or incomplete responses, leaving a final sample of 58 participants for analysis.

The final sample consisted of 37 female, 20 male, and 1 non-binary participant, with a mean age of 28.93 years (SD = 14.68). To assess participants' overall level of musical sophistication, the general factor of the Gold-MSI was measured, yielding an average score of 77.5, with values ranging from 33 to 112.

Written informed consent was obtained from all participants prior to the experiment. Participants were not paid an honorarium but could withdraw at any time without penalty. All procedures were conducted in accordance with the Declaration of Helsinki and the guidelines of the German Psychological Society.

2.4 Procedure

Due to the relatively large number of stimuli used in both parts of the study, an incomplete within-subjects design was implemented. Each participant was randomly assigned stimuli produced by one of the eight music students, evaluating five melodies and six musical pieces via an online study conducted using SoSciSurvey [28]. This approach ensured direct comparability between AI-generated and human-composed stimuli while maintaining a manageable workload for participants. Consequently, stimuli created by each music student were evaluated by approximately seven to eight participants.

Each participant first evaluated the melodies, presented in random order, by rating them on four dimensions—musically logical and meaningful, interesting, convincing, and personal liking—using a 5-point Likert scale. Subsequently, the six musical pieces were also presented in randomized order. After listening to each piece, participants indicated whether they believed it was composed by AI or a human composer. Additionally, they reported whether they recognized the song; trials involving recognized songs were excluded from further analysis.

At the end of the experiment, participants provided demographic information, and the general factor of the Gold-MSI was assessed to measure their overall level of musical sophistication.

3 Results

3.1 Melody continuation task

3.1.1 Analysis of Variance (ANOVA)

Before each repeated-measures ANOVA, we assessed the assumption of sphericity using Mauchly's test. Results were as follows: for interesting, $W(9) = 0.754$, $p = 0.082$; for logical and meaningful, $W(9) = 0.850$, $p = 0.454$; for convincing, $W(9) = 0.948$, $p = 0.968$; and for liking, $W(9) = 0.934$, $p = 0.930$. In all cases,

$p \geq 0.05$, indicating that the sphericity assumption held and that uncorrected degrees of freedom could be used.

A series of repeated-measures ANOVAs were then conducted to examine whether the ratings of musical stimuli differed across the five conditions. The results revealed significant main effects of category for all four dependent variables (interesting, logical and meaningful, convincing, and liking). The effect sizes (Partial Eta-Squared) ranged from 0.098 to 0.148, indicating small to moderate effects (see Table 1).

For interesting, a significant effect of category was found, $F(4,224) = 8.85, p < 0.001, \eta_p^2 = 0.1365$, suggesting that the perceived interestingness of the melodies differed significantly between groups. Similarly, significant main effects were observed for logical and meaningful ($F(4,224) = 6.09, p < 0.001, \eta_p^2 = 0.0981$), convincing ($F(4,224) = 6.89, p < 0.001, \eta_p^2 = 0.1096$), and liking ($F(4,224) = 9.76, p < 0.001, \eta_p^2 = 0.1485$), indicating systematic differences in how the musical continuations were evaluated across conditions.

Table 1: Repeated Measures ANOVA Results

DV	*F(4, 224)*	*p*	η_p^2
Interesting	8.85	< 0.001	0.1365
Logical	6.09	< 0.001	0.0981
Convincing	6.89	< 0.001	0.1096
Liking	9.76	< 0.001	0.1485

3.1.2 Multiple Regression with Musical Sophistication

To examine whether musical sophistication (Gold-MSI) influenced ratings, multiple regression analyses were conducted with `category`, `Gold-MSI`, and their interaction as predictors. The models revealed relatively low explanatory power (R^2 ranging from 0.087 to 0.124), suggesting that only a small portion of the variance in ratings could be attributed to these factors. Importantly, `Gold-MSI` was not a significant predictor in any model ($p > 0.1$), indicating that individual musical sophistication did not systematically influence the evaluations of the melodic continuations. Moreover, no significant interactions between `Gold-MSI` and `category` were observed, suggesting that the effect of musical sophistication did not vary across AI- and human-generated melodies.

3.1.3 Post-hoc Comparisons

Bonferroni-corrected post-hoc paired t-tests were conducted to further explore differences between the conditions. Across all four dependent variables, a consistent pattern emerged: significant differences were found between AI-generated melodies (GPT and GEM) and human-generated melodies (HUM), while AI models did not significantly differ from each other. These results are illustrated in Figure 1 and detailed in Table 2, which also report Cohen's *d* as a measure of effect size.

Table 2: Post-hoc Comparisons Across Dimensions

	t	*p*	*d*
Interesting			
GPT_nF vs. HUM	-5.04	< .001	-0.67
GPT_wF vs. HUM	-2.93	.049	-0.39
GEM_nF vs. HUM	-4.22	< .001	-0.56
GEM_wF vs. HUM	-4.93	< .001	-0.65
Logical and Meaningful			
GPT_nF vs. HUM	-4.16	.001	-0.55
GPT_wF vs. HUM	-3.72	.004	-0.49
GEM_wF vs. HUM	-4.67	< .001	-0.62
Convincing			
GPT_nF vs. HUM	-3.10	.030	-0.41
GPT_wF vs. HUM	-3.48	.009	-0.46
GEM_nF vs. HUM	-3.20	.022	-0.42
GEM_wF vs. HUM	-4.87	< .001	-0.65
Liking			
GPT_nF vs. HUM	-5.10	< .001	-0.68
GPT_wF vs. HUM	-4.14	.001	-0.55
GEM_nF vs. HUM	-4.97	< .001	-0.66
GEM_wF vs. HUM	-5.01	< .001	-0.66

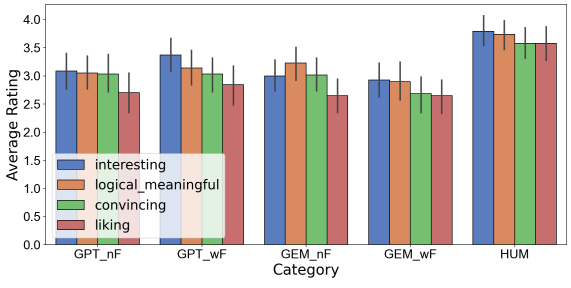


Fig. 1: Comparison of ratings across categories and dependent variables. The rating scale ranged from 1 (not at all) to 5 (very much). Error bars represent 95% confidence intervals.

3.2 AI Detection Task

For each participant, a *d'* value was calculated based on the Signal Detection Theory (SDT) paradigm [23, 24] to measure their sensitivity in distinguishing AI-generated from human-composed musical pieces (*d'* = Z(hit rate) - Z(false alarm rate)). The average *d'* value across all participants was $M = 0.711$ ($SD = 1.212$), indicating that participants were able to identify Suno-generated songs at a level above chance. However, the sensitivity was low, suggesting that differentiating AI-generated from human-composed music remained challenging.

To investigate whether musical sophistication, as measured by the Gold-MSI, is associated with the ability to distinguish AI-generated from human-composed music, several analyses were conducted.

A median split on the Gold-MSI scores divided participants into high and low musical sophistication groups. A two-sample *t*-test comparing their sensitivity (*d'*) values revealed no significant difference between groups, $t(56) = 0.471$, $p = .640$, Cohen's $d = 0.124$. The average *d'* value for the high musical sophistication group was $M = 0.787$ ($SD = 1.354$), while the low musical sophistication group had an average *d'* of $M = 0.636$ ($SD = 1.069$).

Figure 2 shows a Kernel Density Estimation (KDE) plot illustrating the distribution of *d'* values for both high and low musical sophistication groups. The left-skewed distribution observed in the low Gold-MSI group suggests that while most participants performed similarly to the high Gold-MSI group, a subset exhibited notably lower sensitivity (*d'*), indicating greater individual variability in distinguishing between AI-generated and human-composed music.

Additionally, a Pearson correlation analysis between Gold-MSI scores and *d'* values yielded $r = 0.119$, $p = .375$, indicating no significant linear relationship. This further supports the notion that higher musical sophistication does not necessarily correspond to greater sensitivity in identifying AI-generated music.

To examine whether sensitivity (*d'*) in distinguishing AI-generated from human-composed music varied across the three song variants (male vocals, female vocals, and instrumental), a series of chi-square tests were conducted. The analysis compared the detection rates of AI-generated (Suno) and human-composed (HUM) songs across these categories, with Cramér's *V* calculated as a measure of effect size.

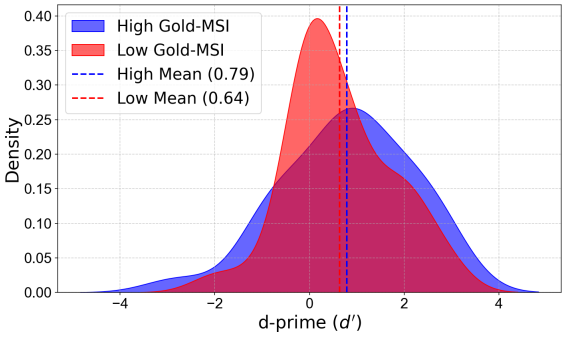


Fig. 2: Kernel Density Estimation (KDE) plot for *d'* by Gold-MSI group.

The results indicated that none of the comparisons reached statistical significance (all *p*-values > .05), suggesting participants were not systematically more or less accurate in identifying AI-generated music based on the presence or type of vocals. Across all conditions, effect sizes remained small (Cramér's $V < 0.3$), further reinforcing the conclusion that the presence of male vocals, female vocals, or the absence of vocals did not substantially impact participants' ability to distinguish AI-generated from human-composed songs.

4 Discussion

The present study aimed to assess the perceived quality and detectability of AI-generated music compared to human-composed music across two experimental paradigms: a melody continuation task and an AI detection task.

4.1 Evaluation of AI-Generated Melodies

The results of the melody continuation task indicate that AI-generated melodies were consistently rated lower than human-composed melodies across all four aesthetic dimensions (interesting, logical and meaningful, convincing, and liking). This pattern replicates previous findings by Schreiber et al. [3], reinforcing the notion that, despite recent advancements, AI-generated melodies still exhibit qualitative shortcomings when compared to human compositions.

Importantly, the presence of iterative expert feedback during AI generation (GPT_wF, GEM_wF) did not

lead to significantly higher ratings compared to their respective no-feedback conditions (GPT_nF, GEM_nF). This suggests that while AI systems can integrate feedback in text-based tasks, their ability to refine creative outputs in music generation may be more limited. Unlike human composers, who can abstract and creatively reinterpret feedback, AI systems may struggle to make meaningful structural and stylistic improvements beyond surface-level modifications.

Furthermore, post-hoc comparisons revealed that while significant differences existed between AI-generated and human-composed melodies, no significant differences were found between the different AI models or between their with- and without-feedback conditions. This suggests that the type of AI system (GPT vs. GEM) had little impact on perceived quality, and that feedback did not enhance compositional effectiveness in the present study.

Musical sophistication, as measured by the Gold-MSI, did not significantly influence ratings of AI- or human-generated melodies, nor did it moderate the effect of category. Consistent with the findings of [3], this suggests that both musically trained and untrained listeners evaluate AI-generated music similarly, reflecting a shared perception of its aesthetic limitations.

4.2 Detection of AI-Generated Music

The results from the AI detection task revealed that participants were able to distinguish AI-generated from human-composed songs at a level above chance ($d' = 0.711$), yet their sensitivity remained relatively low. This aligns with prior studies showing that while AI-generated music can be recognized with some accuracy, it often approximates human-like compositional characteristics closely enough to make identification difficult [4, 7, 5].

No significant differences were found between the ability to detect AI-generated music based on whether the song contained male vocals, female vocals, or no vocals. This finding contradicts the expectation that participants would be particularly sensitive to unnatural vocal rendering, as suggested by previous research on voice perception [20, 21, 22]. While it remains possible that AI-generated vocals introduce subtle unnatural characteristics, these were not strong enough to consistently impact detection accuracy.

Furthermore, musical sophistication did not significantly correlate with sensitivity in AI detection, and no differences were found between participants with high and low Gold-MSI scores. This suggests that experience and expertise in music do not necessarily enhance the ability to identify AI-generated music, potentially because the distinctions between AI and human compositions are not solely based on musical complexity but also on production quality and stylistic coherence.

5 Conclusions

While this study provides insights into the perception and detection of AI-generated music, several limitations should be acknowledged.

First, the study focused on a relatively small set of AI-generated and human-composed melodies and songs, limiting the generalizability of the findings. Although the stimuli were carefully selected and controlled, future studies should expand the dataset to include a broader range of musical styles, complexity levels, and AI generation techniques to assess whether the observed effects hold across different musical contexts.

Second, the AI models used in the melody continuation task (GPT-4o and Gemini 2.0) were not originally designed for music composition, but rather adapted for this purpose. While these models have shown promise in generating musical structures, their outputs may not fully reflect the capabilities of dedicated music AI systems such as ByteDance's Seed-Music, OpenAI's MuseNet, Google's MusicLM, or Suno's commercial platform. A direct comparison between general-purpose AI models and music-specific AI systems could provide a more comprehensive understanding of their respective strengths and weaknesses.

Third, the effectiveness of expert feedback on AI-generated melodies may have been constrained by the structure of the feedback process itself. Music students provided iterative feedback to AI models, but the way in which AI systems processed and incorporated this feedback remains unclear. It is possible that the current generation of AI models lacks the capacity to meaningfully refine compositions in response to nuanced musical critique. A detailed log of reprompt messages, coupled with an automated evaluation of melodic features (e.g., comparing pitch histograms or harmonic transitions before vs. after feedback), would help quantify the actual impact of expert guidance. Future studies could also explore alternative interaction paradigms,

such as real-time co-creative systems where AI and human composers iteratively develop music together, rather than AI simply responding to written prompts.

Fourth, in the AI detection task, only one commercial AI system (Suno) was tested. While Suno represents a state-of-the-art model in AI-generated music [26], the extent to which the findings generalize to other AI music generators remains uncertain. Different AI systems may have distinct sonic characteristics that influence detectability, and the use of a single system limits the scope of conclusions about AI music detection in general.

Finally, the study employed an incomplete within-subjects design, meaning that each participant evaluated only a subset of the available stimuli. While this approach was necessary to manage cognitive load and ensure comparability, it may have introduced variability in participants' exposure to different compositional styles. Future research could explore fully within-subjects designs or counterbalanced approaches to enhance the robustness of the findings.

Addressing these limitations in future research will be essential for developing a more comprehensive understanding of AI-generated music, its creative potential, and how human listeners perceive and interact with it.

References

[1] Mitra, R. and Zualkernan, I., "Music Generation Using Deep Learning and Generative AI: A Systematic Review," *IEEE Access*, 2025.

[2] Yin, Z., Reuben, F., Stepney, S., and Collins, T., "Deep learning's shallow gains: a comparative evaluation of algorithms for automatic music generation," *Machine Learning*, 112(5), pp. 1785–1822, 2023.

[3] Schreiber, A., Sander, K., Kopiez, R., and Thöne, R., "The creative performance of the AI agents ChatGPT and Google Magenta compared to human-based solutions in a standardized melody continuation task," *Jahrbuch Musikpsychologie*, 32, p. e195, 2024, doi:10.5964/jbdgm.195.

[4] Frieler, K. and Zaddach, W.-G., "Evaluating an analysis-by-synthesis model for Jazz improvisation," *Transactions of the International Society for Music Information Retrieval*, 5(1), pp. 20–34, 2022.

[5] Sarmento, P., Loth, J., and Barthet, M., "Between the AI and Me: Analysing Listeners' Perspectives on AI-and Human-Composed Progressive Metal Music," *arXiv preprint arXiv:2407.21615*, 2024.

[6] White, C. W., Kapoor, K., Cosme-Clifford, N., Symons, J., and von Mutius, L., "Humans Perceive Ai-Generated Music as Less Expressive than Comparable Human-Made Content," *Available at SSRN 5087035*, 2025.

[7] Shank, D. B., Stefanik, C., Stuhlsatz, C., Kacirek, K., and Belfi, A. M., "AI composer bias: Listeners like music less when they think it was composed by an AI," *Journal of Experimental Psychology: Applied*, 29(3), pp. 676–692, 2023, doi:10.1037/xap0000447.

[8] Cope, D., "Experiments in musical intelligence (EMI): Non-linear linguistic-based composition," *Journal of New Music Research*, 18(1-2), pp. 117–139, 1989.

[9] Simon, I. and Oore, S., "Performance RNN: Generating Music with Expressive Timing and Dynamics," `https://magenta.tensorflow.org/performance-rnn`, 2017.

[10] Huang, C.-Z. A., Vaswani, A., Uszkoreit, J., Simon, I., Hawthorne, C., Shazeer, N., Dai, A. M., Hoffman, M. D., Dinculescu, M., and Eck, D., "Music Transformer: Generating Music with Long-Term Structure," in *7th International Conference on Learning Representations (ICLR)*, 2019.

[11] Dhariwal, P., Jun, H., Payne, C., et al., "Jukebox: A Generative Model for Music," 2020, arXiv preprint.

[12] Agostinelli, A. et al., "MusicLM: Generating Music From Text," 2023, arXiv preprint.

[13] Copet, J., Kreuk, F., Gat, I., Remez, T., Kant, D., Synnaeve, G., Adi, Y., and Défossez, A., "Simple and controllable music generation," *Advances in Neural Information Processing Systems*, 36, pp. 47704–47720, 2023.

[14] Chen, K., Wu, Y., Liu, H., Nezhurina, M., Berg-Kirkpatrick, T., and Dubnov, S., "Musicldm: Enhancing novelty in text-to-music generation using beat-synchronous mixup strategies," in *ICASSP 2024*, pp. 1206–1210, IEEE, 2024.

[15] Melechovsky, J., Guo, Z., Ghosal, D., Majumder, N., Herremans, D., and Poria, S., "Mustango: Toward controllable text-to-music generation," *arXiv preprint arXiv:2311.08355*, 2023.

[16] Riffusion, "Riffusion AI," 2023, accessed: 2025-03-13.

[17] Suno AI, "Suno AI Audio Generation," 2023, accessed: 2025-03-13.

[18] Bai, Y., Chen, H., Chen, J., Chen, Z., Deng, Y., Dong, X., Hantrakul, L., Hao, W., Huang, Q., Huang, Z., et al., "Seed-music: A unified framework for high quality and controlled music generation," *arXiv preprint arXiv:2409.09214*, 2024.

[19] Afchar, D., Meseguer-Brocal, G., and Hennequin, R., "AI-Generated Music Detection and its Challenges," *arXiv preprint arXiv:2501.10111*, 2025.

[20] Nussbaum, C., Frühholz, S., and Schweinberger, S. R., "Understanding voice naturalness," *Trends in Cognitive Sciences*, 2025.

[21] Pandey, A., Edlund, J., Le Maguer, S., and Harte, N., "Listener sensitivity to deviating obstruents in WaveNet," *Proc. Interspeech. Dublin, Ireland*, pp. 1080–1084, 2023.

[22] Kühne, K., Fischer, M. H., and Zhou, Y., "The human takes it all: Humanlike synthesized voices are perceived as less eerie and more likable. evidence from a subjective ratings study," *Frontiers in neurorobotics*, 14, p. 593732, 2020.

[23] Thurstone, L. L., "A law of comparative judgment," *Psychological review*, 34(4), pp. 27–286, 1927.

[24] Green, D. M., Swets, J. A., et al., *Signal detection theory and psychophysics*, volume 1, Wiley New York, 1966.

[25] Müllensiefen, D., Gingras, B., Musil, J., and Stewart, L., "The Musicality of Non-Musicians: An Index for Assessing Musical Sophistication in the General Population," *PLoS ONE*, 9(2), p. e89642, 2014.

[26] Grötschla, F., Solak, A., Lanzendörfer, L. A., and Wattenhofer, R., "Benchmarking Music Generation Models and Metrics via Human Preference Studies," in *ICASSP 2025*, pp. 1–5, IEEE, 2025.

[27] Faul, F., Erdfelder, E., Lang, A.-G., and Buchner, A., "G*Power 3: A flexible statistical power analysis program for the social, behavioral, and biomedical sciences," *Behavior Research Methods*, 39(2), pp. 175–191, 2007, doi:10.3758/BF03193146.

[28] Leiner, D. J., "SoSci Survey: The Solution for Online Surveys," `https://www.soscisurvey.de`, 2025, accessed: 2025-03-13.

Audio Engineering Society

Conference Paper 28

Presented at the AES International Conference on Machine Learning and Artificial Intelligence for Audio
2025 September 8–10, London, UK

This paper was peer-reviewed as a complete manuscript for presentation at this conference. This paper is available in the AES E-Library (http://www.aes.org/e-lib), all rights reserved. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Journal of the Audio Engineering Society.

Faust Autodiff: Towards Audio Domain-Specific Machine Learning

Thomas Rushton[1], Yann Orlarey[1], Romain Michon[1], Tanguy Risset[1], and Stéphane Letz[2]

[1]*Inria, INSA Lyon, CITI, EA3720 , 69621 Villeurbanne, France*
[2]*Grame CNCM, CITI, EA3720 , 69621 Villeurbanne, France*

Correspondence should be addressed to Thomas Rushton (thomas.rushton@inria.fr)

ABSTRACT

Differentiable programming via automatic differentiation (AD) is the foundation for gradient-based optimisation techniques, and forms the basis for many current approaches to machine learning. Though well catered-for in general-purpose programming languages, the availability of AD in a domain-specific language (DSL) could offer novel perspectives on optimisation problems in the field of audio. We present a general scheme for differentiable programming in the Faust programming language, a high-performance DSL tailored to audio synthesis and signal processing. Faust's rich ecosystem, coupled with a comprehensive AD implementation, can provide support for audio optimisation and machine learning applications on a multitude of platforms, from FPGAs to the web.

1 Introduction

Gradient-based methods are ubiquitous in tackling a wide variety of mathematical optimisation and machine learning problems, and the availability of such methods hinges on the possibility of computing the derivatives of computer programs. In the field of machine learning in particular, automatic differentiation (AD) [1] has emerged as the dominant differentiable programming paradigm. In AD, the primitive operations of a programming language are defined differentiably, thus programs composed from those operations are in turn differentiable end-to-end with respect to their input variables. In a gradient-based optimisation scenario, the partial derivatives of an algorithm terminating in a loss function constitute a gradient vector, which can be backpropagated, causing the algorithm's input variables, weights in a neural network perhaps, to approach optimal values. Techniques of this sort employed in the audio domain are referred to as *differentiable digital signal processing* (DDSP) [2], and find applications from least mean squares (LMS) filtering [3] to differentiable audio synthesis [4].

Building on previous work on AD in the Faust programming language [5], here we exploit Faust's pattern matching abstraction to extend to a more general AD syntax, demonstrating that arbitrary compositions of signal processing blocks can be differentiated, and to higher order if desired. With AD in Faust, it is possible to create differentiable infinite impulse response (IIR) filters and neural network building blocks. Our work provides the basis for an audio-focused ecosystem for the development of solutions to problems based on machine learning.

2 The Faust Programming Language

Faust is a functional paradigm, domain specific language for audio digital signal processing [6]. It is built atop a domain-agnostic *block diagram algebra* [7, 8] that describes primitive operations connected into graphs via a selection of *composition operators* (see Figure 2). A textual language, Faust can, for example, define a diagram consisting of two blocks, A and B, composed in sequence, with code of the form

```
process = A : B;
```

Lst. 1: Sequential composition of two Faust blocks.

Conceptually, a given block diagram, D, may undergo semantic transformation, notated $[\![\cdot]\!]$, into a signal processor [8]:[1]

$$\begin{gathered} [\![D]\!] \,:\, \mathbb{S}^n \to \mathbb{S}^m \\ [\![D]\!](s_1,\ldots,s_n) \to (y_1,\ldots,y_m) \\ \mathbf{y}(t) = f(\mathbf{s}(t)) \,, \end{gathered} \tag{1}$$

where $\mathbb{S}$ is the set of all signals, f is a mathematical function analogous to the primitive operations and compositions of D, and t is time. The signal processor receives a vector of input signals, $\mathbf{s}$, and produces a vector of output signals, $\mathbf{y}$.

2.1 The Faust Ecosystem

Faust is a highly optimised, compiled language [9], with C++ as its default compilation target. One of the great strengths of the Faust ecosystem is its *write once, run anywhere* model; a variety of additional backends are supported, including LLVM [10], WebAssembly [11], and Rust [12], plus numerous software and hardware platforms. Faust code can be compiled for use in VST plugins [13], for embedded systems [14] and FPGAs [15]; it is used by artists and digital luthiers [16, 17, 18], and features sophisticated libraries for wave digital filter modelling [19] and physical modelling synthesis [20, 21], amongst others.

[1] The colon symbol on the first line of Equation (1) introduces the *type* of the signal processor $[\![D]\!]$ and is not to be confused with Faust's sequential composition operator.

2.2 Differentiable Programming in Faust

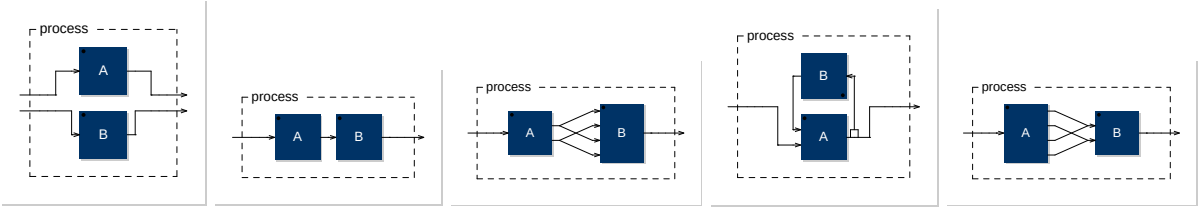
In light of Faust's versatility, efficiency, and strengths as a DSL in tackling audio problems it may come as little surprise that it has been turned to machine learning problems in the audio domain, via *DawDreamer* [22], which uses a Python backend to transpile Faust code for gradient-based optimisation with the JAX Python library. JAX, and other well-known machine learning frameworks, such as TensorFlow and PyTorch, enable gradient-based optimisation approaches by way of *automatic differentiation*, a technique for computing the numerical outputs of both a mathematical expression (or, more commonly, a computer program) and its derivatives in parallel [23, 1, 24].

Differentiable programming [25] is, broadly speaking, the creation of computer programs using primitive operations which are themselves differentiable; a computer program consisting of differentiable primitives may be differentiated automatically, producing the desired *primal* and *gradient* numerical outputs for use in a gradient-based optimisation algorithm. In this vein, a Faust-based LMS algorithm has been used for real-time acoustic control [3], though the DDSP aspect of this work was limited to the particular application. The feasibility of a more general approach to DDSP in the Faust language has been demonstrated via the implementation of forward-mode AD primitives [5].

One way of framing forward-mode AD is as an arithmetic of dual numbers [23, 26] of the form $\langle u, \nabla u \rangle$, where u is the aforementioned primal, and $\nabla u = \begin{bmatrix} \frac{\partial u}{\partial x_1} & \cdots & \frac{\partial u}{\partial x_N} \end{bmatrix}^T$, the gradient of u, its partial derivatives with respect to a vector of N input variables, $\mathbf{x}$. A dual number can be passed into a forward-mode differentiable primitive, f, yielding an output dual number, $\langle f(u), \nabla u f'(u) \rangle$. Faust is a language of *signals* and signal processors; consequently we will refer to the *dual signal*, $\langle s, \nabla s \rangle \in \mathbb{S}_d$, where $\mathbb{S}_d = \mathbb{S}^{N+1}$. The signal processor based on the differentiated block diagram D has the following semantic scheme:

$$\begin{gathered} \Delta_N[\![D]\!] \,:\, \mathbb{S}_d^M \to \mathbb{S}_d^P \\ \Delta_N[\![D]\!](\langle s_1, \nabla s_1 \rangle, \ldots, \langle s_M, \nabla s_M \rangle) \\ \to (\langle y_1, \nabla y_1 \rangle, \ldots, \langle y_P, \nabla y_P \rangle) \\ \langle \mathbf{y}(t),\, \mathbf{J}_\mathbf{y}(t) \rangle = \langle f(\mathbf{s}(t)),\, \mathbf{J}_f(t)\mathbf{J}_\mathbf{s}(t) \rangle \,, \end{gathered} \tag{2}$$

where $\Delta_N[\![\cdot]\!]$ indicates derivative transformation with respect to N input variables, and $\mathbf{J}_\mathbf{y}$, $\mathbf{J}_f = \nabla f(\mathbf{s})$, and



Parallel, A, B. Sequential, $A : B$. Split, $A <: B$. Recursive, $A \sim B$. Merge, $A :> B$.

Fig. 2: Example compositions of Faust blocks.

$\mathbf{J_s} = \nabla \mathbf{s}$ are, respectively, $P \times N$, $P \times M$ and $M \times N$ Jacobian matrices.

AD primitives have been demonstrated in Faust using the language's pattern matching abstraction [5]. Pattern matching is a source code transformation mechanism, and exploits the fact that a Faust function may receive, as a parameter, a programmatic expression. For example, a differentiable addition primitive, acting on dual signals $\langle s_1, \nabla s_1 \rangle, \langle s_2, \nabla s_2 \rangle$, may be created by defining a function that accepts the addition operator as a parameter, and produces, in parallel, the addition primitive's primal output, $s_1 + s_2$, and its gradients, $\nabla s_1 + \nabla s_2$. A

```
env(vars) = environment {
  // ...
  diff(+) = interleave(vars.N + 1, 2)
    : +,par(i, vars.N, +);
  // ...
};
```

Lst. 2: Definition of a differentiable addition primitive in the Faust language.

dual-signal addition operator implementation is given in Listing 2; the operator is defined within a *differentiable environment*, which receives and enumerates a vector of input variables, ensuring that all derivative transformations produce the appropriate number of partial derivatives; the function `interleave` delivers the primal signals and their respective partial derivatives to the addition operations in the appropriate order: $s_1, s_2, \frac{\partial s_1}{\partial x_1}, \frac{\partial s_2}{\partial x_1}, \ldots, \frac{\partial s_1}{\partial x_N}, \frac{\partial s_2}{\partial x_N}$, as illustrated in Figure 3. One addition primitive is created for the primal operation, and a further N additions in parallel, which we can notate $\left\|_{l=1}^{N} +\right.$, for computing partial derivatives.

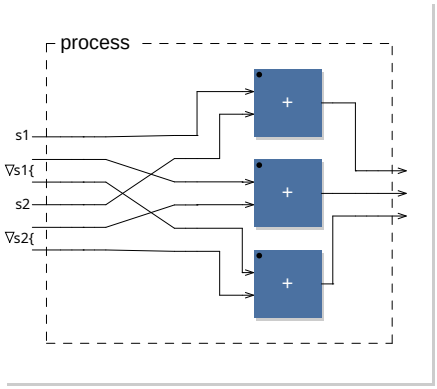


Fig. 3: Differentiable addition primitive applied to dual signals $\langle s_1, \nabla s_1 \rangle, \langle s_2, \nabla s_2 \rangle$; implicit input variable vector $\mathbf{x}$ of length $N = 2$.

2.3 Motivation

Considering the plethora of existing approaches to DDSP [2], why pursue an implementation in Faust? The availability of the Faust-to-JAX transpiler would appear to obviate such an initiative, but for the fact that the AD transformations facilitated by JAX lie the wrong side of the transpilation step. A comprehensive differentiable programming framework in the Faust language itself would allow DDSP algorithms to be written once, and run anywhere. A domain-specific approach to audio-oriented machine learning is not without precedent; indeed, a suite of ML-centric tools, with an emphasis on human computer interaction, exists for the ChucK programming language [27]. Faust, however, with its functional approach, and cross-platform capabilities, offers a unique point of vantage.

The observation has been made that the relative structures of neural networks and digital filters bear striking similarities [28]. There is a very real sense in which

machine learning problems, and gradient-based optimisation problems in general, are signal processing problems. The possibility of turning a signal processing DSL to problems of this sort paves the way to novel approaches and fresh perspectives on this dynamic field of research.

3 A General Scheme for Differentiating Faust Programs

In Faust's block diagram algebra, a given block diagram, D, is comprised of a primitive block, P, or the composition of two block diagrams via a composition operator according to the syntactic rule

$$D = P \mid (D_1 \diamond D_2) \,, \tag{13}$$

where $\diamond$ is one of the sequential (:), parallel (,), split (<:), merge (:>), or recursive ($\sim$) composition operators. The creation of differentiable Faust programs has been shown to be possible by *undifferentiated* composition of differentiated primitive blocks [5], the syntactic rule being

$$\Delta_N[\![D]\!] = \Delta_N[\![P_1]\!] \diamond \Delta_N[\![P_2]\!] \diamond \cdots \diamond \Delta_N[\![P_n]\!] \,. \tag{14}$$

While effective in facilitating differentiable programming, this syntax imposed an undesirable verbosity. Our preferred syntactic rule should be a direct transformation of Equation (13):

$$\Delta_N[\![D]\!] = \Delta_N[\![P]\!] \mid \Delta_N[\![D_1 \diamond D_2]\!] \,. \tag{15}$$

The ideal outcome is to permit the programmer to subject an entire Faust program to automatic differentiation in a single expression, minimising the amount of re-writing required to ready their code for transformation. This would render a great proportion of existing Faust code differentiable, and, in doing so, prepare that wealth of material for prospective use in optimisation problems, and machine learning applications. In order to achieve such an outcome, it is necessary to provide differentiation rules for the five composition operators, taking into account their arithmetical significance, if any, in a block diagram to which they contribute.

3.1 Differentiable Block-Diagram Compositions

To define syntactic rules for the derivatives of Faust's composition operators entails analysing them as if they were themselves primitive blocks. To do this, we must consider them in tandem with rules for some of Faust's fundamental primitive operators; these will assist us in achieving differentiable compositions.

3.1.1 Identity Primitive

Faust's identity function, notated _, describes a connection between two blocks. Treated as a signal processor, it adheres to the following semantic scheme [8]:

$$\begin{aligned} [\![_]\!] &: \mathbb{S} \to \mathbb{S} \\ [\![_]\!](s) &\to (s) \,. \end{aligned} \tag{16}$$

It receives one signal and returns that same signal unchanged, analogous to multiplying it by a constant signal of value 1. In dual signal arithmetic, this is

$$\langle s, \nabla s \rangle \langle 1, \nabla 1 \rangle = \langle 1s, 1\nabla s + s\nabla 1 \rangle = \langle s, \nabla s \rangle \,, \tag{17}$$

and the semantic rule for the derivative of the identity function is thus

$$\begin{aligned} \Delta_N[\![_]\!] &: \mathbb{S}_d \to \mathbb{S}_d \\ \Delta_N[\![_]\!](\langle s, \nabla s \rangle) &\to (\langle s, \nabla s \rangle) \,. \end{aligned} \tag{18}$$

The dual signal $\langle s, \nabla s \rangle$ being composed of one primal signal and N gradient signals, we can further state that

$$\Delta_N[\![_]\!] = \|_{l=1}^{N+1} _ \,. \tag{19}$$

3.1.2 Parallel Composition

Blocks composed in parallel (see Figure 1a) do not influence one-another, arithmetically-speaking. The outputs of the program A, B, $y_1, \ldots, y_{o_A}$ and $y_{o_A+1}, \ldots, y_{o_A+o_B}$, are essentially the outputs of separate block diagrams; thus $\Delta_N[\![A,B]\!] = \Delta_N[\![A]\!], \Delta_N[\![B]\!]$, and, more generally

$$\Delta_N \left[\!\!\left[\|_{l=1}^{L} D_l \right]\!\!\right] = \|_{l=1}^{L} \Delta_N[\![D_l]\!] \,. \tag{20}$$

3.1.3 Sequential Composition

Implicit in Equations (1) and (2) are the input and output signals of the Faust block diagram, D, and the following equivalence applies:

$$D = \|_{l=1}^{i_D} _ : D \tag{21}$$

$$D = D : \|_{l=1}^{o_D} _ \,, \tag{22}$$

where i_D and o_D are, respectively, the number of input and output signals of D. Applying derivative transformation to Equation (21), and bringing Equations (19) and (20) to bear,

$$\begin{aligned} \Delta_N[\![D]\!] &= \Delta_N[\![\|_{l=1}^{i_D} _ : D]\!] \\ &= \|_{l=1}^{i_D} \|_{m=1}^{N+1} _ : \Delta_N[\![D]\!] \\ &= \|_{l=1}^{i_D} \Delta_N[\![_]\!] : \Delta_N[\![D]\!] \,. \end{aligned} \tag{23}$$

$$\Delta_N[\![A,B]\!] = \Delta_N[\![A]\!],\Delta_N[\![B]\!] \quad (3)$$

$$\Delta_N[\![A : B]\!] = \Delta_N[\![A]\!] : \Delta_N[\![B]\!] \quad (4)$$

$$\Delta_N[\![A <: B]\!] = \Delta_N[\![A]\!] <: \Delta_N[\![B]\!] \quad (5)$$

$$\Delta_N[\![A \sim B]\!] = \Delta_N[\![A]\!] \sim \Delta_N[\![B]\!] \quad (6)$$

$$\Delta_N[\![A :> B]\!] = \Delta_N[\![A : \chi(i_B,k)]\!] : F : \Delta_N[\![B]\!] \quad (7)$$

$$F = \|_{l=1}^{i_B} \left(\|_{l=1}^{k} \Delta_N[\![_]\!] \; :> \; \Delta_N[\![_]\!] \right)$$

$$k = \frac{o_A}{i_B}$$

Parallel iteration, `par(l, L, D)`:

$$\|_{l=1}^{L} D_l = D_1, \ldots, D_L \quad (8)$$

Signal interleaving function:

$$\chi(r,c) = \|_{l=1-}^{r\times c} <: \|_{m=0}^{r-1} \|_{n=1}^{c} \left(\|_{p=1}^{m} !, -, \|_{q=1}^{r-m-1} ! \right) \quad (9)$$

$$\Delta_N[\![_]\!] = \|_{l=1}^{N+1} - \quad (10)$$

$$\Delta_N[\![!]\!] = \|_{l=1}^{N+1} ! \quad (11)$$

$$\Delta_N[\![\texttt{mem}]\!] = \|_{l=1}^{N+1} \texttt{mem} \quad (12)$$

Fig. 4: Automatic differentiation rules for Faust's composition operators, plus supporting primitive operators and routing expressions. Use of $\Delta_N[\![\texttt{mem}]\!]$ is implicit in the definition of $\Delta_N[\![A \sim B]\!]$. N is the number of input variables with respect to which partial derivatives are calculated.

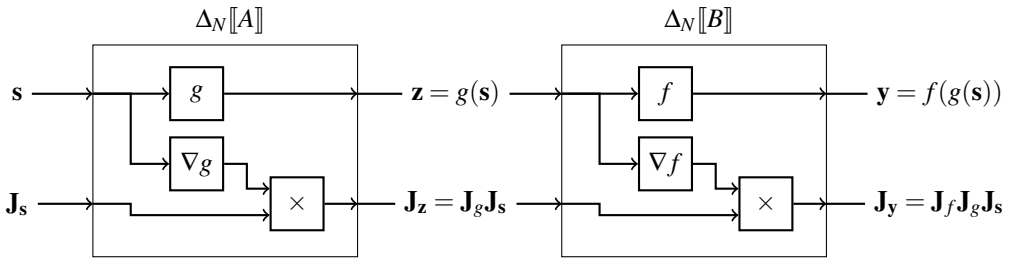


Fig. 5: Derivative transformation of the sequential composition of two Faust blocks, $\Delta_N[\![A : B]\!]$, is analogous to composition of two mathematical functions of a vector of dual signals, $f(g(\langle \mathbf{s}, \mathbf{J_s} \rangle))$, and is thus equivalent to the sequential composition of the blocks following separate derivative transformation.

The arithmetical transparency of sequential composition holds in the general case; expanding on the semantic scheme in Equation (2), $\Delta_N[\![A : B]\!]$ is analogous to composing two functions, g and f:

$$\begin{aligned} &\Delta_N[\![A : B]\!](\langle \mathbf{s}, \mathbf{J_s} \rangle) \to (\langle \mathbf{y}, \mathbf{J_y} \rangle) \\ &\langle \mathbf{y}(t), \mathbf{J_y}(t) \rangle = f(g(\langle \mathbf{s}(t), \mathbf{J_s}(t) \rangle)) \\ &\qquad = \left\langle f(g(\mathbf{s}(t))), \mathbf{J}_f(t)\mathbf{J}_g(t)\mathbf{J_s}(t) \right\rangle . \end{aligned} \quad (24)$$

As illustrated in Figure 5, the composition of functions of dual signals has the effect of primal function composition on the one hand, and application of the chain rule of differentiation on the other, and works so long as $o_A = i_B$,[2] thus

$$\Delta_N[\![A : B]\!] = \Delta_N[\![A]\!] : \Delta_N[\![B]\!] . \quad (25)$$

[2]Though not the case in Faust's early days [8], the equality of o_A and i_B is long-established as an absolute condition of sequential composability.

3.1.4 Split Composition

Composing A and B, $i_B = ko_A$, with the split composition operator (see Figure 1c) duplicates the output signals of A, k times, passing each group of duplicates to B, in order:

$$A <: B = A \; : \; \|_{l=1-}^{o_A} <: \; \|_{m=1}^{k} \|_{l=1-}^{o_A} \; : \; B . \quad (26)$$

Signal order is maintained within each group of duplicates, so, applying derivative transformation to both sides of Equation (26), and recalling Equation (23),

$$\begin{aligned} &\Delta_N[\![A <: B]\!] \\ &= \Delta_N[\![A]\!] : \|_{l=1}^{o_A} \Delta_N[\![_]\!] <: \|_{m=1}^{k} \|_{l=1}^{o_A} \Delta_N[\![_]\!] : \Delta_N[\![B]\!] \\ &= \Delta_N[\![A]\!] <: \Delta_N[\![B]\!] . \end{aligned} \quad (27)$$

3.1.5 Cut Primitive

The cut primitive, notated !, terminates a signal, and combinations of the cut and identity functions can

be used to create complex signal routings. For instance, to interleave two groups of two signals, such that $s_{1,1}, s_{1,2}, s_{2,1}, s_{2,2}$ becomes $s_{1,1}, s_{2,1}, s_{1,2}, s_{2,2}$, one may write

$$_,_,_,_ \;<:\; _,!,_,!,!,_,!,_\,. \tag{28}$$

To differentiate an expression of this kind, the appropriate accompaniment to Equations (19) and (27) for the derivative of the cut primitive is

$$\Delta_N[\![!]\!] = \left\|_{l=1}^{N+1}\right. !\,. \tag{29}$$

A general interleaving function, as used in Listing 2, can be written using the cut and identity primitives, plus parallel and split composition (see Listing 3). Comprising only differentiable elements, it too is differentiable.

```
interleave(r, c) = par(l, r * c, _)
  <: par(m, r,
       par(n, c, cutpq(m, r - m - 1)))
with {
  cutpq(0, q) = _,par(l, q, !);
  cutpq(p, 0) = par(l, p, !),_;
  cutpq(p, q) =
    par(l, p, !),cutpq(0, q);
};
```

Lst. 3: A Faust function, $\chi(r,c)$ (see Equation (9) for its mathematical definition), that interleaves c groups of r signals into r groups of c signals.

3.1.6 Delay Primitive

Faust provides two primitives for delaying signals; here we will focus on the `mem` primitive, which, when composed in sequence with a signal, applies a one-sample delay. In general, a fixed delay can be represented as the convolution of a signal, $s(x,t)$, with a time-shifted impulse [29],

$$s(x,t-\tau) = s(x,t) * h(t)\,, \quad h(t) = \delta(t-\tau)\,. \tag{30}$$

Taking the temporal-parametric Fourier transform of the derivative of the convolution with respect to x, then using the convolution and differentiation properties of the transform [29], yields

$$\begin{aligned} &\mathscr{F}\left[\frac{\partial}{\partial x}\left(s(x,t) * h(t)\right)\right] \\ &= j\omega_x S(j\omega_x, j\omega_t) H(j\omega_t) \qquad (31)\\ &= \mathscr{F}\left[\frac{\partial}{\partial x} s(x,t)\right] \cdot \mathscr{F}\left[h(t)\right]\,, \qquad (32)\end{aligned}$$

which demonstrates that differentiation is time-invariant, and allows us to deduce the convolution rule of differentiation — to differentiate a convolution we need only differentiate one of the functions being convolved. Applied to dual numbers,

$$\langle s, \nabla s\rangle * \langle h, \nabla h\rangle = \langle s * h, \nabla s * h\rangle\,, \tag{33}$$

giving us the following semantic scheme:

$$\begin{aligned} \Delta_N[\![\texttt{mem}]\!] &: \mathbb{S}_d \to \mathbb{S}_d \\ \Delta_N[\![\texttt{mem}]\!](\langle s, \nabla s\rangle) &\to (\langle y, \nabla y\rangle) \\ \langle y(t), \nabla y(t)\rangle &= \langle s(t-1), \nabla s(t-1)\rangle\,. \end{aligned} \tag{34}$$

3.1.7 Recursive Composition

The differentiability of recursive filters is well-established [30] — in short, the derivative of a recursive expression is itself recursive. In the specific case of Faust's recursive composition operator (see Figure 1d), to establish a rule for derivative transformation we can take a similar approach to Equation (24), making g an inner function that acts upon a vector of past output signals:

$$\begin{aligned} \langle \mathbf{y}(t), \mathbf{J}_\mathbf{y}(t)\rangle = \Big\langle & f(\mathbf{s}(t), g(\mathbf{y}(t-1))), \\ & \frac{\partial f}{\partial \mathbf{s}(t)}\mathbf{J}_\mathbf{s}(t) + \frac{\partial f}{\partial g(\mathbf{y}(t-1))}\mathbf{J}_g(t-1)\mathbf{J}_\mathbf{y}(t-1)\Big\rangle\,. \end{aligned} \tag{35}$$

In the absence of a feedforward component (notwithstanding the fact that such a system would produce constant output signals of value zero), the result simplifies to

$$\langle f(g(\mathbf{y}(t-1))),\, \mathbf{J}_f(t-1)\mathbf{J}_g(t-1)\mathbf{J}_\mathbf{y}(t-1)\rangle\,, \tag{36}$$

which, comparing with Equation (24), tells us that recursive composition is simply *sequential composition through time*, and points to the implicit, and trivially-differentiable, one-sample delay introduced by Faust's recursive composition operator. We may conclude that

$$\Delta_N[\![A \sim B]\!] = \Delta_N[\![A]\!] \sim \Delta_N[\![B]\!]\,. \tag{37}$$

3.1.8 Merge Composition

Merge composition (Figure 1e) entails summation [8], thus the addition rule of differentiation must be obeyed when merging dual signals. A merge $A :> B$, can be viewed as an interleaving of the outputs of A into i_B groups of $k = o_A/i_B$ signals, followed by a merging of each group into one signal path:

$$A :> B = A : \chi(i_B, k) : \|_{l=1}^{i_B} \left(\|_{m=1}^{k} _ :> _ \right) : B \,. \quad (38)$$

Clearly $\Delta_N[\![A :> B]\!]$ cannot be defined recursively, however. Note that for $i_B = 1$ the right hand side of Equation (38) reduces to $A :> _$, which is essentially the term in parenthesis, and can be handled thanks to the definitions in Equations (19) and (20),

$$\Delta_N[\![\|_{m=1}^{k} _ :> _]\!] = \|_{m=1}^{k} \Delta_N[\![_]\!] :> \Delta_N[\![_]\!] \,. \quad (39)$$

The order of signals is maintained, so the resulting dual signal is guaranteed to be of the desired form $\left\langle \sum_{m=1}^{k} s_m, \sum_{l=m}^{k} \nabla s_m \right\rangle$. For the full expression for differentiable merge composition, see Equation (7).

4 Discussion

The rules for differentiating Faust's composition operators are collected in Figure 4. With the implementation of these rules via pattern matching, Faust programs can be recursively decomposed according to their composition operators, and re-composed in differentiable fashion, satisfying the syntactic rule in Equation (15). For example:

$$\begin{aligned} \Delta_N[\![(A, C) : B]\!] &= \Delta_N[\![A, C]\!] \;:\; \Delta_N[\![B]\!] \\ &= \Delta_N[\![A]\!], \Delta_N[\![C]\!] \;:\; \Delta_N[\![B]\!] \,. \end{aligned} \quad (40)$$

An artificial neuron of the form $y = \sigma(\mathbf{w}\mathbf{x} + b)$, such as could be used in a binary classification problem, can be created as per Listing 4. The code describes a neuron and sigmoid activation function in primal form, in a library file, `diff.lib`; the `dot` function (line 14) makes use of the familiar `interleave` function to compute the dot product, **wx**; in `neuron.dsp`, differentiable weights and biases, `wb`, are created according to a network layer specification (line 27) — one pair of signals indicating one layer containing one neuron with `NW` weights; weights and biases are used to create a differentiable environment (`d`, line 28; described in Section 2.2 and Listing 2); the wrapper-function `diffInN`

```
1  // diff.lib
2  neuron(NINPUTS, activationFunction) =
3    WX + b : activationFunction
4  with {
5    WX = linalg.dot(NINPUTS);
6    b = _;
7  };
8
9  activations = environment {
10   sigmoid = *(-1) : 1 / (1 + exp);
11 }
12
13 linalg = environment {
14   dot(N) = interleave(N, 2)
15     : par(i, N, *) :> _;
16 };
17
18 // neuron.dsp
19 NW = 3; // Number of weights.
20
21 // Send inputs, and weights/biases,
22 // to the differentiated neuron.
23 process = par(i, NW, _),
24   par(i, NW + 1, wb.var(i + 1))
25   : d.diffInN(neuron, NW)
26 with {
27   wb = weightsAndBiases((1, NW));
28   d = df.env(wb);
29   neuron = d.neuron(NW,
30     d.activations.sigmoid);
31 };
```

Lst. 4: Excerpt from a differentiable artificial neuron implementation. See Section 4 for detail.

(line 25) enacts the recursive derivative transformation; its first `NW` inputs are considered input signals, **s**, the rest as input variables, **x**.

Worthy of reflection is that algorithms of this sort, consisting of differentiable primitives, plus our now-differentiable composition operators, can be automatically differentiated to arbitrary order, $\Delta_N[\![\Delta_N[\![\cdots \Delta_N[\![\cdot]\!] \cdots]\!]\!]$.

4.1 Limitations

Although the scheme presented here renders a great deal of existing Faust code differentiable, certain primitives and language constructs cannot, at present, be subjected to AD.

Faust's `route` primitive cannot be pattern-matched, which precludes the derivative transformation of arbitrary signal routing expressions based on that primitive, such as those in the `routes.lib` library.[3] As described in Section 3.1.5, however, such expressions can be recreated with the identity and cut primitives; common routing tasks like interleaving can be achieved differentiably with code such as in Listing 3.

User interface primitives (sliders and buttons, for example) are matchable in principle, but in practice, `hslider("x1", 0, -1, 1, .01)`, whose parameters are a string identifier, initial value, minimum, maximum and step, must be matched verbatim; clearly, in the general case, this is not practical. The obvious derivative transformation of a user interface element is to an input variable, i.e. a parameter for optimisation such as a weight in a neural network. Input variables must be identifiable so as to create the appropriate corresponding dual signal, $\left\langle x_i, \begin{bmatrix} 0 & \cdots & \frac{\partial x_i}{\partial x_i} & \cdots & 0 \end{bmatrix}^T \right\rangle$. The approach proposed is to move input variables out of the general AD transformation; a wrapper function is used to indicate the number of input signals (whose gradients are zero vectors) and input variables received by the main body of the differentiated block diagram (see Listings 4 and 5). Though imposing a certain coding convention, we consider this comparable with the particular requirements imposed by PyTorch, TensorFlow, and JAX, with their respective dedicated datatypes for learnable parameters.

Though not a frequently-employed feature of the language, unapplied lambda abstractions appear in some key definitions, the `iir` function in `filters.lib`, for example. Implemented in transposed-direct-form [31], `iir` is defined with the abstraction `ma.sub`, which accepts two arguments, `x` and `y`, and returns the result `y-x`. Attempting $\Delta_N[\![\texttt{sub}]\!]$, the abstraction is essentially treated as a numerical constant, and given a vector of zeros as its gradient. Fortunately, constructions of this kind — in this case reversing signal order before applying some primitive operation — are effectively matters of signal-routing and, as described in Section 3.1.5, can be recreated using differentiable primitives.

4.2 Applications & Outlook

Despite the above limitations, and beyond the example given in Listing 4, the AD system described here can be put to practical use. A differentiable transposed-direct-form IIR filter implementation is given in Listing 5; though reliant on some careful signal routing, particularly for directing coefficients to the feedback (`fir(N)`, line 7) and feedforward (`fir(N + 1)`, line 9) portions of the filter, it overcomes the obstacles to differentiability posed by its counterpart in `filters.lib`. Structurally, it is similar to the neuron implementation in Listing 4; parameters, in this case learnable filter coefficients, are defined outside of the differentiable environment, and passed, along with an input signal, to the wrapper function `diffInN`. An expanded version of this example (plus that given in Listing 4) is available online[4] and can be run in Faust's web IDE.[5] Comparing, via a differentiable loss function, the output of the filter against that of an equivalent filter with fixed, *hidden* parameters, and backpropagating the resulting gradients, learnable parameters can be optimised in real time. This generalises to filters of higher order and those described by metaparameters such as cut-

```
1  N = 2; // filter order.
2
3  // ...Set up AD vars & environment...
4
5  // Differentiable TDF IIR filter.
6  iir = ((sub,par(l, N, _))
7    ~ (fir(N))),par(l, N + 1, _)
8    : _,par(l, N, !),par(l, N + 1, _)
9    : fir(N + 1)
10 with {
11   sub = _,_ <: !,_,_,! : -;
12   fir(n) =
13     (_ <: par(l, n, seq(m, l, mem))),
14       par(l, n, _)
15     : linalg.dot(n);
16 };
17
18 // Send a signal input, and coeffs,
19 // to the differentiated filter.
20 process = _,coeffs : d.diffInN(iir, 1)
21 with {
22     coeffs = par(i, 2 * N + 1,
23                  vars.var(i + 1));
24 };
```

Lst. 5: Excerpt from a differentiable IIR filter implementation.

[3] https://github.com/grame-cncm/faustlibraries/blob/65594336a3476d9a82a4df71e88c92f3e4b94821/routes.lib

[4] https://codeberg.org/hatchjaw/aes-faust-ddsp

[5] https://faustide.grame.fr/

off frequency, and the length of multiple hidden delay lines can be found by optimisation against a differentiable FIR filterbank; these examples point the way to a variety of real-time adaptive processing applications.

Though demonstrated with *known* algorithms, in principle the above can be extended to optimisation against a target *black box* filter, granted that both target and estimate receive the same input signal. This hints, however, at another limitation of the current approach — Faust is primarily concerned with time-domain operations, thus loss is computed in the time domain on a per-sample basis (albeit with the possibility of taking a running-average of recent sample values, for example). For phase-invariant and perceptually-informed optimisation scenarios, frequency-domain loss computation is required; the recent introduction of multi-rate processing to the Faust language promises to enable spectral signal analysis. Developments of this sort will improve the feasibility of taking on larger-scale, typically deferred-time problems such neural audio synthesis. These call for generalisation to a corpus of training data (rather than optimisation to a small number of input variables); with that end in mind, the example in Listing 4 should be extended to a general neural network structure, and a supporting infrastructure created for delivering training data and capturing weights and biases for real-time inference.

5 Conclusion

In this paper we have described a general scheme for differentiating Faust programs, moving an important step toward providing the means with which to tackle audio optimisation and machine learning problems with the domain-specific and cross-platform benefits of the Faust language. A variety of time-domain, gradient-based optimisation problems can be tackled in Faust, with results produced in real-time.

Many challenges and opportunities remain as impetus for further development of the work presented here. Having only been put to informal use to-date, the proposed AD approach should be assessed for its efficacy and efficiency. Beyond the correctness of the semantic scheme, being based on forward-mode AD, which, on paper, compares unfavourably with reverse-mode in terms of computational expense, its combination with the optimisations of the Faust compiler should be explored. At the level of the Faust compiler, changes that would facilitate the transformation of elements of the language that currently elude pattern matching would be of benefit here, as would further development of Faust's adaptability to frequency domain processing.

References

[1] Baydin, A. G. et al. "Automatic Differentiation in Machine Learning: A Survey". In: *Journal of Machine Learning Research* 18 (2018).

[2] Hayes, B. et al. "A Review of Differentiable Digital Signal Processing for Music and Speech Synthesis". In: *Frontiers in Signal Processing* 3 (Jan. 11, 2024).

[3] Alexandre, L., Lecomte, P., and Galland, M.-A. "Experimental Active Noise Control Using Faust On FPGA: Comparison Between A Multi-Point and Spherical Harmonics Method". In: *Forum Acusticum 2023 - 10th Convention of the European Acoustics Association*. Torino, Italy, Sept. 2023.

[4] Engel, J. et al. "Differentiable Digital Signal Processing". In: *Proceedings of the Eighth International Conference on Learning Representations*. Online, 2020.

[5] Rushton, T. A. "Differentiable DSP in Faust". In: *Proceedings of the International Faust Conference (IFC-24)*. Turin, Italy, Nov. 2024.

[6] Orlarey, Y., Fober, D., and Letz, S. "FAUST: An Efficient Functional Approach to DSP Programming". In: *New computational paradigms for computer music* (2009).

[7] Orlarey, Y., Fober, D., and Letz, S. "An Algebra for Block Diagram Languages". In: *Proceedings of International Computer Music Conference*. Gothenburg, Sweden, 2002.

[8] Orlarey, Y., Fober, D., and Letz, S. "Syntactical and Semantical Aspects of Faust". In: *Soft Computing* 8.9 (2004).

[9] Orlarey, Y. et al. "A New Intermediate Representation for Compiling and Optimizing Faust Code". In: *Proceedings of the 2nd International Faust Conference (IFC-20)*. Paris, France, Dec. 1, 2020.

[10] Letz, S., Orlarey, Y., and Fober, D. "An Overview of the FAUST Developer Ecosystem". In: *Proceedings of the 1st International Faust Conference (IFC-18)*. Mainz, Germany, July 17, 2018.

[11] Letz, S., Orlarey, Y., and Fober, D. “Compiling Faust Audio DSP Code to WebAssembly”. In: *Web Audio Conference*. London, United Kingdom, Aug. 2017.

[12] Gaster, B. R. and Cole, M. “Audio Anywhere with Faust”. In: *Proceedings of the 2nd International Faust Conference (IFC-20)*. Paris, France, 2020.

[13] Albouy, A. and Letz, S. “Faust Audio DSP Language for JUCE”. In: *Proceedings of the Linux Audio Conference*. Saint-Etienne, France, May 18, 2017.

[14] Michon, R. et al. “Embedded Real-Time Audio Signal Processing With Faust”. In: *Proceedings of the 2nd International Faust Conference (IFC-20)*. Paris, France, Dec. 2020.

[15] Cochard, P. et al. “Programming FPGA Platforms for Real-Time Audio Signal Processing In C++”. In: *Proceedings of the 2024 Sound and Music Computing Conference (SMC-24)*. Porto, Portugal, July 2024.

[16] Michon, R. et al. “The Faust Programming Language As a Platform for Creating Hybrid Acoustical and Digital Musical Instruments”. In: *Forum Acusticum 2020 Proceedings*. Lyon, France, Dec. 2020.

[17] Caren, M., Michon, R., and Wright, M. “The KeyWI: An Expressive and Accessible Electronic Wind Instrument”. In: *Proceedings of The International Conference on New Interfaces for Musical Expression (NIME ’20)*. Birmingham, UK, July 21, 2020.

[18] Hansen, M. “Weather Organ: A Sparse Stochastic Synthesizer in Faust”. In: *Proceedings of the 1st International Faust Conference (IFC-18)*. Mainz, Germany, July 17, 2018.

[19] Roosenburg, D. and Michon, R. “A Wave Digital Filter Modeling Library for the Faust Programming Language”. In: *Proceedings of the 18th Sound and Music Computing Conference*. Online, June 29, 2021.

[20] Russo, R. et al. “Introducing Finite Difference Schemes Synthesis in Faust: A Cellular Automata Approach”. In: *Proceedings of the 18th Sound and Music Computing Conference*. Online, June 29, 2021.

[21] Michon, R. et al. “The Faust Physical Modeling Library: A Modular Playground for the Digital Luthier”. In: *Proceedings of the 1st International Faust Conference (IFC-18)*. Mainz, Germany, 2018.

[22] Braun, D. *DawDreamer: Bridging the Gap Between Digital Audio Workstations and Python Interfaces*. Nov. 18, 2021. arXiv: 2111.09931. Pre-published.

[23] Rall, L. B. “The Arithmetic of Differentiation”. In: *Mathematics Magazine* 59.5 (Dec. 1986).

[24] Harrison, D. *A Brief Introduction to Automatic Differentiation for Machine Learning*. Oct. 14, 2021. arXiv: 2110.06209. Pre-published.

[25] Wang, F. et al. “Demystifying Differentiable Programming: Shift/Reset the Penultimate Backpropagator”. In: *Proceedings of the ACM on Programming Languages* 3 (ICFP July 26, 2019).

[26] Yu, W. and Blair, M. “DNAD, a Simple Tool for Automatic Differentiation of Fortran Codes Using Dual Numbers”. In: *Computer Physics Communications* 184.5 (May 2013).

[27] Li, Y. and Wang, G. “ChAI => Interactive AI Tools in ChucK”. In: *The International Conference on New Interfaces for Musical Expression (NIME ’24)*. Utrecht, The Netherlands, Sept. 4, 2024.

[28] Kuznetsov, B., Parker, J. D., and Esqueda, F. “Differentiable IIR Filters for Machine Learning Applications”. In: *Proceedings of the 23rd International Conference on Digital Audio Effects (DAFx-20)*. Vienna, Austria, 2020.

[29] McClellan, J. H., Schafer, R. W., and Yoder, M. A. *Signal Processing First*. Pearson, 2003.

[30] Shynk, J. J. “Adaptive IIR Filtering”. In: *IEEE ASSP Magazine* 6.2 (Apr. 1989).

[31] Smith, J. O. *Introduction to Digital Filters: With Audio Applications*. W3K Publishing, 2008.

Audio Engineering Society

Conference Paper 29

Presented at the AES International Conference on Machine Learning and Artificial Intelligence for Audio
2025 September 8–10, London, UK

This paper was peer-reviewed as a complete manuscript for presentation at this conference. This paper is available in the AES E-Library (http://www.aes.org/e-lib), all rights reserved. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Journal of the Audio Engineering Society.

Integrating IP broadcasting with audio tags: Workflow and challenges

Rhys Burchett-Vass[2], Arshdeep Singh[1,2], Gabriel Bibbó[1,2], and Mark D. Plumbley[1,2]

[1]*Centre for Vision Speech and Signal Processing (CVSSP)*
[2]*University of Surrey, UK*

Correspondence should be addressed to Arshdeep Singh (arshdeep.singh@surrey.ac.uk)

ABSTRACT

The broadcasting industry has adopted IP technologies, revolutionising both live and pre-recorded content production, from news gathering to live music events. IP broadcasting allows for the transport of audio and video signals in an easily configurable way, aligning with modern networking techniques. This shift towards an IP workflow allows for much greater flexibility, not only in routing signals but with the integration of tools using standard web development techniques. One possible tool could include the use of live audio tagging, which has a number of uses in the production of content. These could include adding sound effects to automated closed captioning or identifying unwanted sound events within a scene. In this paper, we describe the process of containerising an audio tagging model into a microservice, a small segregated code module that can be integrated into a multitude of different network setups. The goal is to develop a modular, accessible, and flexible tool capable of seamless deployment into broadcasting workflows of all sizes, from small productions to large corporations. Challenges surrounding latency of the selected audio tagging model and its effect on the usefulness of the end product are discussed.

1 Introduction

The audio track contains a wide array of descriptive information about the sound events in a scene. Detecting sound events in real-time could have a number of uses in the broadcasting industry, from aiding operators in programme creation to enhancing end user accessibility. For example, BBC Research and Development [1] employed a sound event detection framework to identify sounds that may disrupt the ambience of a program. This work was targeted at the BBC's *Autumnwatch* programme which uses wildlife cameras capturing the movement of animals. To avoid undesirable noises interrupting the live stream, such as cars passing by or people talking, an icon is overlayed onto the operator's interface, indicating the undesirable sound event so the operator does not to switch to that source. Another use of sound event detection is closed captioning. While the majority of the existing work in captioning broadcast audio has been focused on analysing the speech events [2, 3], only a few attempts of identifying sound events in a real time (live) transmission has been conducted [1]. To achieve full captioning (closed captions of both sound events and speech) within IP broadcasting, general sound event detection models that detect sound events including speech are required.

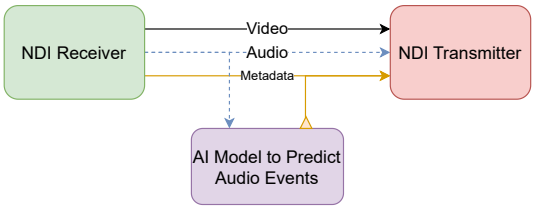


Fig. 1: Basic flow of audio, video and metadata frames travelling through their respective data streams using Network Device Interface (NDI) system.

Internet Protocol (IP) broadcasting describes the process of transmitting audio and video signals from one location to another using IP networking. One approach traditionally used for transmitting audio/video is Serial Digital Interface (SDI), with fixed connections between dedicated hardware devices. IP broadcasting allows software to replace some of these hardware devices, enabling greater scalability and easy re-configuration. Cloud technology and containerisation methods such as Docker [4] can be utilised to take advantage of such scalability.

There are a few challenges in creating software for use in an IP broadcasting environment. First, as with most modern web applications, is scalability and containerisation of software, which allows the infrastructure to adapt depending on the demand on the system by starting up new containers when required. Containerisation also allows for the same task to be conducted independently on different streams or sources by having a container per stream. Another advantage of containerisation is that, if a fault occurs on one container, it does not damage the entire system as a whole and can be fixed independently. The second challenge is handling of the inelastic audio and video traffic, which does not adapt well to changes in network conditions, without introducing delay and jitter to the transmission.

To overcome above challenges and integrate sound event data into IP broadcasting, this paper contributes:

- Containerising audio tagging applications to isolate each component from the other elements of the system, i.e. other processing units, transmission and reception code. Code isolation allows the component to run on any machine. Containerisation makes it easy to create multiple instances of a component if needed, allowing for scalability and the use of cloud platforms.
- Leveraging an Artificial Intelligence (AI) model to generate audio tags to transmit meta-information alongside audio. This added information about the contents of the audio track has a number of uses in the area of automation - from better production tools to improved accessibility via more descriptive automated captioning. An overall proposed framework is shown in Figure 1 and more details including challenges around our approach can be found in Section 5. Our codes are available at Github[1].

2 Related work

2.1 A Brief Overview on IP Broadcasting Technology

There are a few technologies currently used for IP broadcasting. The first of these are described in standards from the Society of Motion Picture and Television Engineers (SMPTE) as the ST 2110 suite of standards [5]. SMPTE standards are used by the industry with examples including the Serial Digital Interface (SDI) standards for transmission between equipment over a direct connection, i.e. coaxial or fibre optic cable. Secondly, the Networked Media Open Specifications (NMOS) from the Advanced Media Workflow Association (AMWA) uses ST 2110 along with other standards to define APIs allowing for the connection of multiple receivers and senders on a network in a vendor agnostic way. NMOS is not software, but a set of specifications aiding development in the software. On the other hand, Network Device Interface (NDI) by NewTek [6] is an open standard with fully developed software and Software Development Kit (SDK), designed to allow for easy integration of IP broadcasting into existing software by utilising the NDI SDK.

2.2 Audio Recognition in Broadcasting

There has been some work conducted in broadcasting related to recognition of audio events [1]. However most examples relate to speech recognition and transcription, commonly used for tagging content for archiving purposes. Recognising speech allows for easy searching without having to manually tag content. For example, Raimond et al. [2] describe a system to automate the tagging of content within the BBC's

[1] https://github.com/Rhysbv/panns_ndi

radio archive based on speech audio. Levin et al. [3] describe a system using automatic speech recognition for captioning of news programming. This system runs against a re-speaker, a person repeating speech in a more readable and understandable way for the system, avoiding the issues surrounding the acoustic environment and overlapping speakers. However, this system only supports the processing of speech and does not consider sound events in general. More modern proprietary solutions do exists [7, 8] which remove the requirement for a re-speaker but are still incapable of including sound events. Additionally, BBC Research and Development [1] have designed an application program (a software tool) to identify sound events for the purpose of audio monitoring.

3 System Design

In contrast to previous work, we separate the audio tagging software from any other application programs. In our work, a modular approach is opted for and a container specifically for general audio tagging is built to allow multiple applications on the network to take advantage of the technology without repeating work. This is helpful considering the computational overhead associated with AI models. Our system can include the environment audio event monitoring system as described by BBC Research and Development [1] in addition to other systems e.g. for captioning.

3.1 Selecting IP Broadcasting Approach

For our work, we need an IP based technology that is both well used by the industry and is simple to implement, allowing for wide adoption of the audio tagging technology. Standards have been created to support new IP based workflows. One example discussed in Section 2.1 is the ST 2110 suite of standards authored by SMPTE. These describes the transport of compressed and uncompressed audio, video and metadata via individual Real-time Transport Protocol (RTP) streams. However, the complexity of understanding the suite of standards means that it is only practical for large corporations to implement. An alternative standard mentioned in Section 2.1 is the Network Device Interface (NDI), an open standard created by NewTek [6] which has an easy to use Software Development Kit (SDK). NDI is available in a wide variety of software and hardware applications, enabling its wide spread adoption in both large and small operations. This has lead to NDI being the selected approach for our work. NDI transports data in the form of frames that contain the relevant data as well as supporting information to aid in its use. There are three types of frames used by NDI: audio, video and metadata (Figure 1). NDI also handles the detection of sources allowing for routing of NDI frames.

3.2 AI Model used for Audio Tagging

To identify audio tags, we use convolutional neural networks (CNNs), that have shown good performance in many audio classification tasks [9, 10]. For example, pre-trained audio neural networks (PANNs) [10] have been widely used to recognize a variety of audio events. A description of the CNN models used in this paper for predicting the audio tags is given below.

Pre-trained Audio Neural Networks (PANNs): CNN14 [10] is a pre-trained audio neural network that is trained on Google AudioSet dataset [11]. CNN14 is trained by extracting the log-mel spectrograms from the audio clips. CNN14 has 81M parameters and it takes 21G multiply-accumulate operations (MACs) to predict tags of the audio of length 10 seconds. The trained CNN14 can predict wide range of sound events such as car passing by, speech, siren, animal etc. This helps identifying sounds in the wide array of possible scenarios the system could be exposed to, i.e. different types of broadcast programming such as news gathering in various locations or a panel show within a studio.

Efficient PANNs: E-PANNs [12] is an efficient version of original PANNs with reduced memory requirement (24M parameters) and a reduced computational complexity (13G MACs per 10 seconds audio). The efficient AI models are beneficial in an IP networking environment, especially one involving inelastic traffic (network traffic that is sensitive to variations in delay, e.g. audio and video streams). This will be explored in Section 5.

3.3 NDI Integration

We use the NDI SDK [13] to create a software module including the PANNs algorithm. Due to the reliance on Python based packages within the PANNs module such as PANNs inference [14], we use Python for implementation. Specifically, community Python bindings [15] to interface between Python and the C++ SDK are used to enable NDI support. An additional Python package

is created to simplify the process of integrating NDI into both the PANNs module and potential proof of concept applications described in Section 4. Our Python package contains three classes: A receiver, transmitter and finder as seen in Figure 1. An application can find NDI network sources using the finder class. Frames can be received from an NDI source using the receiver class. Received Frames can then be processed and transmitted by creating its own NDI source using the transmitter class. An example of how this is used here can be seen in Figure 1. The flow of audio, video and metadata frames is uninterrupted between the receiver and transmitter. Each audio frame is intercepted and a copy is taken for analysis while the original copy is sent straight to the transmitter, minimising delay and jitter. One issue surrounding the community supplied Python bindings were the associated bugs, especially surrounding memory management. This led to having to convert each frame to a Python dataclass so that it could be effectively freed and dealt with by the Python garbage collector, an issue that would not have been encountered using the original C++ SDK.

3.4 Integrating Sound Events Metadata

In order for E-PANNs to produce sound event predictions from the PANNs model and make it compatible with other NDI applications, we follow the pipeline as shown in Figure 3 that takes the incoming audio frames from NDI and creates metadata frames containing the audio tag to be sent across the network.

We use two ring buffers. The first ring buffer stores incoming audio frames. From each audio frame, we extract the individual audio samples and store them in a second ring buffer. Once a sufficient number of samples has been collected in the second ring buffer the entire contents of the ring buffer is fed into the PANNs model. The size of the second ring buffer determines the duration of the audio window that PANNs analyses. The impact of the size of the window on the models latency is discussed in Section 5. To distribute the predicted sound event across the NDI network, we use metadata frames. These frames transport XML data, which can include third-party metadata as used here. The output text from PANNs is inserted into an XML template for transmission. Other NDI applications can then receive this XML via the metadata frames to access the sound event prediction.

A summary of various steps is explained below:

1. Store received audio frames in the first ring buffer.
2. Extract the floating point Pulse Code Modulated (PCM) audio samples from each frame and store these in the second ring buffer.
3. Wait until a given number of samples have been collected.
4. Feed the entire contents of the second ring buffer into the CNN model.
5. Generate a metadata frame containing the prediction from the CNN model.

4 Example Workflow

The proposed containerised component allows for the integration of audio tagging capabilities into a multitude of different systems and use cases. Below, we provide two examples of integration of audio tagging system into existing IP broadcasting framework,

4.1 Audiowatch Example

Figure 2 demonstrates a pipeline inspired by the BBC Audiowatch project, showing an audio tagging module separated from other application programsWe use Docker [4] for containerisation, creating multiple instances of the audio tagging software to analyse several NDI sources simultaneously. A sound event detection front end is a dashboard user interface as shown in Figure 3 and it generates metadata corresponding to input audio. Metadata containing sound event information is then sent to the icon selector module for processing. Next, various icon selector containers extract the sound events from the audio track supplied within the metadata frames. After identifying the unwanted sound events, an appropriate icon overlay is transmitted as an NDI video frame. Next, a video mixing software tool such as Open Broadcaster Software (OBS) [16] is used to superimpose the icon onto the original video source for displaying on the operators multiview, which is used to monitor all video sources.

4.2 Online Closed Captioning

Another example integration could be the use of audio tagging to enhance closed captioning. As discussed in Section 2.2, while work has been conducted to automated closed captioning in real time using automatic

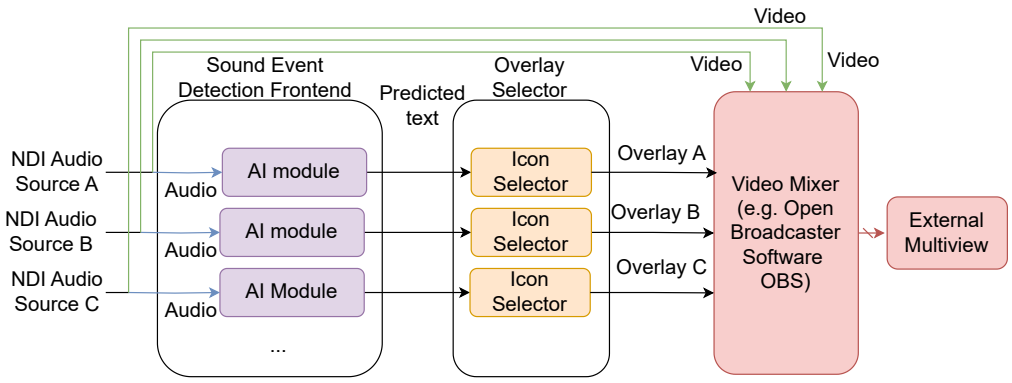


Fig. 2: Proposed integrated pipeline: Audiowatch [1] framework with a separated audio tagging unit.

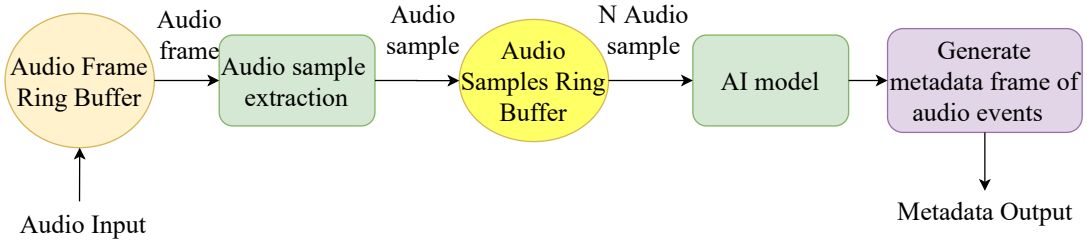


Fig. 3: Metadata generation pipeline.

speech recognition, these do not include descriptions of sound events. By combining the two technologies, full closed captioning could be achieved. This would involve first parsing the audio through the audio tagging model using our container. When the result is returned as human speech, the audio would then be passed through a second speech recognition model to generate subtitles. One major concern would be the accuracy of the audio tagging model. If the speech was not always detected, we would miss large portions of speech text. Additionally the difference in latency between a sound event being inserted and speech going through two models would have to be accounted for.

5 AI model Integration Challenges

There are a number of integration challenges to consider while designing AI based software fit for broadcasting. These challenges include the accuracy of the prediction and the latency of the model delaying the signal. Generally, PANNs and E-PANNs give similar prediction results.

Model latency: The latency of the model here describes the amount of time it takes given a number of audio samples to produce an accurate sound event prediction. Consideration of the model latency is significant given that we are dealing with inelastic audio and video network traffic. This means that any delay in processing contributes to a delay in the resulting transmission depending on the infrastructure. The delay can be mitigated using a design similar to that shown in Figure 1, however there is the issue of predictions being desynchronised from events heard in the audio track Although we have minimal control over the IP network using the audio tagging module, and thus cannot manage the network's latency, we can still select an optimal model that minimises latency while maintaining accuracy.

Buffer size versus model latency: To analyse buffer size and latency of model, we performed experiments using a set of audio recordings with known sound events. The first audio recording is taken directly from the PANNs repository. It involves a telephone ringing followed by human speech and is of seven seconds.

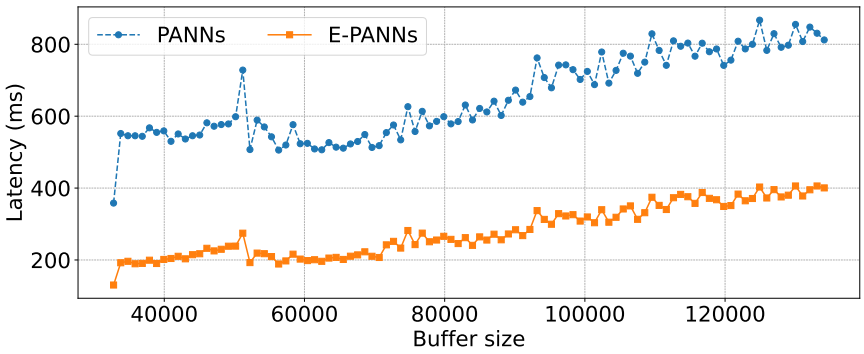


Fig. 4: PANNs/E-PANNs model latency vs buffer size when inputting audio sampled at 48KHz. Experiments are performed a on AMD Ryzen 5 2500U system at 2GHz.

The second audio recording of a car driving into the distance. The third audio recording is created by mixing a car driving and a running river sound events.

Given the audio recordings, we analyse the latency of the CNN model at different number of audio samples. We generate different length audio segments. The audio samples taken are of multiples of 1024 (assuming frames containing 1024 samples are used) and represents the size of the buffer. Given the audio samples of different length, we use the PANNs or E-PANNs model to produce predictions while measuring the time taken for the model to produce a prediction. Figure 4 shows the latency of the PANNs and E-PANNs models at different buffer size. Both PANNs and E-PANNs follow a similar trajectory, with E-PANNs showing a considerable improvement in latency. This suggests that choosing an appropriate model contributes to try to improve latency and hence making integration of audio events more real-time while using less resources.

Experiments found that in our example a buffer size of 47 frames with 1024 samples each This equates to an audio window with a duration of 1.002s sampled at 48KHz, that gives correct results with the minimal latency. Model latency computed on an AMD Ryzen 5 2500U and Intel Core i9-13900HX hardware can be found in Figure 4 and prediction results can be found in Table 1.

Table 1: Comparison of PANNs and E-PANNs predictions across different buffer sizes

Buffer Size	Groundtruth	PANNs	E-PANNs
32768	Tone Car River & Car Phone & Voice	Sine wave Music Waves, surf Telephone bell ringing	Sine wave Music Boat, water vehicle Telephone bell ringing
38912	Tone Car River & Car Phone & Voice	Sine wave Music Vehicle Music	Sine wave Music Boat, water vehicle Telephone bell ringing
48128	Tone Car River & Car Phone & Voice	Sine wave Silence Vehicle Music	Sine wave Vehicle Vehicle Telephone bell ringing
56320	Tone Car River & Car Phone & Voice	Sine wave Vehicle Vehicle Music	Sine wave Vehicle Vehicle Telephone bell ringing
61440	Tone Car River & Car Phone & Voice	Sine wave Vehicle Vehicle Telephone bell ringing	Sine wave Vehicle Vehicle Telephone bell ringing
65536	Tone Car River & Car Phone & Voice	Sine wave Vehicle Vehicle Telephone bell ringing	Sine wave Vehicle Vehicle Telephone bell ringing
102400	Tone Car River & Car Phone & Voice	Sine wave Vehicle Waves, surf Telephone bell ringing	Sine wave Vehicle Vehicle Telephone bell ringing
134144	Tone Car River & Car Phone & Voice	Sine wave Vehicle Vehicle Speech	Sine wave Vehicle Vehicle Speech

6 Discussion and Conclusion

The integration of IP broadcasting with audio tagging offers significant potential for enhancing broadcast workflows, but it also presents several challenges. The transition to IP broadcasting enables a more flexible, scalable, and reconfigurable infrastructure compared to traditional methods based on Serial Digital Interface (SDI). This flexibility is further enhanced by containerisation technologies making the system more resilient and adaptable. However, implementing an audio tagging system introduces challenges primarily related to latency and the accuracy of audio tagging models.

One of the primary challenges discussed is the latency

associated with the audio tagging model. Given the real-time nature of broadcasting, any delays introduced by processing can impact the overall operation. This makes the choice of buffer size crucial. A smaller buffer reduces latency but might compromise the accuracy of sound event detection. Conversely, a larger buffer improves accuracy but increases latency. Experiments conducted (Table 1) show that an acceptable balance can be achieved with a buffer size of 48128 samples, which provides an acceptable latency while maintaining accuracy. The use of Efficient PANNs (E-PANNs) further helps in reducing the computational complexity and memory requirements, making it a suitable choice for real-time applications.

Containerisation offers a robust solution to scalability issues. By isolating the audio tagging functionality into a microservice, it becomes possible to scale the system by simply adding more containers as needed. This isolation also ensures that a fault in one container does not affect the entire system, enhancing overall reliability. The use of Docker to containerise these services allows for easy deployment and management across different network setups. Additionally, the integration with NDI, which is widely adopted in the industry, ensures broad applicability.

Despite these advantages, real-world deployment of such a system is not without hurdles. The reliance on Python bindings to interface with the NDI SDK, while practical, introduces potential issues with memory management that need careful handling.

6.1 Conclusion

Integrating IP broadcasting with audio tagging presents a promising advancement for the broadcasting industry. The use of containerisation and audio tagging for real-time sound event detection can significantly enhance content production and accessibility. However, addressing the challenges of latency, accuracy and real-world deployment is crucial for the successful implementation of this approach. Future work includes re-writing the codebase to use the NDI C++ SDK directly, avoiding the issues surrounding the community Python bindings. Additionally, we would like to analyse more complex models such as transformers [17, 18] within our broadcasting framework. Finally, the creation of the discussed proof of concept applications would allow for full demonstration of the usefulness of this technology.

7 Acknowledgements

This work was supported by the Engineering and Physical Sciences Research Council [grant number EP/T019751/1] and an Adobe Research Gift. For the purpose of open access, the authors have applied a Creative Commons Attribution (CC BY) licence to any Author Accepted Manuscript version arising.

References

[1] Ward, S. and Dawes, R., "AudioWatch - Live audio monitoring for Autumnwatch 2021 - BBC R&D," 2021.

[2] Raimond, Y., Lowis, C., Hodgson, R., and Tweed, J., "Automated semantic tagging of speech audio," in *Proceedings of the 21st International Conference on World Wide Web*, pp. 405–408, 2012.

[3] Levin, K., Ponomareva, I., Bulusheva, A., Chernykh, G., Medennikov, I., Merkin, N., Prudnikov, A., and Tomashenko, N., "Automated closed captioning for Russian live broadcasting," in *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[4] Docker, "Docker: Accelerated Container Application Development," 2022, https://www.docker.com/.

[5] SMPTE, "SMPTE OV 2110-0:2018 - SMPTE Overview Document - Professional Media over Managed IP Networks Roadmap for the 2110 Document Suite," 2018, https://cta-redirect.hubspot.com/cta/redirect/5253154/c8fbb67b-5cba-421d-ad95-29de41239ef7.

[6] NewTek, "NDI 5.6 White Paper," Technical report, NewTek, 2023, https://ndi.video/wp-content/uploads/2023/09/NDI-5.6-White-Paper-2023.pdf.

[7] IBM, "Closed Captioning Software - IBM Watson Media," 2025, https://www.ibm.com/consulting/closed-captioning.

[8] "enCaption: Automated Closed Captioning System | ENCO Systems," 2025.

[9] Purwins, H., Li, B., Virtanen, T., Schlüter, J., Chang, S.-Y., and Sainath, T., "Deep learning for audio signal processing," *IEEE Journal of Selected Topics in Signal Processing*, 13(2), pp. 206–219, 2019.

[10] Kong, Q., Cao, Y., Iqbal, T., Wang, Y., Wang, W., and Plumbley, M. D., "PANNs: Large-scale pretrained audio neural networks for audio pattern recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28, pp. 2880–2894, 2020.

[11] Gemmeke, J. F., Ellis, D. P., Freedman, D., Jansen, A., Lawrence, W., Moore, R. C., Plakal, M., and Ritter, M., "Audio Set: An ontology and human-labeled dataset for audio events," in *IEEE International Conference on acoustics, Speech and Signal Processing (ICASSP)*, pp. 776–780, 2017.

[12] Singh, A., Liu, H., and Plumbley, M. D., "E-PANNs: Sound Recognition Using Efficient Pre-trained Audio Neural Networks," in *INTER-NOISE and NOISE-CON Congress and Conference Proceedings*, volume 268, pp. 7220–7228, Institute of Noise Control Engineering, 2023.

[13] NewTek, "NDI SDK V6.0.1," 2024, https://ndi.video/for-developers/ndi-sdk.

[14] kong, Q., "qiuqiangkong/panns_inference," 2020, original-date: 2020-03-08T06:22:30Z.

[15] Kondo, N., "NDI-Python bindings," 2022, https://github.com/buresu/ndi-python.

[16] Project, O., "OBS studio project," 2025, https://github.com/obsproject/obs-studio.

[17] Gong, Y., Chung, Y.-A., and Glass, J., "AST: Audio spectrogram transformer," *Interspeech*, 2021.

[18] Schmid, F., Koutini, K., and Widmer, G., "Efficient large-scale audio tagging via transformer-to-CNN knowledge distillation," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2023.

Audio Engineering Society

Conference Paper 30

Presented at the AES International Conference on Machine Learning and Artificial Intelligence for Audio
2025 September 8–10, London, UK

This paper was peer-reviewed as a complete manuscript for presentation at this conference. This paper is available in the AES E-Library (http://www.aes.org/e-lib), all rights reserved. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Journal of the Audio Engineering Society.

Perceptual Evaluation of Machine Learning and Non-ML Emulations of the Vox AC30 Amplifier

Mario Alberto Vallejo Reyes[1], Gavin Kearney[1], and Michael McLoughlin[1]

[1]*School of Physics, Engineering and Technology, University of York*

Correspondence should be addressed to Mario Alberto Vallejo Reyes (mario.vallejo@york.ac.uk)

ABSTRACT

Twenty-two listeners evaluated the perceptual similarity of three commercially available emulations of a Vox AC30 amplifier to the original hardware reference under controlled conditions: (1) a machine-learning plugin trained on the exact test amplifier, (2-3) two non-ML plugins modeling unspecified AC30 units. Using MUSHRA methodology, results showed the unit-specific ML plugin achieved perceptual indistinguishability from the hardware reference in 1/3 cases and outperformed both non-ML alternatives in 5/6 pairwise comparisons. This demonstrates that while non-ML plugins necessarily generalize across hardware units, ML techniques can achieve high perceptual fidelity when trained on target amplifiers. This represents a previously unattainable capability with significant implications for audio preservation and production.

1 Introduction

Over time, digital technology has progressively shaped guitar amplifier emulation, with increasingly sophisticated techniques emerging to capture the nuanced, nonlinear behaviors of analog amplifiers [1]. As studio environments grow more diverse and flexible [2], offering digital versatility while preserving cherished hardware characteristics has become crucial.

Recent innovations include the application of machine learning (ML) in amplifier emulation, a shift that promises new levels of authenticity compared to conventional digital signal processing (DSP). However, skepticism persists in the broader audio community: a survey of 1,500 participants by Bedroom Producers Blog [3] reflects limited enthusiasm for AI tools in music production. This intersection of technological advancement and user acceptance underscores that in the field of music production and audio engineering, accurately reproducing and quantifying sound quality remains a key focus of technological development [4]. While objective measurements, such as numerical measures of interference, noise, and artifacts, are indispensable, subjective listening tests remain critical in determining perceptual quality [5].

The present study evaluates perceptual similarities between ML-based and non-ML amplifier emulations relative to a genuine Vox AC30 amplifier through listening tests following the MUSHRA protocol (Multi-Stimulus Test with Hidden Reference and Anchor). The goal is to offer empirical data on these comparisons. While existing research has focused on experimental or prototype ML emulations, this study evaluates commercially available plugins, including a machine learning-based

model (Neural Amp Modeler [6]) and two widely used non-ML emulation tools: Waves GTR3 (Version 14.12 used in this study, released in 2022 [7]; originally released in 2007 [8]) and IK Multimedia AmpliTube 5 (Version 5.7.5 used in this study, released in 2020 [9]; originally released in 2002 [10]). These non-ML plugins remain relevant benchmarks due to their widespread use in contemporary workflows. Ultimately, the findings may contribute to ongoing discussions within the music industry regarding the role and acceptance of AI-based tools in production workflows.

1.1 Previous Findings

Pakarinen and Yeh [1] highlight that the emulation of vacuum-tube guitar amplifiers has long been a challenging endeavor due to the complex nonlinearities and dynamic behaviors inherent in their design. According to their work, traditional approaches, such as analog solid-state emulations and digital signal processing (DSP) techniques, have sought to replicate the distinctive tonal characteristics of tube amplifiers. Pakarinen and Yeh explain that these methods often rely on static waveshaping, oversampling, and circuit simulation to approximate the amplifier's response. However, they note that accurately capturing the intricate interdependencies and dynamic effects, such as grid capacitor charging, cathode bypass behavior, and power amplifier sagging, remains a significant challenge. While DSP-based emulations offer flexibility and efficiency, enabling real-time modeling and integration into software plugins, Pakarinen and Yeh emphasize that the complex dynamical nonlinearities of the tube-amplifier circuit make accurate real-time simulation extremely demanding. Building on the foundational insights provided by Pakarinen and Yeh, the discussion progresses toward exploring more advanced techniques, such as machine learning (ML), which aim to leverage data-driven approaches to model the complex behaviors of tube amplifiers more effectively.

Zhang et al. [11] investigated modeling vacuum-tube guitar amplifiers using LSTM networks, focusing on replicating tube-driven distortion and tonal characteristics. Their approach involved exploring various hyperparameter configurations and assessing model performance through objective metrics and casual feedback from semi-professional guitarists. Although their method approximated tube amplifier responses, guitarist evaluations indicated perceptual differences that still require refinement.

Wright et al. [12] demonstrated the feasibility of real-time emulation by comparing WaveNet-inspired and recurrent neural network architectures. They trained models using only a few minutes of audio data and evaluated them through subjective listening tests. The results showed that while some models still deviated perceptually from the reference signals, others were virtually indistinguishable, underscoring the promise of ML methods for efficient, perceptually convincing emulations.

Juvela et al. [13] proposed an automated, data-driven framework using an LSTM network conditioned on amplifier control parameters. This approach aimed to capture the amplifier's full control range and dynamics. Their perceptual tests, involving expert listeners, found that the proposed model achieved subjective quality comparable to that of a white-box Simulation Program with Integrated Circuit Emphasis (SPICE)[14] simulation. SPICE, an industry-standard tool for electrical circuit simulation, requires significant expert effort and computational resources, often taking hours to render audio samples. In contrast, the LSTM-based model achieved comparable perceptual quality while maintaining real-time computational efficiency.

Collectively, these studies highlight the growing sophistication of ML-based amplifier modeling. However, perceptual discrepancies remain, Zhang et al. [11] noted that guitarists still heard differences in emulated tube responses, and Wright et al. [12] found certain models deviated perceptually from reference signals. Nevertheless, these studies indicate that ML-based modeling continues to improve in realism, flexibility, and efficiency.

2 Methods

2.1 Stimuli Creation

The dataset consists of three 12-second clean guitar recordings from a Fender Stratocaster, recorded through a Universal Audio 76 interface at 48 kHz/24-bit. All performances were by the same musician using identical equipment and settings, with each sample featuring rhythm guitar playing simple pop-rock chord progressions that differ only in the specific chords and tempo. The recordings adhere to MUSHRA guidelines for stable audio content [15].

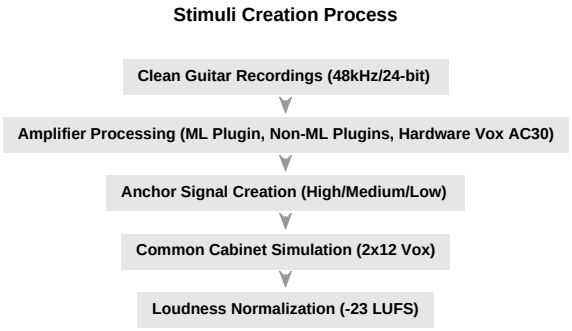


Fig. 1: Flowchart of the stimuli creation process.

Neural Amp, a machine learning plugin developed by Steve Atkinson, was used to profile a Vox AC30 amplifier. The profiling process involved playing Neural Amp's provided test signal through the amplifier's High Input Top Boost channel at 48kHz/24-bit, recorded at -18 dBFS. The signal chain included a Focusrite interface, Radial Engineering re-amp box for impedance matching, and a Torpedo Captor 16 load box to attenuate the amplifier's speaker output signal. This setup bypassed the built-in speaker, ensuring a pure recording of the amplifier's output before the cabinet. All amplifier controls were standardized at midpoints with time-based effects disabled. The recorded output was processed through Neural Amp's Google Colab environment to generate the amplifier profile (.NAM file). The final samples were created in Ableton Live 12 [16] (chosen for VST/AU compatibility) using the Neural Amp plugin with the generated profile.

IK Multimedia Amplitube and Waves GTR plugins were used to create non-machine learning samples using their Vox AC30 emulations. Both plugins were configured in Pro Tools [17] (48kHz/24-bit) with controls set to midpoints, time-based effects disabled, and cabinet simulation bypassed to match the configuration of the real amplifier recording. Neither plugin specifies whether it emulates the high gain or low gain input section of the amplifier. The clean guitar samples were processed through both emulations and exported for subsequent cabinet processing and loudness normalization.

To create the hardware Vox AC30 samples, the clean guitar tracks were processed through the amplifier using a 48kHz/24-bit Pro Tools session. The signal chain included a Focusrite interface, Radial Engineering re-amp box for impedance matching, and Torpedo Captor 16 load box for signal attenuation. Signals were routed to the amplifier's High Input Top Boost channel with all controls set to midpoints and time-based effects disabled. The processed audio was recorded back into Pro Tools and exported for subsequent cabinet processing and normalization.

2.1.1 Creating the Anchor Signals

MUSHRA methodology requires high, medium and low-quality anchor signals to validate participant responses. The high quality anchor is the same as the reference file (hardware Vox AC30). The standard medium and low quality anchors are created using 7 kHz (medium-quality) and 3.5 kHz (low-quality) high-cut filters. However, these standard cutoff frequencies produced no perceptible quality degradation in guitar amplifier evaluation due to their limited high-frequency content above 5 kHz. Therefore, the standard octave reductions from 20 kHz (1.5 and 2.5 octaves) were doubled, resulting in new cutoff frequencies of 2466 Hz for medium-quality and 616 Hz for low-quality anchors. These modified frequencies created perceptually distinct quality levels appropriate for guitar amplifier evaluation.

2.2 Stimuli Processing

To ensure fair amplifier comparisons during listening tests, all signals were recorded directly from amplifier outputs without cabinets to focus solely on amplifier characteristics. A consistent cabinet simulation—the '2_12InchOpenBack' model from Waves GTR, based on a Vox AC30 Celestion Alnico Blue—was applied to standardize output. This process involved routing all signals through the cabinet emulation in Pro Tools, exporting the processed files as WAVs, and normalizing them in MATLAB.

Normalization followed the EBU R128 standard [18], with integrated loudness and gain adjustments implemented in MATLAB using MathWorks' recommended methods [19]. All files were adjusted to -23 LUFS. Subjective evaluations by the researchers confirmed consistent loudness, reducing bias and allowing participants to focus on tonal differences among emulations.

To exemplify both the spectral differences between the hardware amplifier and its emulations and between tasks, Figure 2 shows a 1-second-excerpt spectrogram of the hardware amplifier alongside its emulations, and

Figure 3 shows 1-second-excerpt spectrograms illustrating spectral differences between tasks for the ML stimuli.

2.3 MUSHRA Listening Test

This study employed GoListen [20], a web-based platform to implement the MUSHRA listening test methodology. The test consisted of three evaluation tasks. Each task comprised seven audio files: a reference (hardware Vox AC30), the machine learning emulation, two non-machine learning emulations, and three anchor signals. The anchors included a high-quality anchor (an audio file identical to the hardware reference), as well as medium quality and low quality anchors created using high-cut filters.

Participants were provided with bilingual instructions in English and Spanish, guiding them to rate each sample's similarity to the reference on a scale from 0 (no similarity at all) to 100 (highly similar).

The platform randomized the order of stimuli for each participant to minimize potential bias from fixed sequences. Brief supplementary questions gathered information about participants' backgrounds and listening equipment.

2.4 Statistical Analysis of Perceptual Differences

Only responses that met predefined inclusion criteria were retained for analysis: participants had to rate File 04 (the identical-to-reference anchor) as the highest or second-highest, and File 05 (the low-quality anchor) as the lowest. This ensured attentiveness and allowed for cases where some emulations were perceptually indistinguishable from the reference.

Descriptive statistics (medians and interquartile ranges, 25th to 75th percentiles) were calculated to summarize central tendency and dispersion. These were visualized with violin plots to support intuitive comparison of rating distributions across conditions.

To examine perceptual differences across the three emulation types (ML, Non-ML A, Non-ML B), participants' ratings were compared using either parametric or non-parametric methods, depending on the distribution and variance characteristics of the data.

Normality of rating distributions within each emulation group was assessed using the Shapiro–Wilk test ($\alpha = 0.05$). Homogeneity of variances across groups was evaluated using Levene's test. When variances were equal (Levene's $p > 0.05$), one-way ANOVA was used for group comparison, even when one group showed mild non-normality, due to ANOVA's robustness with balanced group sizes. When both assumptions were violated, the Kruskal–Wallis test[21] was used as a non-parametric alternative.

Post-hoc comparisons were performed when significant group differences were found: Tukey's HSD followed ANOVA, and Dunn's procedure (with multiple comparison correction) followed Kruskal–Wallis. For non-significant results ($p > 0.05$), Bayesian t-tests with Cauchy priors (scale = 0.707) were conducted using raw scores for normally distributed data and rank-transformed data otherwise, to quantify evidence strength for null effects.

3 Participants

3.1 Participants Background

The study included 22 participants: 5 university lecturers in audio engineering/music technology, 7 professional sound engineers/music producers, 6 students in music technology or audio engineering, and 4 early-career individuals. Individual experience ranged from newcomers to over 30 years. Participants' areas of expertise included mixing, recording, production, research, and live sound engineering.

3.1.1 Participants Equipment

Participants completed listening tests either in-person at the University of York (AudioLab and Music Research Centre) or online. Both formats followed identical procedures, differing only in initial instruction delivery: the researcher provided an introductory briefing for in-person sessions, while online participants accessed written instructions.

In-lab tests utilized a Mac computer (MacBook Pro 13", 2022), a Focusrite Scarlett 18i20 audio interface (2nd Gen), and open-back Sennheiser HD 650 headphones. Online participants were required to use studio-grade monitoring equipment. The specific audio equipment used by participants is summarized in Table 1.

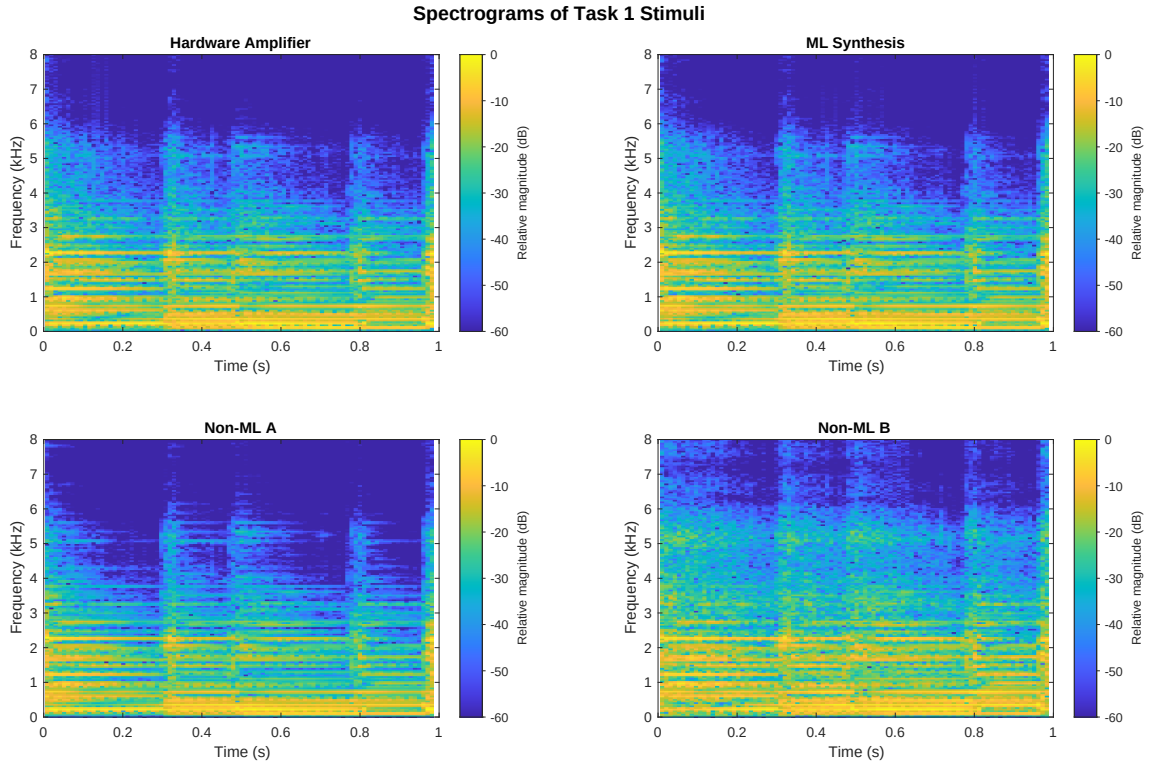


Fig. 2: 1-second spectrograms for Task 1 stimuli: Hardware Vox AC30, ML emulation, non-ML A, non-ML B.

Table 1: Summary of audio equipment used by participants

Equipment	Type
ADAM Audio S2A	Speakers
ADAM Audio T5V	Speakers
ADAM Audio T7V	Speakers
AKG K52	Headphones
beyerdynamic DT770 80OHM	Headphones
Genelec 4020C	Speakers
JBL 305P MKII	Speakers
Sennheiser HD650	Headphones
Yamaha HS8 w/ADAM Audio Subwoofer	Speakers

4 Results

Tables 2, 3, and 4 summarize the median similarity ratings and interquartile ranges (25th–75th percentiles) for Tasks 1, 2, and 3, respectively. The corresponding violin plots, which display the rating distributions (kernel-density outline) together with the median, interquartile range, and minimum/maximum values, are shown in Figures 4, 5, and 6.

Significant differences in perceived similarity among emulation conditions were observed for all three tasks. ANOVA results showed statistically significant group differences for Task 1 ($F(5,96) = 69.28$, $p < 0.001$) and Task 3 ($F(5,90) = 44.45$, $p < 0.001$). For Task 2, assumptions of normality (Shapiro-Wilk, Non-ML B: $p = 0.013$) and homogeneity of variance (Levene's test: $p = 0.015$) were violated, and the Kruskal-Wallis test was performed, which confirmed significant differences ($\chi^2(2) = 16.28$, $p < 0.001$).

Post-hoc pairwise comparisons (Tukey HSD for Tasks 1 and 3, Dunn's test for Task 2) are summarized in Tables 5, 6, and 7. These tables report the mean differences, confidence intervals, and exact p-values for each pairwise contrast.

Additionally, Table 8 presents the outcomes of Bayesian t-tests for pairwise comparisons that did not reach significance in the frequentist analysis, offering complementary evidence based on Bayes Factors.

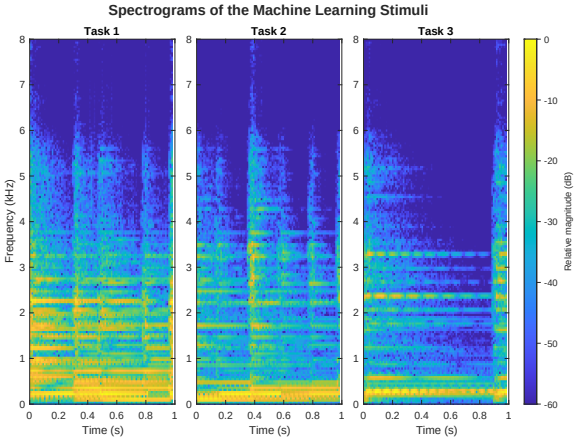


Fig. 3: 1-second-excerpt spectrograms of the machine-learning–generated guitar stimuli after amplifier and cabinet processing and loudness normalization.

Table 2: Task 1 Median Similarity Ratings and Interquartile Ranges (25th–75th percentile)

File (Condition)	Task 1
File 01 (Machine Learning)	100 (88.25–100)
File 04 (Anchor: High Quality)	96 (93.75–100)
File 02 (Non-ML A: Amplitube)	71 (62.25–80.25)
File 06 (Anchor: Medium Quality)	42 (15.75–60)
File 03 (Non-ML B: Waves)	36 (18.75–46.25)
File 05 (Anchor: Low Quality)	0 (0–20)

Table 3: Task 2 Median Similarity Ratings and Interquartile Ranges (25th–75th percentile)

File (Condition)	Task 2
File 04 (Anchor: High Quality)	99 (95.75–100)
File 03 (Non-ML B: Waves)	95 (85.25–100)
File 01 (Machine Learning)	85 (59.75–92)
File 02 (Non-ML A: Amplitube)	71 (55.75–82)
File 06 (Anchor: Medium Quality)	46 (30.75–50)
File 05 (Anchor: Low Quality)	0 (0–16.25)

5 Discussion

5.1 Interpretation of Results: Task 1

Parametric assumptions were satisfied, and a one-way ANOVA confirmed an overall effect of condition on similarity ratings, $F(5,96) = 69.28,\ p < 0.001$. The machine-learning (ML) emulation received the highest median similarity score (100), slightly above the median for the high-quality anchor (File 04, identical to

Table 4: Task 3 Median Similarity Ratings and Interquartile Ranges (25th–75th percentile)

File (Condition)	Task 3
File 04 (Anchor: High Quality)	100 (95.5–100)
File 01 (Machine Learning)	82 (66.5–90)
File 02 (Non-ML A: Amplitube)	74 (57.5–81.5)
File 06 (Anchor: Medium Quality)	48 (22–66)
File 03 (Non-ML B: Waves)	41.5 (22.5–58.5)
File 05 (Anchor: Low Quality)	0 (0–21.5)

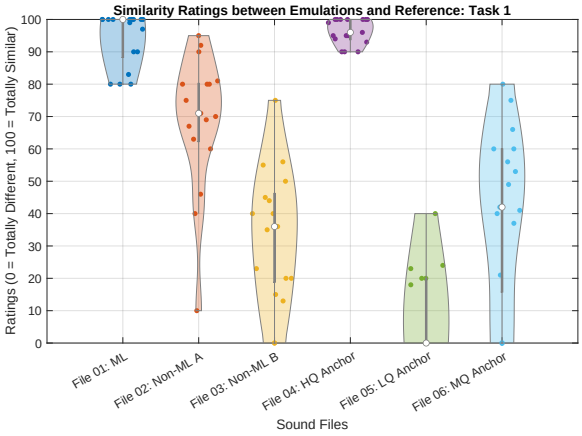


Fig. 4: Similarity ratings of emulations to the reference in Task 1.

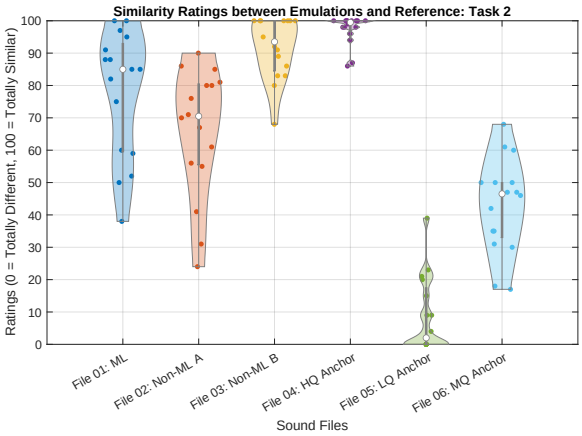


Fig. 5: Similarity ratings of emulations to the reference in Task 2.

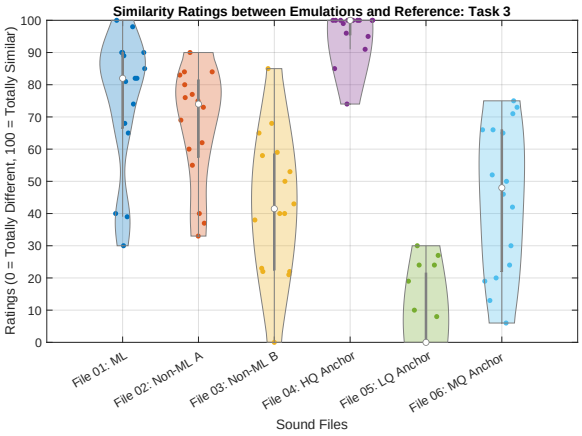


Fig. 6: Similarity ratings of emulations to the reference in Task 3.

Contrast	MeanDiff	95% CI	p-value
ML vs Non-ML A	21.33	[10.22, 32.44]	< 0.001
ML vs Non-ML B	57.67	[42.12, 73.21]	< 0.001
ML vs HQ Anchor	3.67	[-11.22, 3.89]	0.616
Non-ML A vs Non-ML B	36.33	[17.40, 55.27]	< 0.001

Table 5: Task 1: Tukey HSD pairwise contrasts ($\alpha = 0.05$).

Contrast	MeanDiff	95% CI	p-value
ML vs Non-ML A	12.69	[–16.06, 41.43]	0.962
ML vs Non-ML B	–15.78	[–44.52, 12.96]	0.820
ML vs HQ Anchor	–23.44	[–52.18, 5.31]	0.226
Non-ML A vs Non-ML B	–28.47	[–57.29, 0.35]	0.055

Table 6: Task 2: Dunn–Sidak post hoc pairwise comparisons ($\alpha = 0.05$).

Contrast	MeanDiff	95% CI	p-value
ML vs Non-ML A	7.81	[–0.94, 16.57]	0.094
ML vs Non-ML B	32.25	[8.79, 55.71]	0.005
ML vs HQ Anchor	–21.06	[–38.35, –3.77]	0.013
Non-ML A vs Non-ML B	24.44	[4.27, 44.61]	0.014

Table 7: Task 3: Tukey HSD pairwise contrasts ($\alpha = 0.05$).

the hardware Vox AC30) (96). Post-hoc Tukey tests, however, detected no statistically reliable difference between these two conditions ($p = 0.616$, $95\% CI = [-11.22, 3.89]$). This outcome indicates that listeners treated the ML emulation and the high-quality anchor as perceptually indistinguishable; the small numerical advantage for ML is consistent with random rating

Table 8: Bayesian evidence for selected pairwise comparisons ($p > 0.05$)

Comparison	(BF_{10})	Interpretation
Task 2: ML vs HQ Anchor	>100	Decisive evidence for Anchor superiority
Task 2: ML vs Non-ML A	2.98	Anecdotal evidence for ML superiority
Task 2: ML vs Non-ML B	2.00	Anecdotal evidence for Non-ML B superiority
Task 3: ML vs Non-ML A	5.05	Substantial evidence for ML superiority

variability when two stimuli are nearly identical. In contrast, ML received statistically significantly higher similarity ratings than non-ML A (median 71) and non-ML B (median 36), each $p < 0.001$. Overall, in Task 1 the ML emulation was perceptually indistinguishable from the high-quality anchor while exhibiting statistically significantly higher similarity ratings than either non-ML emulation.

5.2 Interpretation of Results: Task 2

The Kruskal-Wallis test revealed significant differences among conditions ($\chi^2(2) = 16.28$, $p < 0.001$). Dunn-Sidak post-hoc tests showed the ML emulation (median = 85) was not significantly different from the high-quality anchor (median = 99; $p = 0.226$), though Bayesian analysis of ranks decisively supported the anchor's superiority. ML showed no significant difference from Non-ML A (medians = 85 vs 71; $p = 0.962$), with Bayesian results suggesting a weak trend favoring ML. For ML versus Non-ML B (medians = 85 vs 95; $p = 0.820$), no significant difference was observed, but Bayesian analysis indicated a weak trend favoring Non-ML B.

Overall, in Task 2 ML was distinguishable from the hardware reference through Bayesian analysis, non-significantly outperformed Non-ML A but with Bayesian support, and showed weak Bayesian evidence for inferiority to Non-ML B.

5.3 Interpretation of Results: Task 3

Parametric assumptions were met, and a one-way ANOVA yielded $F(5,90) = 44.45$, $p < 0.001$. Tukey tests indicated that, in terms of similarity to the hardware Vox AC30, ML was statistically significantly less similar than the high-quality anchor, with medians of 82 and 100, respectively ($p = 0.013$). ML was statistically significantly more similar than non-ML B ($p = 0.005$; medians 82 vs 41.5). The difference between ML and non-ML A (medians 82 vs 74) did

not reach significance ($p = 0.094$); a Bayes factor of $BF_{10} = 5.05$ nevertheless provided substantial evidence favouring an ML advantage. Overall, in Task 3 ML was distinguishable from the hardware amplifier, statistically significantly outperformed non-ML B, and showed a Bayesian-supported but non-significant advantage over non-ML A.

5.4 Across-Excerpt Synthesis

Comparing ML to the high-quality anchor, ML was indistinguishable in one of three tasks and distinguishable in the other two. In Task 2, the high-quality anchor's higher similarity ratings were not statistically significant, but Bayesian analysis decisively supported this advantage. In Task 3, the high-quality anchor's similarity ratings were significantly higher. Across the six ML–non-ML pairwise similarity comparisons to the hardware Vox AC30, ML outperformed in five cases: three were statistically significant in favor of ML, two were non-significant but had Bayesian support for ML, and one was non-significant but Bayesian-supported in favor of the non-ML system (the ML B comparison).

It should be noted that the ML model was profiled on the specific Vox AC30 unit used as the reference in the listening test, whereas the non-ML emulations were modeled after different units of the same amplifier model. This introduced an advantage for the ML condition, though its high similarity to the profiled hardware still provides a valid basis for assessing perceptual fidelity.

5.5 Listener Expertise and Perceptual Ranking Trends

To assess whether listener expertise influenced rating trends, mean similarity scores were analyzed separately for experts and non-experts whose responses met the validity criteria. Although no normality or post-hoc tests were applied due to small group sizes, clear patterns emerged. In Task 1, both groups identified the same top three conditions, but their order differed: experts ranked ML first, followed by the reference and Non-ML A; non-experts placed the reference first, followed by ML and Non-ML A. In Tasks 2 and 3, both groups showed identical rankings. This convergence suggests that listener experience had minimal effect on overall perceptual trends.

6 Conclusion

This study evaluated the perceptual similarity of a commercially available machine-learning (ML) guitar amplifier emulator against two non-ML alternatives and three quality anchors, using the hardware Vox AC30 as reference.

In Task 1, ML was statistically indistinguishable from the high-quality anchor and significantly closer to the reference than both non-ML alternatives. In Task 2, ML was rated less similar than the high-quality anchor; the gap was non-significant, but Bayesian analysis favored the anchor. ML held a non-significant, Bayes-supported edge over non-ML A, while the ML-B comparison was non-significant with Bayesian support for the non-ML system. In Task 3, ML was again significantly less similar than the high-quality anchor, significantly outperformed non-ML B, and showed a non-significant, Bayes-supported advantage over non-ML A. Overall, ML exceeded non-ML emulations in five of six pairwise comparisons, either significantly or with Bayesian support.

Task 2 failed the Shapiro-Wilk normality test for the Non-ML B condition and Levene's test for homogeneity of variance across the three emulation groups. The emulation rating distributions for Task 2 exhibit pronounced skewness, indicating a non-normal distribution of responses. This skewness may arise from the relatively small number of listeners or from intrinsic characteristics of the audio stimulus, such as its tempo, dynamic range, or tonal content. Studies with larger cohorts and a wider variety of clips will help determine which factor predominates.

These results extend prior research—such as Juvela et al.,[13], Wright et al.,[12], and Zhang et al.,[11]—by demonstrating that modern ML-based emulations, when profiled on the test amplifier, can achieve perceptual indistinguishability from the reference unit and generally outperform traditional non-ML approaches modeling different instances of the same amplifier design.

In practical terms, ML-based emulation emerges as a robust option for applications requiring high perceptual fidelity, although its performance still varies with programme material. Future work should include more musical content and apply this evaluation approach to other emulated devices, such as microphones or recording chains, to test generalizability.

References

[1] Pakarinen, J. and Yeh, D. T., "A Review of Digital Techniques for Modeling Vacuum-Tube Guitar Amplifiers," *Computer Music Journal*, 33(2), pp. 85–100, 2009, doi:10.1162/comj.2009.33.2.85.

[2] "The End of the World as We Know It: The Changing Role of the Studio in the Age of the Internet," in *The Art of Record Production*, pp. 98–112, Routledge, 2012, ISBN 9781409405627.

[3] Bedroom Producers Blog, "AI Music Survey: How 1,500 Music Producers Use AI For Music Production," https://bedroomproducersblog.com/2023/05/30/ai-music-survey, 2023, accessed: 2024-05-24.

[4] Bech, S. and Zacharov, N., *Perceptual Audio Evaluation: Theory, Method and Application*, Wiley, 1. aufl. edition, 2007, ISBN 9780470869239.

[5] Kornycky, J., Gunel, B., and Kondoz, A., "Comparison of Subjective and Objective Evaluation Methods for Audio Source Separation," *The Journal of the Acoustical Society of America*, 123(5 Supplement), pp. 3569–3569, 2008.

[6] Atkinson, S., "Neural Amp Modeler," https://www.neuralampmodeler.com, 2024, accessed: 2024-03-12.

[7] Waves Audio, "Waves V14 Release Notes," https://www.waves.com/downloads/release-notesv14, 2022, accessed: 2025-07-03.

[8] MusicRadar, "New amps in Waves GTR3," https://www.musicradar.com/news/guitars/new-amps-in-waves-gtr3-33152, 2007, accessed: 2025-07-03.

[9] MusicRadar, "It's lift-off for AmpliTube 5 as IK Multimedia unveils the next edition of its flagship guitar-software package," https://www.musicradar.com/news/its-lift-off-for-amplitube-5-as-ik-multimedia-unveils-the-next-edition-of-its-flagship-guitar-software-package, 2020, accessed: 2025-07-03.

[10] IK Multimedia, "About Us — Company Timeline," https://www.ikmultimedia.com/about-us/, 2002, accessed: 2025-07-03.

[11] Zhang, Z., Olbrych, E., Bruchalski, J., McCormick, T. J., and Livingston, D. L., "A Vacuum-Tube Guitar Amplifier Model Using Long/Short-Term Memory Networks," in *SoutheastCon 2018*, pp. 1–5, IEEE, 2018, ISBN 1538661330.

[12] Wright, A., Damskägg, E.-P., Juvela, L., and Välimäki, V., "Real-Time Guitar Amplifier Emulation with Deep Learning," *Applied Sciences*, 10(3), pp. 766–, 2020.

[13] Juvela, L., Damskagg, E.-P., Peussa, A., Makinen, J., Sherson, T., Mimilakis, S. I., Rauhanen, K., and Gotsopoulos, A., "End-to-End Amp Modeling: From Data to Controllable Guitar Amplifier Models," in *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5, IEEE, 2023, ISBN 1728163277.

[14] Nagel, L. W. and Pederson, D. O., "SPICE (Simulation Program with Integrated Circuit Emphasis)," Technical Report UCB/ERL M382, University of California, Berkeley, EECS Department, 1973.

[15] International Telecommunication Union, "Recommendation ITU-R BS.1534-1: Method for the subjective assessment of intermediate quality level of coding systems," Technical report, International Telecommunication Union, 2003, accessed: 2024-03-12.

[16] Ableton, "Ableton Live," https://www.ableton.com/en/live, 2025, accessed: 2025-03-13.

[17] Avid Technology, "Pro Tools," https://www.avid.com/pro-tools, 2025, accessed: 2025-03-13.

[18] EBU, "128, Loudness Normalisation and Permitted Maximum Level of Audio Signals," *EBU Recommendation, Geneva*, 2014.

[19] MathWorks, "Loudness Normalization in Accordance with EBU R 128 Standard," https://uk.mathworks.com/help/audio/ug/loudness-normalization-in-accordance-with-ebu-r-128-standard.html, 2024, accessed: 2024-03-12.

[20] Barry, D., Zhang, Q., Sun, P. W., and Hines, A., "Go Listen: An End-to-End Online Listening Test Platform," *Journal of Open Research Software*, 9(1), pp. 20–, 2021.

[21] Kruskal, W. H. and Wallis, W. A., "Use of Ranks in One-Criterion Variance Analysis," *Journal of the American Statistical Association*, 47(260), pp. 583–621, 1952.

Audio Engineering Society

Conference Paper 31

Presented at the AES International Conference on Machine Learning and Artificial Intelligence for Audio
2025 September 8–10, London, UK

This paper was peer-reviewed as a complete manuscript for presentation at this conference. This paper is available in the AES E-Library (http://www.aes.org/e-lib), all rights reserved. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Journal of the Audio Engineering Society.

Broadcast-quality synthetic narration: a workflow for fine-grained text-to-speech intonation and emotion control

Luiz F. Kruszielski[1], Pedro H. L. Leite[1,2], Myllene P. Fernandes[1], André Pereira[1], and Luiz W. P. Biscainho[2]

[1]*Globo Group*
[2]*Federal University of Rio de Janeiro*

Correspondence should be addressed to Luiz F. Kruszielski (luiz.fk@g.globo)

ABSTRACT

The ability to generate high-quality synthetic speech with fine-grained control over intonation and emotion is crucial for broadcast applications such as narration and voice-over production. This paper presents a novel text-to-speech (TTS) workflow designed to generate expressive and user-controllable speech synthesis. By leveraging a one-hot encoded emotional embedding approach through application-aware, manually labeled emotion classes and a fine-tuned neural model, our method allows for precise control over speech tone and intention. We detail the data preparation pipeline, including manual annotation and automatic emotion classification, and describe the implementation of our speech synthesis model using VITS2. Additionally, we introduce an intuitive prosody control mechanism that enables word- and phoneme-level adjustments for pitch and duration. Subjective evaluation results indicate that our synthetic narration achieves parity with professionally recorded voice-overs, with some preference for synthetic speech due to its enhanced customization. This research contributes to the advancement of AI-driven voice production, enabling scalable, high-fidelity, and fine-grained controlled speech synthesis for media and entertainment industries.

1 Introduction

Speech is one of the most primal and indispensable forms of direct transfer of complex information between humans. Natural speech does not only carry information through the semantics of the words being expressed; tone and intonation are just as crucial to conveying information. The tone intention of the speech can change the semantic interpretation of the words; for instance, the meaning of a word can be reversed if a sarcastic tone is used. Furthermore, the tone and emotions in natural speech can vary widely in a single sentence. For this reason, producing emotionally diverse speech samples has recently been a hot topic for general speech generation (e.g. [1, 2, 3, 4, 5, 6, 7]).

A common way to create natural emotion and intonation is to use only the semantics of the text to direct the response in tone of voice. However, this approach does not offer flexibility over the same text input. To avoid this issue, we focus on text-to-speech (TTS) with parallel and user-customized emotional and tone input in this work, allowing for the representation of different tones and intonations for the same text prompt. With parallel emotion information, we may control the output speech prosody to input user-based semantic intention, ranging

from lively and enthusiastic speech to neutral, sad or sober.

The established approach to intentionally customize speech style in TTS is to generate a high-dimensional latent space generally represented by embeddings [8] that encode the desired features into vectors, which will serve as input to the neural layers together with or independently of the text [9, 10, 11, 12, 2, 3, 4, 13]. For example, [2] explores pre-trained embedding spaces. In contrast, [3] generates the embeddings through text content, [4] proposes a specific algorithm to extract an embedding space from a previously labeled dataset, and [14] uses an intuitive prosodic encoder to drive speech features utterance-wise. Although these approaches have proven to be easily generalizable, they do not account for application-specific intonations and may fail when the inference domain may interfere with emotional overlapping, and/or a more fine-grained adjustment is required.

From a production/industrial perspective, a particular speech tone and a high voice quality standard are required to create broadcast narrations and voice-overs. Furthermore, voices must connect with the audience, expressing the necessary emotion and intonation to convey the desired message. Any perception by the audience of something "strange" or "unnatural" in a voice can break trust and disconnect from the immersive diegesis of the audiovisual content.

Due to the advances in complex models with well-validated training architectures in that environment, we argue that it may be better to let neural networks produce an embedding space directly through application-aware and perceptually labeled data. Also, employing "one-hot" generated embeddings can leverage the quality of the generated space by reducing ambiguities, since the generating vectors are orthogonally representing semantic features. This approach is beneficial when implementing TTS systems in practice, as it simplifies the inference pipelines while maintaining controllability and operability, even more so when applied together with fine-grained prosody control for eventual adjustments.

Conversely, languages like Brazilian Portuguese (the target language of our models, as seen further) often lack sufficiently large and high-quality datasets suitable for production-grade applications and high-quality demands for speech [15]. This creates a pressing need for optimized data pipelines and algorithms that are explicitly tailored to the linguistic characteristics of such underrepresented languages.

We thus propose a complete workflow for preparing data, generating classes and embeddings, training domain-specific and language-specific models, and creating robust and easy-to-use application interfaces to enable a TTS model scale that can deliver content for millions of users daily.

2 Emotional Embeddings and one-hot encoding vectors

Pre-trained or generalistic embedding networks can be adapted for unseen emotions, and generate meaningful latent spaces from realistic distributions. However, this approach faces some problems when dealing with application-aware emotions. Although some speech-emotional mapping has already been developed (e.g. [16]), it is not straightforward to use pre-defined emotional dimensions when employing TTS systems in production-ready applications, because generally this requires a level of emotional fine-tuning. Furthermore, encoding speech into those dimensions is also a challenge, usually requiring complex and data-demanding pre-training stages or the adaptation of generalist embeddings, which may bring other undesired components into the resulting latent dimensions. In this work, we propose to mitigate these issues by using one-hot encoding vectors to encode emotions, which may be a better choice when emotion classes can be well defined beforehand and coupled with manually labeled data with application context being taken into account.

When using this one-hot encoding approach, the networks may learn the intrinsic emotional dimensions without needing an early latent space design, which is simpler and less contaminative. The downside may be that this learned space is not entirely focused on emotion, but rather on a generalist speech style as seen in [9]; however, when this style is driven from manually labeled data, this contamination can be controlled by adequately generating label classes for emotions and guaranteeing intraclass semantic coherence.

We kept the one-hot embedding approach as our design choice for these reasons. Also, as the use case involves emotion classes from application employments, orthogonally separating the generating vectors should ensure that the intention that drives the embedding generation is the same as when defining the classes, maintaining application-defined requirements until inference stages.

3 Data conditioning and preparation

We used approximately 5 years of archived promo content voice recordings for the dataset corpus, mainly consisting of 30s commercial inserts, recorded by the same narrator. To address TTS applications, general data is needed for prepared speech with corresponding transcripts, and the emotional, language, and accent context is required at inference time. Thus, these recordings were cut into smaller segments of at least four and at most 15 seconds, following silences larger than 0.5 seconds detected in the original audio. Silence and noise-only audio signals have been discarded, resulting in 25800 short sentences. The Python Pydub library[1] was used for this first processing step. The next task was selecting the intentions and emotions of interest, considering their correspondence with the available data and intended application. The subsequent annotation procedure is based on the classes thus defined. Since the original use of the available audio content is to promote various programs, including sports, several film genres, soap operas, and news, it allows for a wide range of different emotions and intonations. All sentences were automatically transcribed using the Whisper [17] library[2], and later manually reviewed for eventual errors.

3.1 Emotion and intonation Classes

The emotion classes were defined based on content and potential application for a synthetic narrator voice. Several distinct states of excitement were established, ranging from a "very excited" tone (e.g., in promotions such as soccer matches) to a "neutral" tone (for instance, during technical network maintenance announcements), including "serious" (e.g., for action movie promotions) and "sad/down" tones. In addition to these emotion-related categories, other non-emotional categories were also defined. The "signature" category captures the narrator's unique way of announcing the broadcast program's name. In contrast, the "date/signature" category is used when the narrator specifies the month or day of the week that the program will air. Because interrogative sentences appeared less frequently in the primary dataset —– resulting in challenges for properly rendering a questioning tone —– a dedicated "question" category was created. This category is the only one that contains duplicated examples to help balance the dataset. Furthermore, a "slate" category was introduced for the introductory words in each recording section, where the narrator announces the date and the targeted program using a "non-narrator" voice. Lastly, a "discarded" category was created for recordings with poor quality or excessive noise; this category was not included in the training dataset. The dataset comprises 11 categories, 10 of which are used for voice synthesis. In Table 1, we show the final class choices and a brief description with text examples representing the samples within them.

[1] https://github.com/jiaaro/pydub

[2] https://github.com/openai/whisper

3.2 Language specific tokenizer

For Brazilian Portuguese (in short, Pt-Br) speech, some text-speech ambiguities (such as contextual and/or syntax-dependent vowel openness and nasalization) should be resolved beforehand, reducing the need for rule-related learning by the neural networks. Therefore, we introduced a stress-dependent phoneme tokenizer in the text processing layers, transcribing the written text into IPA [18] representations (unique text-phoneme mappings) and adapting stress marked phonemes into distinguishable tokens.

While we targeted a language-specific problem with this tokenizer, this kind of adaptation will usually be needed for languages other than English or Mandarin, which are the languages these models have generally been designed for.

3.3 Class predictor

After the classes had been defined and we had manually labeled around a fifth of the dataset (5400 samples totaling around 12 h), we trained an automatic class predictor to generate labels accordingly, for the remaining dataset. The classifier was structured as a generalist pretrained feature embedding generator followed by a small feedforward network adapted to minimize the cross-entropy for the manually labeled data. The generalistic encoder used was HuBeRT [19], which is unsupervisedly trained with many hours of speech and fine-tuned to work in many different domain-specific tasks. After HuBeRT generates the embedding, a simple feedforward network with one hidden layer is applied to the data. The output layer uses a softmax activation function to produce a probability distribution over the emotion classes. Then, the model is trained using cross-entropy loss to minimize the divergence between predicted and ground-truth labels.

Table 1: Defined classes and brief descriptive textual examples.

Class	Class Description
0 - Slate	"Recording: NHA 3/18"
1 - Very excited	Soccer match: "Flamengo versus Botafogo. Who will become the leader?"
2 - Enthusiastic	Variety programs: "At dancing with the famous, a beauty showoff..."
3 - Happy	Film: "3 Times more adventure, Spider-Man 3"
4 - Narrative	Series: "The town hero cannot do anything to save her"
5 - Serious/Neutral	Telenovela: "Now, Sophie will have the help of her daughter..."
6 - Sad/Down	Film: "Sometimes the only thing to change her life is..."
7 - Signature	Production name: "Secret lies"
8 - Question	Questions: "Can we improve our mind through food?"
9 - Date/Signature	Production name + date/day of the week: "Next Monday, Sunday, tomorrow: the Martin"
10 - Discarded	Coughs, noisy recordings, etc.

To measure the performance of the result before extrapolating the actual labels, we split the labeled dataset into training, validation, and test subsets. We performed rounds of fine-tuning on the first two until satisfactory accuracy was achieved. After checking the adjusted model on the test subset, it was applied to the unlabeled data.

To illustrate that generated labels were satisfyingly accurate for our pipeline, Figure 1 shows the confusion matrix for the generated labels in our test subset. We can see a significant correlation between ground-truth labels and the predicted data.

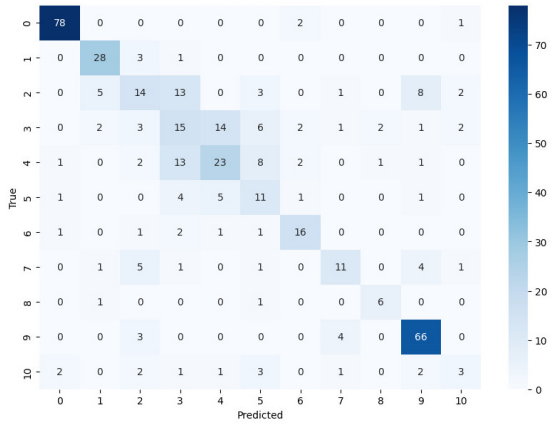


Fig. 1: Confusion matrix for the automatic class predictor against manually labeled data.

However, we can observe that the confusion matrix reveals a few situations in which emotions are not clearly differentiated (e.g. classes 3 and 4 in Figure 1); in fact, they reflect the very difficulty of subjectively classifying the nature of emotion in such cases, which will generally produce class overlap. Although this overlap may suggest a possible class reduction, misclassifications are not harmful in this case, as they can mean that intentions can be interchanged according to the context. This is another advantage of generating application-aware classes, as the overlapping classes may still fit the end-user intention.

Class 10 (which represents "bad quality" samples) is the only one with significant prediction spread, which should be an effect of this different class nature. This is another reason why we discarded this class afterwards.

This way, we could further enhance the synthesis quality by drastically increasing the dataset size without semantic loss, given that the predictor is trained to follow the human criteria to generate labels.

With this automatic class predictor, we completed the necessary data processing. The complete block diagram for all the data pre-processing pipelines is illustrated in Figure 2.

4 Model implementation and training

With the data formatted as a `.csv` file, we used the VITS2 [11] open-source model repository. VITS2 is an evolution of VITS [10], a well-established TTS model with official training implementation procedures out-of-the-box, which made it a community validated codebase. With VITS2, the embedding input architecture was enhanced to modify the phoneme features deeply,

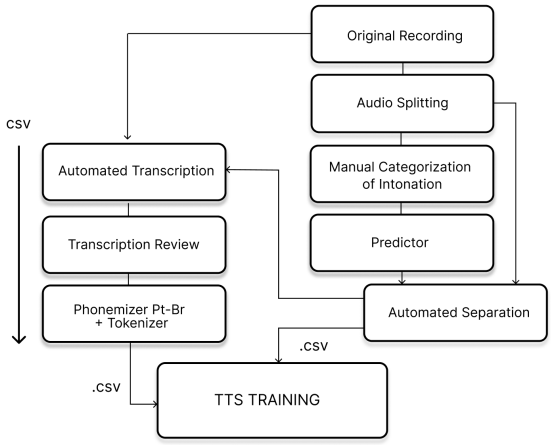


Fig. 2: Block diagram of data-processing pipelines.

and we take advantage of that to get increased emotion control precision and quality.

The VITS2 model was trained with the original hyperparameters (except for batch size) and mostly the original architecture available on the official repository[3], and the training took around 300 thousand steps to converge, which took a week using four NVIDIA RTX 8000 GPUs. The follow-up early stop procedure was to perceptually determine when prosody and quality stopped improving for some time.

With a free embedding network and labeled data, the optimization could find a latent space that encoded emotional features from data content. By following this strategy, we could successfully control intonations by replicating the data present for each label without losing audio quality, as can be verified in the examples available on our website[4]. There, we present samples of the exact text spoken with distinct and controlled emotions/prosody.

5 Fine-grained control

By our experience, even with intonation class control, the editorial team sometimes needs to fine-tune the prosody for a syllable or word. Because of this, we also designed a fine-grained prosody control where the user can choose the duration and pitch with phoneme or word granularity. With this, we can avoid the common text-changing iterations generally needed for high quality speech when the TTS system does not capture the intended prosody.

We enabled fine-grained control by dissecting the VITS2 model. Inside the text feature extraction modules, VITS2 predicts duration and pitch for each token (in our case, phoneme representation). We extracted these phoneme-level features by inferring just the early layers in the network for the given text, which gives us duration and pitch information for each of the corresponding IPA characters, producing two arrays of floats: one for character pitch, and one for character duration, with the length of the phrase in IPA characters plus spacing.

These phoneme-level pitch and duration arrays were then made available for user customization through a graphical interface, in two options: phoneme- or word-level controls. When using phoneme-level controls, the changes are applied directly to the corresponding phonemes, as shown in Figure 3. For word-level knobs, on the other hand, we have to first aggregate the features by taking the mean pitch and duration sum for each word, and then disaggregate them after customization by scaling the individual features using the same factor by which the word-level value was scaled in the interface.

Figure 4 shows an example of the word-level controls in the user interface. The prosody customization is done, in practice, as follows: the user may drag the boxes around a word to increase or decrease its duration and change the slider level inside the boxes to customize pitch for a specific word. The same controls could be more fine-grained, controlling phoneme per phoneme with the same interface.

After the necessary changes, the inference is resumed from the user-tweaked features, telling the models to adapt the speech for slower/faster pronunciation or higher/lower pitch. In order allow controlling the general tendency for a word or space between words, we also grouped same-word phonemes and spaces into chunks, so the user can change the tendency for each chunk as a whole without having to access each character, which is generally preferred for speed.

[3] https://github.com/daniilrobnikov/vits2

[4] https://gpa-smt-ufrj.github.io/aes-aimla2025

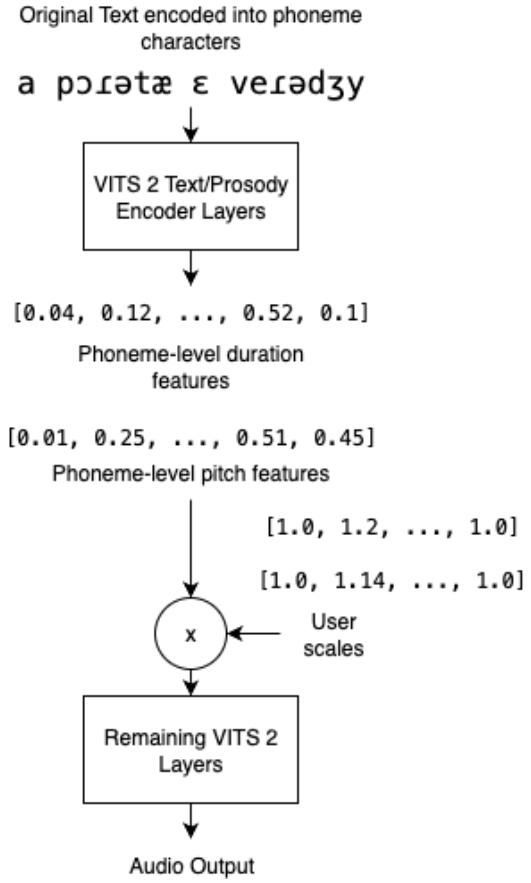


Fig. 3: Block diagram for the phoneme-level feature customization system. The VITS2 architecture is dissected to extract character prosodic features, which are scaled individually as desired by the user.

6 Subjective evaluation

A blind subjective test was conducted to validate the quality of the synthetic voice. In this test, three pairs of video clips were tested in a comparison setup, containing promos of a sports match, a drama series, and a documentary. The same video, music, and sound effects background was maintained for all comparison pairs, and only the narration was changed. The test samples were chosen to match the applications of the promo content, using real promos that aired recently before the time they were selected, and corresponding synthetic replications of them. To ensure consistency, they were also normalized to match the ITU 1770 loudness

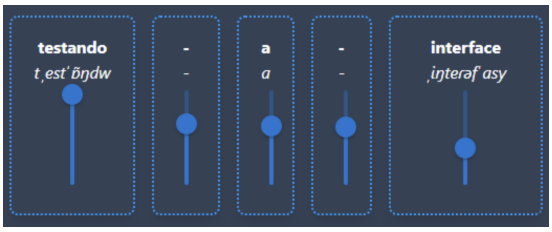


Fig. 4: Fine-grained word-level prosody controls. Each word has a box and a slider. Changing the size of the box by dragging it around customizes the duration of the word, while moving the slider changes the desired pitch intention for that word.

recommendation [20] at -23 LUFS using the *Waves WLM Plus Loudness Meter*[5] plugin for the *Avid Pro-Tools*[6] DAW, and were subject to the same equalization patterns.

For each pair, one clip featured the original recorded narration, while the other presented the synthetic voice generated using our proposed pipeline, with fine-grained adjustments applied to specific words or phonemes when the operator understood them as unnatural. The synthetic samples were also generated to mimic the emotions and intonations that are historically used in such narrations for each context, but without guidance from the real samples. We used a five-point discrete scale where users could evaluate the generated audio against original content, with the given options as in Table 2, where synthetic and original samples were randomly chosen as Sample A or Sample B for each content pair.

The idea was to assess whether or not there was a difference between synthetic-generated narrations and original recordings on the end-user-ready content, and, if there was such a difference, how significant would it be. To address that, we opted for a categorical answer list instead of a continuous/numerical scale, aiming to simplify the evaluation process and minimize potential misunderstandings regarding numerical ratings and their relationship to perceived audio quality.

The subject panel consisted of 29 specialists, resulting in 87 pair evaluations. All subjects were chosen for having notable previous audio and video post-production,

[5] https://www.waves.com/plugins/wlm-loudness-meter

[6] https://www.avid.com/pro-tools

Table 2: Possible options to be chosen by subjects in the audition tests, with assigned linear indexes.

Question Answer options
Sample A is MUCH BETTER than Sample B.
Sample A is SLIGHTLY BETTER than Sample B.
Both samples are EQUALLY ACCEPTABLE.
Sample B is SLIGHTLY BETTER than Sample A.
Sample B is MUCH BETTER than Sample A.

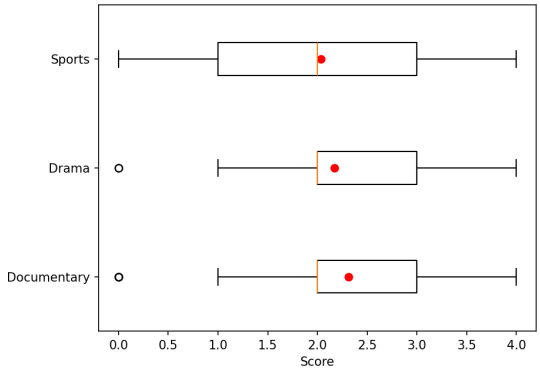


Fig. 5: Box plot for the audition tests. The whiskers show the response range without outliers (white circles), the box represents where 50% of the responses were, orange lines are the medians, and red dots represent the means.

editing, and/or mixing professional experience. To avoid biases, we did not give the subjects prior information about the nature of each piece of content. Also, there was no indication that artificial intelligence was used, and the subjects were asked to optionally leave their observations about the content using a textual field, right after the test.

The equipment used for the tests was a 42-inch TV and a Sony 7506 headphone with equal sound volume for every subject, played from a notebook computer. The original recordings in this test were not used for any data or training procedure.

6.1 Results

Assuming a linear preference scale for the discrete results where 0 represents the response "*Original content is MUCH BETTER than synthetic content*" and 4 represents "*Synthetic content is MUCH BETTER that original content*", we draw a box plot for each content as shown in Figure 5.

We can infer from the results that there is no significant difference between the synthetic and the original recordings, and we even observe a slight tendency towards synthetic preference. This tendency can be explained by the enhanced controllability given to the voice directors, who could adjust prosody according to the finalized content. In the human case, narrations are generally the first asset to be created, and cannot be easily customized afterwards due to logistical difficulties.

As for the optional textual feedback, none of the participants reported any degradation in the speech samples. In fact, some explicitly stated they couldn't detect any difference in quality between them. We then highlight that these comments should also be good indicators that the synthetic content is ready for production, as even attentive listeners were unable to perceive any notable differences in quality.

The pieces used in the subjective tests and other generated speech samples are available at our website[7]. We also make the individual datapoints from the test available, so it is possible to verify the detailed response distribution.

7 Conclusion and future works

Considering these results, we advanced this workflow to production, since we concluded that delivering the content to a broader audience without any significant perception of loss in quality was feasible.

The proposed method of producing synthetic speech has introduced a new paradigm for scalable and controllable speech production, which is validated by a fast-paced production environment in a relevant broadcast company. Training and pre-processing steps were also optimized and adequate for fast-paced broadcast environments, not to mention the increased controllability and robustness for generating speech content driven

[7] https://gpa-smt-ufrj.github.io/aes-aimla2025/samples

by the operator's intention, with selectable emotions and fine-grained prosody control.

Also, this tool allowed the scriptwriters to test their text choices before the recording stages started and eventually make early changes that could avoid retakes later in the pipeline.

The same methodology was also tested and applied in production for another broadcast environment: voice-overs for pre-recorded clips in a variety show, with new classes and another notable voice in Brazil. With the pipeline validated for promo content, this extension was straightforward and very similar (the only necessary adaptation being to change the data entries and the definition of classes), showing that the proposed workflow can be easily replicated as needed.

As can be seen, this field has vast potential for future research that directly impacts many audio production areas. In the future, we aim to have the same process in a broad-spectrum form, with multiple speakers, accents, and emotions. We hope this can be achieved by decoupling and isolating latent spaces through curated datasets and carefully designed embedding systems, which should employ unseen dimensions for different datasets. We hope this workflow documentation can further impact the industry and academy as a starting point for good practices and an example of a validated and scalable end-to-end synthetic speech system design.

Acknowledgment

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) Finance Code 001.

References

[1] Tang, H., Zhang, X., Cheng, N., Xiao, J., and Wang, J., "ED-TTS: Multi-Scale Emotion Modeling Using Cross-Domain Emotion Diarization for Emotional Speech Synthesis," in *2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 12146–12150, 2024.

[2] Jia, Y., Zhang, Y., Weiss, R. J., Wang, Q., Shen, J., Ren, F., Chen, Z., Nguyen, P., Pang, R., Moreno, I. L., and Wu, Y., "Transfer learning from speaker verification to multispeaker text-to-speech synthesis," in *32nd International Conference on Neural Information Processing Systems (NeurIPS 2018)*, pp. 4485–4495, 2018.

[3] Kwon, O., Jang, I., Ahn, C., and Kang, H.-G., "Emotional Speech Synthesis Based on Style Embedded Tacotron2 Framework," in *2019 International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC)*, pp. 1–4, 2019.

[4] Um, S., Oh, S., Byun, K., Jang, I., Ahn, C. H., and Kang, H.-G., "Emotional Speech Synthesis with Rich and Granularized Control," in *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7254–7258, 2020.

[5] Kharitonov, E., Vincent, D., Borsos, Z., Marinier, R., Girgin, S., Pietquin, O., Sharifi, M., Tagliasacchi, M., and Zeghidour, N., "Speak, read and prompt: High-fidelity text-to-speech with minimal supervision," *Transactions of the Association for Computational Linguistics*, 11, pp. 1703–1718, 2023.

[6] Wang, C., Chen, S., Wu, Y., Zhang, Z., Zhou, L., Liu, S., Chen, Z., Liu, Y., Wang, H., Li, J., He, L., Zhao, S., and Wei, F., "Neural Codec Language Models are Zero-Shot Text to Speech Synthesizers," `https://arxiv.org/abs/2301.02111`, 2023, online; accessed 1 Oct 2024.

[7] Zhao, W. and Yang, Z., "An Emotion Speech Synthesis Method Based on VITS," *Applied Sciences*, 13(4), 2023.

[8] Bengio, Y., Ducharme, R., and Vincent, P., "A neural probabilistic language model," *Journal of Machine Learning Research*, 3, pp. 932–938, 2000.

[9] Wang, Y., Stanton, D., Zhang, Y., Skerry-Ryan, R., Battenberg, E., Shor, J., Xiao, Y., Ren, F., Jia, Y., and Saurous, R. A., "Style tokens: Unsupervised style modeling, control and transfer in end-to-end speech synthesis," in *35th International Conference on Machine Learning (ICML)*, pp. 5180–5189, 2018.

[10] Kim, J., Kong, J., and Son, J., "VITS: Conditional variational autoencoder with adversarial Learning for end-to-end text-to-speech," in *2021 International Conference on Machine Learning (ICML)*, pp. 5530–5540, 2021.

[11] Kong, J., Park, J., Kim, B., Kim, J., Kong, D., and Kim, S., "VITS2: Improving quality and efficiency of single-stage text-to-speech with adversarial learning and architecture design," in *24th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, volume 24, pp. 4374–4378, 2023.

[12] Kim, J., Kim, S., Kong, J., and Yoon, S., "Glow-TTS: A generative flow for text-to-speech via monotonic alignment search," in *34th International Conference on Neural Information Processing Systems (NeurIPS)*, pp. 8067–8077, 2020.

[13] Doddipatla, R. S., Braunschweiler, N., and Maia, R., "Speaker adaptation in DNN-based speech synthesis using d-vectors," in *18th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 3404–3408, 2017.

[14] Raitio, T., Rasipuram, R., and Castellani, D., "Controllable Neural Text-to-Speech Synthesis Using Intuitive Prosodic Features," in *21st Annual Conference of the International Speech Communication Association, Interspeech 2020, Virtual Event, Shanghai, China, October 25-29, 2020*, volume 21, pp. 4432–4436, 2020.

[15] Leite, P. H. L., Hoyle, E., Álvaro Antelo, Kruszielski, L. F., and Biscainho, L. W. P., "Neutral TTS female voice corpus in Brazilian Portuguese," in *41st Brazilian Symposium on Telecommunications and Signal Processing (SBrT)*, pp. 1–5, 2023.

[16] Zhou, K., Sisman, B., Liu, R., and Li, H., "Emotional voice conversion: Theory, databases and ESD," *Speech Communication*, 137(1), pp. 1–18, 2022.

[17] Radford, A., Kim, J. W., Xu, T., Brockman, G., McLeavey, C., and Sutskever, I., "Robust speech recognition via large-scale weak supervision," `https://arxiv.org/abs/2212.04356`, 2022, online, accessed 15 May 2023.

[18] "International Phonetic Alphabet," `https://internationalphoneticassociation.org/content/ipa-chart`, 2023, online; accessed 06 Mar 2023.

[19] Hsu, W.-N., Bolte, B., Tsai, Y.-H. H., Lakhotia, K., Salakhutdinov, R., and Mohamed, A., "HuBERT: Self-Supervised Speech Representation Learning by Masked Prediction of Hidden Units," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29, pp. 3451–3460, 2021.

[20] International Telecommunication Union, Radiocommunication Sector (ITU-R), "ITU-R BS.1770-4: Algorithms to measure audio programme loudness and true-peak audio level," Technical Report BS.1770-4, International Telecommunication Union, Geneva, Switzerland, 2015, accessed: 2025-07-22.

Audio Engineering Society

Conference Paper 32

Presented at the AES International Conference on Machine Learning
and Artificial Intelligence for Audio
2025 September 8–10, London, UK

This paper was peer-reviewed as a complete manuscript for presentation at this conference. This paper is available in the AES E-Library (http://www.aes.org/e-lib), all rights reserved. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Journal of the Audio Engineering Society.

Neutone SDK: An Open Source Framework for Neural Audio Processing

Christopher Mitcheltree[1], Bogdan Teleaga[1], Andrew Fyfe[1], Naotake Masuda[1], Matthias Schäfer[1], Alfie Bradic[1], and Nao Tokui[1]

[1]*Neutone Inc.*

Correspondence should be addressed to Christopher Mitcheltree (christopher@neutone.ai)

ABSTRACT

Neural audio processing has unlocked novel methods of sound transformation and synthesis, yet integrating deep learning models into digital audio workstations (DAWs) remains challenging due to real-time / neural network inference constraints and the complexities of plugin development. In this paper, we introduce the Neutone SDK: an open source framework that streamlines the deployment of PyTorch-based neural audio models for both real-time and offline applications. By encapsulating common challenges such as variable buffer sizes, sample rate conversion, delay compensation, and control parameter handling within a unified, model-agnostic interface, our framework enables seamless interoperability between neural models and host plugins while allowing users to work entirely in Python. We provide a technical overview of the interfaces needed to accomplish this, as well as the corresponding SDK implementations. We also demonstrate the SDK's versatility across applications such as audio effect emulation, timbre transfer, and sample generation, as well as its adoption by researchers, educators, companies, and artists alike. The Neutone SDK is available at github.com/Neutone/neutone_sdk.

1 Introduction

Neural audio processing, a research field where machine learning and neural networks are used to analyze, transform, and generate sound, has seen rapid developments in the last decade alongside the increasing popularity of deep learning and the computational efficiency improvements of computers. Bridging the gap between researchers and practitioners is especially challenging in this field because neural audio processing brings its own set of problems, such as fixed buffer size and sample rate requirements as well as unique streaming and caching properties of neural architectures (see Section 3.1 for more), which are time-consuming to address and require expert domain knowledge to solve effectively. These problems are in addition to the usual real-time safety and C++ plugin development required for real-time audio processing. Therefore, when researchers publish new neural audio processing systems, they are often controlled asynchronously through a command line interface, which is in stark contrast to the feedback-driven workflow introduced by audio plugins and the Virtual Studio Technology (VST) standard.

The Neutone SDK is an open source framework for running real-time and non-realtime audio processing neural networks (which we refer to as models) in external audio software applications such as Digital Audio

Workstations (DAWs). The models exported using the SDK are wrapped in a consolidated interface layer that allows for agnostic interoperability across model deployment hosts. The Neutone SDK, as well as its counterpart free VST/AU host plugins Neutone FX and Neutone Gen, aim to bridge the gap between neural audio research and the music creator community. Although the focus of this paper is on the Neutone SDK, it is important to acknowledge that many of the design decisions of the SDK complement the design decisions made for the two host plugin inference engines that run the models exported using the SDK.

The Neutone project was conceived in early 2022 by Qosmo[1], a Tokyo-based AI creativity and music lab, and is now owned and maintained by Neutone[2], a company focused on building next-generation AI tools for artists and musicians. The open source Neutone SDK and accompanying Neutone FX plugin followed on from the RAVE Audition VST project in 2021 (which was limited to deploying a single specific model architecture) to provide a model-agnostic solution for the deployment and distribution of neural audio technologies in the DAW.

Deploying arbitrary audio processing neural networks in a host-agnostic manner requires various interfaces for control parameters, audio processing, DAW compatibility, and real-time / non-realtime inference. In this paper, we provide an overview of each of these conceptual requirements and how they are implemented in the Neutone SDK with an emphasis on Python-only development and ease of use.

2 Background

Some notable projects that exist to help bring neural audio research into the DAW include RT-Neural [1], a lightweight solution with a C++ inference library for hard real-time applications. RT-Neural's real-time safety constraints make it highly desirable in many audio processing scenarios, however it is limited to handling a subset of operators and model layer types. Machine learning frameworks like PyTorch, JAX, and Tensorflow, are at the cutting edge of deep learning research and provide the functionality required to build modern and complex neural networks. RT-Neural and similar libraries are not compatible with many model architectures, and a solution built around the external runtimes of these frameworks is often required instead.

Prior to the introduction of the Neutone SDK, researchers interested in productionizing their real-time neural audio models would need to develop a lot of additional infrastructure on top of the model architecture itself to meet the requirements for real-time audio streaming environments. To the best of the authors' knowledge, there were no publicly available solutions that streamlined the process of deploying real-time neural audio models to the DAW before the release of the Neutone SDK. However, the Neutone SDK drew inspiration from the earlier project Audacitorch that provides a mechanism for deploying non-realtime machine learning models in the Audacity DAW [2]. Audacitorch implements a wrapper design that its model inference counterpart can then interpret within Audacity as part of an integration. The Audacity integration allows models to be selected and inferred from within the DAW software where the model outputs are also rendered, thus mitigating the need to leave the DAW environment. The Neutone SDK and its host plugins took inspiration from this improved workflow for DAW users, as working within the DAW is familiar and efficient for many and reduces the need to context-switch between software environments when trying to stay focused on the creative process.

Both Neutone SDK and Audacitorch build on top of PyTorch and its model serialization tool TorchScript. Serializing a model using TorchScript renders it to a format that does not rely on Python dependencies and can be used in C++ external software applications. The ease of use, object-orientated design, and utilities of PyTorch, in addition to its popularity among the neural audio research community, make PyTorch and TorchScript the machine learning framework of choice for the Neutone SDK.

Since the release of the Neutone SDK there have been various other tools to improve the state of running neural networks in real-time audio environments like the DAW. Similarly to Neutone FX, Anira [3] is a C++ inference engine catered for the deployment of real-time models built for major framework runtimes like LibTorch, TensorFlow Lite, and ONNX, and can be embedded inside an application without having to build custom infrastructure to support these inference capabilities. 'nn_tilde' [3] by IRCAM is another project that

[1] qosmo.jp/en
[2] neutone.ai
[3] github.com/acids-ircam/nn_tilde

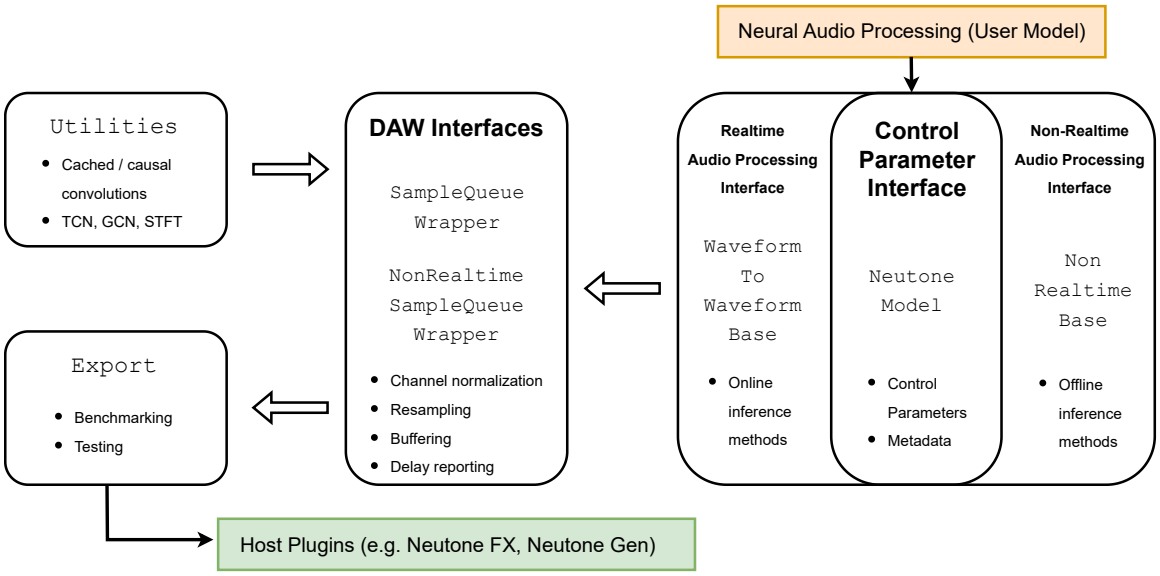


Fig. 1: Overview of the Neutone SDK. Components in `monospace` font are corresponding SDK implementations.

allows the deployment of serialized TorchScript models as a MAX/MSP external. Historically, nn_tilde deployed only RAVE models [4], but has since demonstrated compatibility with other real-time model architectures. Additionally, HARP [5] addresses the problem using a different approach by sending and receiving model inputs and outputs via a server to a Python environment and a Gradio interface. This is ideal for large models that must be hosted remotely on a GPU server and scenarios where there is a stable network connection. The machine learning tooling of HARP is in Python which is much more familiar to many neural audio processing researchers and means that models can be run from Python directly. This greatly reduces the complexity for researchers to share their work with the artist community and is why the Neutone SDK is also written in Python, but with an emphasis on real-time and local models running on CPU.

3 Implementation

3.1 Overview

The Neutone SDK[4] consists of several interfaces and utilities that enable users to wrap, test, benchmark, and export neural audio processing models written in PyTorch so that they can be loaded by a plugin and run in the DAW. Running neural networks in the DAW can be broken down into several key tasks: handling control parameters, audio processing, dynamic sample rate / channel / buffer size management, and real-time / non-real-time inference. Each of these tasks is discussed in more detail in the following subsections, and has a corresponding interface in the SDK. Figure 1 provides an overview of the design of the SDK and its components.

All of the classes in the SDK are compatible with TorchScript, meaning they can be exported to C++ automatically, thus enabling the user to work entirely in Python without needing to learn specialized audio plugin development skills. This also makes it very flexible since users can overwrite and extend any method or class if they have custom requirements for their specific applications. In addition to this, the SDK abstracts away many technical challenges that arise when working with real-time audio processing such as:

- Variable buffer sizes / optimal buffer size selection
- Variable sample rates, optimal sample rate selection, and resampling
- Channel normalization
- Delay compensation and reporting
- Streaming and caching support for real-time audio buffer processing
- Control parameter handling and aggregation
- Minimizing dynamic memory allocations (since they are not real-time safe)
- Lookbehind buffer management
- Spectral processing and freq. domain conversion

[4] `github.com/Neutone/neutone_sdk`

This abstraction is especially important for neural audio processing because many of the aforementioned technical challenges become exacerbated by neural networks, since they typically have fixed sample rate, buffer size, and channel requirements.

3.2 Neural Audio Processing

Before taking advantage of the Neutone SDK, a user must start with a neural audio processing system that takes audio, control parameters, and/or text as inputs and outputs one or more tracks of audio. Sections 3.8.4 and 4 provide an overview of representative use cases and practical implementations of such systems.

3.3 Control Parameter Interface

The first interface needed to deploy a neural audio processing system in the DAW is one that defines its control parameters. This interface ensures that users can influence model behavior in real time or offline and can be manipulated via external input, such as knobs, sliders, or text fields in a plugin UI.

The Neutone SDK implements this as an abstract base class called `NeutoneModel` which inherits from `torch.nn.Module`. This base class is inherited by `WaveformToWaveformBase` (Section 3.4) and `NonRealtimeBase` (Section 3.6) which are then used to wrap arbitrary `torch.nn.Module` audio processing neural networks (which we refer to as models). This is where the names and default values of the control parameters of the model are defined. The Neutone SDK currently supports three types of control parameters:

1. `ContinuousNeutoneParameter`: a continuous value between 0 and 1 that can be further post-processed by the user and provides control values at a sample or buffer level of granularity.
2. `CategoricalNeutoneParameter`: a categorical value with discrete positions of 2 to N and descriptive labels.
3. `TextNeutoneParameter`: a text input with a maximum number of permitted characters (only available for non-realtime models).

The base class also defines metadata about the model being wrapped that the user must provide, such as its name, authors, tags, description, citation, and technical links which can then be made available to a host plugin.

3.4 Real-time Audio Processing Interface

Next, an interface must be defined for handling input and output audio buffers. Real-time models generally have a one-to-one mapping between input and output tracks compared to offline models, which is why the Neutone SDK splits these interfaces into corresponding real-time (this section) and non-realtime (Section 3.6) implementations. The `WaveformToWaveformBase` class (W2W) serves as the real-time audio processing interface in the SDK and inherits from `NeutoneModel`. It wraps an arbitrary `torch.nn.Module` model that processes sequential audio buffers which means it takes as input an audio tensor of shape (C_{in}, N_{model}) and outputs an audio tensor of shape (C_{out}, N_{model}) where C is the number of channels and N_{model} is the number of samples. Currently, W2W supports values of C_{in} and C_{out} of 1 (mono) and 2 (stereo) as well as any buffer length (i.e., $N_{model} \in Z^+$).

The `WaveformToWaveformBase` class contains all audio-specific metadata. It includes methods for defining C_{in} and C_{out} of the model being wrapped, all supported N_{model} values, what sample rates the model is compatible with (f_{model}), and whether / how control parameter values are aggregated from a sample to buffer level of granularity. If the user's model introduces delay to the input signal, this can be specified and is then made available for the host plugin to query and communicate to the DAW which then compensates the delay to keep all tracks in sync. Finally, the user can also specify N_{LB} samples of lookbehind buffer which changes the shape of the input audio tensor from (C_{in}, N_{model}) to (C_{in}, N_{LB} + N_{model}). This is useful for neural networks with a large temporal receptive field that could benefit from looking at previously processed audio samples, but where introducing additional delay to the system via a lookahead buffer is undesirable.

3.5 Real-time DAW Interface

Once a neural model has been equipped with a control parameter interface and an audio processing interface, it must still be adapted to operate within the constraints of a DAW. This final layer of abstraction is handled by the real-time DAW interface, which bridges the gap between the wrapped model and the host environment. Unlike the previous two interfaces that focus on model behavior and input / output audio, this interface is responsible for managing runtime constraints imposed by the DAW, such as varying

buffer sizes, sample rates, and channel configurations. The Neutone SDK encapsulates this functionality in the `SampleQueueWrapper` class which addresses many of the technical challenges listed in Section 3.1. It silently handles any mismatches between the DAW and model buffer sizes (N_{DAW} and N_{model}), channel dimensions (C_{DAW}, C_{in}, and C_{out}), and sample rates (f_{DAW} and f_{model}). Normal use of the SDK does not involve interacting with this class, but the SQW is automatically wrapped around W2W classes during the export process (see Section 3.8.3).

When loading an exported Neutone model, the SQW first determines the optimal buffer size N_{model} and sample rate f_{model} for the wrapped model based on the metadata provided by the user in the audio processing interface and the output audio requirements N_{DAW} and f_{DAW} specified by the DAW and communicated to the SQW by the host plugin. It also calculates any additional buffering delay caused by mismatched DAW and model buffer sizes. These calculations are non-trivial and often non-intuitive because they depend on the cyclical properties of the modular arithmetic between the two different buffer sizes. Resampling between commonly used sample rates such as 48 kHz and 44.1 kHz can also result in coprime resampled input and model buffer sizes which means it is critical to compute the buffering delay correctly to avoid dropouts and stuttering during playback. We implement the algorithm described in [6] which guarantees the lowest possible amount of additional delay from buffer size adaptation (which we refer to as buffering delay). Once all buffer sizes and sample rates have been calculated and selected, the SQW pre-allocates input, output, control parameter, and lookbehind circular queues as required and reports the total resampling, buffering, and model delay to the host plugin.

During inference, the SQW makes use of the channel normalization and resampling sandwiches (see Section 3.8.1) to convert incoming and outgoing audio buffers to the specified model input / output channel sizes ($C_{DAW} \rightarrow C_{in}$ and $C_{out} \rightarrow C_{DAW}$) and selected optimal model sample rate ($f_{DAW} \rightarrow f_{model} \rightarrow f_{DAW}$). It also provides methods for the host plugin to perform audio thread, background thread, and offline inference, each of which require different buffering implementations and overall make the Neutone SDK compatible with most host plugin architectures and audio processing use cases. Finally, the SQW can adapt the wrapped model to dynamically changing f_{DAW} and N_{DAW} through its `set_daw sample_rate_and_buffer_size()` method and it also includes a `reset()` method for clearing all model, audio processing interface, and real-time DAW interface state. Figure 2 visualizes the flow of an audio buffer through the SDK, as well as showing an example initialization sequence that occurs when f_{DAW} and N_{DAW} change.

3.5.1 Neutone FX

Neutone FX[5], shown in Figure 3, is a free VST/AU host plugin for MacOS and Windows that was developed alongside the Neutone SDK and provides an interface for artists, creators, and researchers to access real-time neural audio processing models. It includes a model browser that allows one to filter, search through, and download user models that have been shared and uploaded to the Neutone servers as well as providing the option to load your own locally wrapped and exported model via the SDK (see Section 3.8.3 and Figure 5).

Neutone FX is written in C++ using the JUCE framework, and is built as an audio effect, meaning it takes individual audio buffers as input, and processes / outputs them sequentially in a streaming fashion. Its UI supports four custom control parameter knobs (`ContinuousNeutoneParameter` and `CategoricalNeutoneParameter`), as well as wet, dry, input gain, and output gain controls. The plugin interfaces with the SQW and handles moving audio buffers between the audio (foreground) thread and a background thread, which is required for running neural audio processing models with fixed buffer sizes. This thread management challenge cannot be solved by the Neutone SDK alone, as the SDK is written in TorchScript compatible PyTorch, limiting direct control over multi-threading. This constraint is an intentional design choice to ensure the SDK remains entirely in Python. Fortunately, open-source frameworks like Anira [3] provide solutions for efficient real-time processing at the C++ and audio plugin level, making them a natural complement to the Neutone SDK.

3.6 Non-realtime Audio Processing Interface

Offline audio processing models generally have fewer constraints compared to real-time models, but are consequently more versatile and have a wider variety of inputs and outputs. The `NonRealtimeBase`

[5] `neutone.ai/fx`

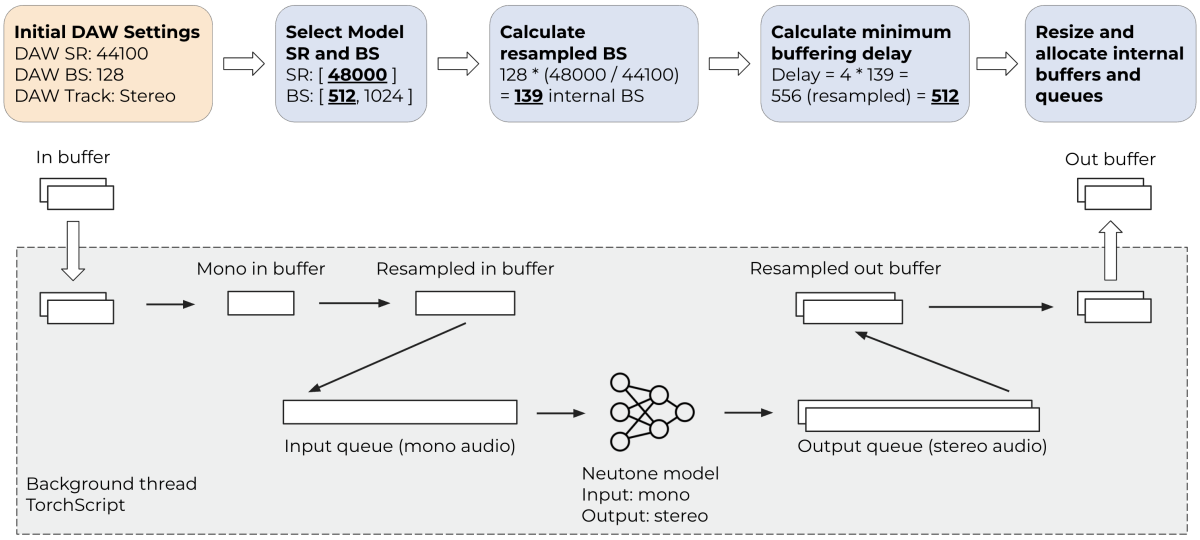


Fig. 2: Flow of an audio buffer through the Neutone SDK. The sequence at the top visualizes the steps that occur when an exported Neutone model is loaded by a host plugin such as Neutone FX. Everything inside the gray box has been compiled to C++ automatically via TorchScript and is handled by the `SampleQueueWrapper`, `WaveformToWaveformBase`, and `NeutoneModel` classes in the SDK.

class (NRB) in the SDK provides the interface for non-realtime audio processing models. It inherits from `NeutoneModel` and wraps an arbitrary `torch.nn.Module` audio processing model for offline inference.

The NRB is very similar to the W2W class (Section 3.4), except that it has been generalized to support multiple input and output audio tracks as well as text control parameters. This improves the compatibility of the SDK with the comparatively wider range of non-realtime model applications. A common use case that requires multi-track functionality is stem separation, where a model takes a single input audio track and extracts the individual instrument tracks from it. The NRB also includes methods for early termination of model inference calls and querying inference progress, which enables updating host plugin UIs accordingly.

3.7 Non-realtime DAW Interface

Just like real-time models, offline models also require an interface to seamlessly handle dynamic host conditions. The `NonRealtimeSampleQueueWrapper` (NRSQW) provides this interface for models wrapped with `NonRealtimeBase`, offering functionality analogous to the real-time `SampleQueueWrapper`, but optimized for offline processing. The NRSQW provides the same buffer size, sample rate, and channel normalization as outlined in Section 3.5, generalized to multi-track audio inputs and outputs. During inference, inputs are padded, block-segmented, and processed sequentially, with delay compensation, progress tracking, and cancellation support for responsive host plugin UI updates. Outputs are then concatenated and de-padded to preserve alignment with the original input. The NRSQW supports both one-shot inference as well as block-wise streaming for large inputs, making it well-suited for complex models that exceed real-time constraints.

3.7.1 Neutone Gen

As a natural complement to Neutone FX, Neutone Gen[6] (shown in Figure 4) is a free VST/AU host plugin for neural audio models that do not run in real time. The plugin loosely emulates the design of traditional audio sampler software, whereby users can populate a custom library of audio samples output by the model being hosted. These audio samples can be generated, auditioned, and trimmed within the plugin before being dragged into the DAW's timeline for further processing.

Neutone Gen's workflow enables a wider variety of models to be wrapped with the SDK, including those with text prompt input configurations. Heavier, high-quality variants of models used in Neutone FX can also be deployed, such as intensive stem separation

[6] `neutone.ai/gen`

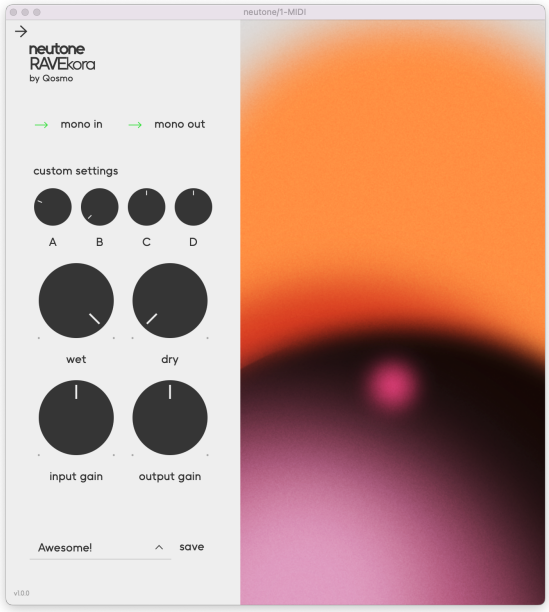


Fig. 3: User interface of the Neutone FX host plugin.

and timbre transfer algorithms. The plugin can interface with both the `SampleQueueWrapper` and the `NonRealtimeSampleQueueWrapper` interfaces, thus making all real-time models for Neutone FX compatible with Neutone Gen. Audio playback can also be synchronized to the DAW via automatic tempo adjustment and looping controls which facilitates the use of offline models in both music production and live performance scenarios.

3.8 Additional SDK Functionality

Besides the general interfaces discussed in the previous sections, the SDK also comes with several utility classes and methods to help with building and exporting neural audio processing systems in PyTorch for deployment in the DAW.

3.8.1 Sandwiches

The sandwich classes play an integral role in the DAW interface classes and provide implementations of input and output audio channel normalization and resampling algorithms, thus "sandwiching" a W2W or NRB wrapped Neutone model in the middle. Like most components in the Neutone SDK, users can inherit from the existing sandwich classes to implement their own custom algorithms.

Channel Normalization —
the `ChannelNormalizerSandwich` class is very

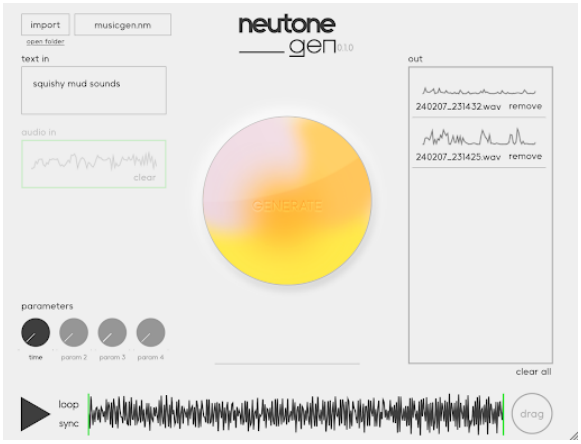


Fig. 4: User interface of the Neutone Gen host plugin.

simple and performs $C_{DAW} \to C_{in}$ and $C_{out} \to C_{DAW}$ mappings by taking the mean of multiple channels when $C_{DAW} > C_{in}$ and $C_{out} > C_{DAW}$, and duplicating channels when $C_{DAW} < C_{in}$ and $C_{out} < C_{DAW}$.

Resampling — The `ResampleSandwich` is an abstract base class defining a bidirectional resampling interface ($f_{DAW} \to f_{model}$ and $f_{model} \to f_{DAW}$). High-quality, band-limited resampling is computationally expensive, which presents a challenge for neural audio processing: any cycles spent on filtering and interpolation cannot be spent on expensive neural network inference. To address this, the Neutone SDK provides two zero-allocation, in-place resamplers:

1. `InplaceLinearResampler`:
   a simple linear interpolator (equivalent to convolving with a triangular kernel) that does not apply an anti-alias filter when downsampling and sacrifices quality for speed and simplicity. This implementation is ∼40% faster than `torch.nn.functional.interpolate` for common sample-rate ratios, with zero dynamic memory allocations.
2. `Inplace4pHermiteResampler`:
   A 4-point cubic Hermite spline interpolator (informed by [7]), also without dynamic memory allocations. The Hermite kernel yields a slightly steeper frequency roll-off than the triangular kernel, providing mild low-pass smoothing that reduces (but does not entirely eliminate) aliasing, yet remains only ∼1.8× slower than the linear resampler. This offers the best quality vs. complexity trade-off among polynomial interpolators on un-oversampled audio (see [7] for more details).

3.8.2 Utilities

While many neural audio processing researchers are familiar with PyTorch or similar deep learning frameworks, writing TorchScript compatible PyTorch code for real-time inference can be quite challenging, which raises the barrier to entry for using the Neutone SDK. As a result, the SDK includes several modular, commonly used components for neural audio processing that are optimized, avoid dynamic memory allocations, and are TorchScript-able.

Circular Queue — an optimized ring buffer for storing audio or parameter control values and is used extensively by the SQW.

Cached and Causal Convolution — cached (streaming) and causal 1D convolution implementations with receptive field reporting for time domain neural audio processing. Interested readers may refer to [8] and [9] for a detailed overview of this topic.

TCN — temporal convolutional neural network (TCN) implementation that supports conditional feature-wise linear modulation (FiLM) [10], streaming, and causal convolutions. TCNs are a popular architecture for neural audio processing due to their efficiency and large temporal receptive field, making them suitable for real-time applications. Interested readers may refer to [9] for an in-depth explanation.

GCN — gated convolutional neural network (GCN) [11] implementation: a special case of a TCN for modeling behavior over longer time scales using time-varying FiLM conditioning.

Spectral Processing — a streaming implementation (`RealtimeSTFT` class) of the short-time Fourier transform (STFT) and its inverse for building spectral processing algorithms. It includes helper methods for handling spectrogram normalization, crossfading, phase information, and delay reporting. The SDK also includes a `CachedMelSpec` class which enables streaming computation of Mel spectrogram features.

Filters — highpass, lowpass, bandpass, and bandstop streaming finite impulse response (FIR) and infinite impulse response (IIR) filter implementations as well as a state variable filter (SVF) implementation.

3.8.3 Export

Once a user's model has been wrapped using the audio processing interfaces, models are exported by the SDK as a TorchScript file. As mentioned in Section 3.5, models are automatically wrapped with their corresponding DAW interface (SQW or NRSQW) when exported. To make it easier for users of the SDK to load models in a host plugin and contribute models upstream, all relevant metadata and input/output example audio samples are bundled inside the TorchScript model file. The metadata bundled within the exported model file consists of both functional metadata which is used by the host plugin to perform inference correctly and cosmetic metadata which is used to display information about the model like control knob descriptions and default values in the host plugin UI. A schema is used for the metadata to provide consistency for downstream applications.

During export, an additional round of optimization and inference preparation is applied to the model and all Neutone SDK layers before scripting them together. Once the model is exported to disk, it can be locally loaded into a host plugin like Neutone FX or Gen and tested by the developer in the context of a DAW. Since the SDK is open source, models can also be loaded and run by third-party host plugins. The exported file can be shared with anyone, but users can also submit it on GitHub[7] to be added to the model browser of Neutone FX, making it available to all users of the plugin around the world. Models publicized in this way must comply with Neutone's ethics, training data, and copyright policy[8]. The entire model development, local testing, export, and publication process is summarized in Figure 5.

Benchmarking — Two benchmarking utilities are provided within the SDK. The speed benchmark can be used to calculate the real-time factor (RTF) of the model on a given machine and verify that the model is fast enough to be executed in real time. The latency benchmark can be used to calculate the number of delay samples that will be reported to the DAW for a given model. Both benchmarks can be run for different combinations of sample rates and buffer sizes and their results are communicated to the user during export. A helper method for profiling models is also available and can be used to debug CPU and RAM usage and detect dynamic memory allocations.

[7] `github.com/Neutone/neutone_sdk/issues`
[8] `neutone.ai/blog/neutone-on-ai-and-copyright`

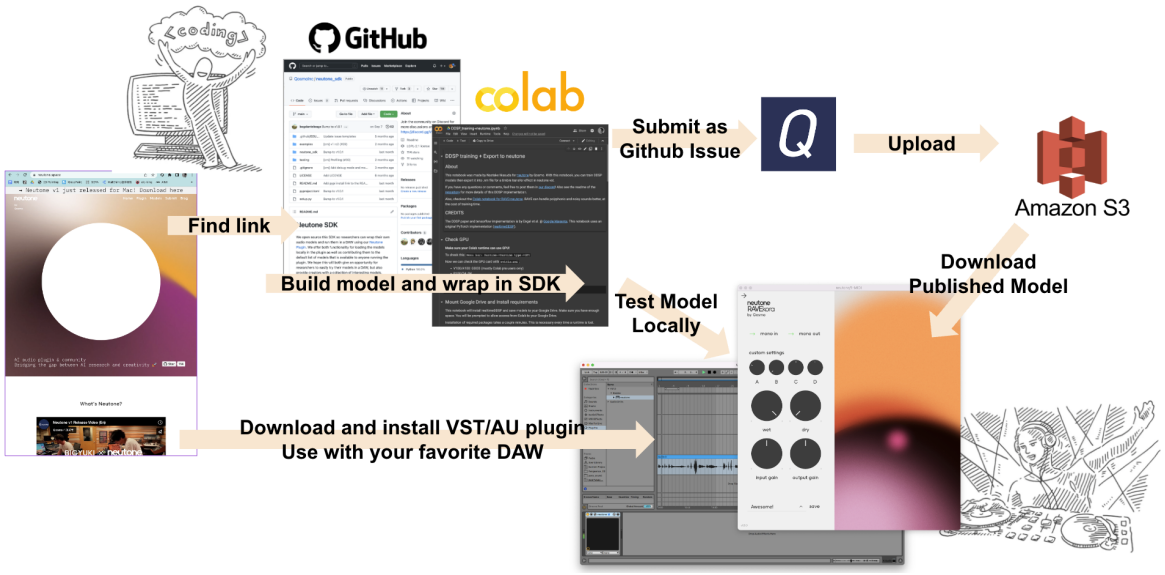


Fig. 5: Workflow for developing, testing, and publishing a neural audio processing model using the Neutone SDK.

3.8.4 Examples

The Neutone SDK comes with several simple examples and Colab notebooks that can serve as templates for users to get started with the SDK as quickly as possible. While the Neutone SDK enables building neural audio processing models without writing any C++ code, the Colab notebook examples take this a step further and allow users to train and export models without writing any code at all.

4 Applications

Since the release of the Neutone SDK in early 2022, there has been steady adoption from the community, with the Neutone Discord server[9] containing more than 2000 members as of writing. Users range from independent artists that aim to train models on their own data, to researchers using the SDK for expanding the reach of their published papers, to professors and teachers using it in their classes for educational purposes, to companies using it for rapid internal prototyping before investing in custom C++ plugin development. Since any audio-to-audio model can be wrapped with the SDK, the possibilities are endless. Some common model types are: audio effect emulation / inversion / removal, timbre and tone transfer, stem separation, controllable sample generation, and spectral processing.

Papers that use the Neutone SDK to showcase and make their models accessible in the DAW include: a neural fuzz effect [11], a DDSP-based vocoder [12], causal modeling of time-varying phaser, flanger, and chorus audio effects [13], a neural spring reverb [14], a time-varying differentiable filter evaluated on the Roland TB-303 Bass Line synthesizer [15], and a DDSP wavetable synthesizer for modulation discovery [16]. Several papers have also been made accessible in the DAW using the Neutone SDK after publication such as MelGAN [17], DDSP [18], RAVE [4], randomized overdrive neural networks [19], steerable discovery of neural audio effects [20], MusicGen [21], NoiseBandNet [22], and more. One benefit of publicizing academic research models through the Neutone SDK is that others can determine the limits of the system, even if they are not explored in the original academic work. In addition to this, allowing users to provide their own inputs reduces the cherry-picking bias of pre-selected audio samples.

Despite all the academic and industry neural audio processing research that leverages the Neutone SDK, the majority of users of the SDK are artists who are incorporating the Neutone host plugins in their creative process and performances.[10] At a time when many AI companies are directly competing with artists and training on their work without permission, the Neutone SDK democratizes this nascent technology and serves as a foundation for AI-powered tools that enhance, rather than replace, human creativity and artistic expression.

[9] discord.com/invite/r6WwYCvJTS

[10] neutone.ai/artists

5 Conclusion

In this paper, we have presented the interfaces required for real-time and offline neural audio processing in the DAW, a technical overview of the Neutone SDK, and discussed its applications and users. Development of the SDK is ongoing and we hope to explore the following improvements and directions in the future:

- New and updated examples and Colab notebooks
- Switching from TorchScript to ONNX to improve efficiency and cross-framework compatibility
- Implementing more utility components and layers as the neural audio processing field progresses
- Supporting symbolic models and releasing the Neutone MIDI SDK prototype

Readers interested in contributing are encouraged to open a pull request or submit an issue through the Neutone SDK GitHub repository.

Acknowledgments

We would like to thank Robin Jungers and Akira Shibata for their significant contributions to Neutone during its formation, as well as Nic Becker and Julian Lenz for their work on the Neutone MIDI SDK prototype.

References

[1] Chowdhury, J., "RTNeural: Fast Neural Inferencing for Real-Time Systems," *arXiv preprint arXiv:2106.03037*, 2021.

[2] Garcia, H. F., Aguilar, A., Manilow, E., Vedenko, D., and Pardo, B., "Deep Learning Tools for Audacity: Helping Researchers Expand the Artist's Toolkit," in *NeurIPS Workshop on Machine Learning for Creativity and Design*, 2021.

[3] Ackva, V. and Schulz, F., "ANIRA: An Architecture for Neural Network Inference in Real-Time Audio Applications," in *IEEE Int. Symp. on the Internet of Sounds (IS2)*, pp. 1–10, 2024.

[4] Caillon, A. and Esling, P., "RAVE: A Variational Autoencoder for Fast and High-quality Neural Audio Synthesis," *arXiv preprint arXiv:2111.05011*, 2021.

[5] Benetatos, C., Cwitkowitz, F., Pruyne, N., Garcia, H. F., O'Reilly, P., Duan, Z., and Pardo, B., "HARP 2.0: Expanding Hosted, Asynchronous, Remote Processing for Deep Learning in the DAW," in *International Society for Music Information Retrieval (ISMIR): Late Breaking Demo*, 2024.

[6] Letz, S., "Callback Adaptation Techniques," Technical report, GRAME, 2001, available at: `hal.science/hal-02158912`.

[7] Niemitalo, O., "Polynomial Interpolators for High-Quality Resampling of Oversampled Audio," 2001, available at: `yehar.com/blog/wp-content/uploads/2009/08/deip.pdf`.

[8] Caillon, A. and Esling, P., "Streamable Neural Audio Synthesis With Non-Causal Convolutions," in *Int. Conf. on Digital Audio Effects (DAFx)*, 2022.

[9] Steinmetz, C. J. and Reiss, J. D., "Efficient Neural Networks for Real-time Modeling of Analog Dynamic Range Compression," in *AES Convention 152*, Audio Engineering Society, 2022.

[10] Perez, E., Strub, F., de Vries, H., Dumoulin, V., and Courville, A., "FiLM: Visual Reasoning with a General Conditioning Layer," in *AAAI*, AAAI Press, 2018, ISBN 978-1-57735-800-8.

[11] Comunità, M., Steinmetz, C. J., Phan, H., and Reiss, J. D., "Modelling Black-Box Audio Effects with Time-Varying Feature Modulation," in *Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5, 2023.

[12] Südholt, D. and Erkut, C., "Vocal Timbre Effects with Differentiable Digital Signal Processing," in *Int. Conf. on Digital Audio Effects (DAFx)*, 2023.

[13] Mitcheltree, C., Steinmetz, C. J., Comunità, M., and Reiss, J. D., "Modulation Extraction for LFO-driven Audio Effects," in *Int. Conf. on Digital Audio Effects (DAFx)*, 2023.

[14] Papaleo, F., Lizarraga-Seijas, X., and Font, F., "Evaluating Neural Networks Architectures for Spring Reverb Modelling," in *Int. Conf. on Digital Audio Effects (DAFx)*, 2024.

[15] Yu, C.-Y., Mitcheltree, C., Carson, A., Bilbao, S., Reiss, J. D., and Fazekas, G., "Differentiable All-pole Filters for Time-varying Audio Systems," in *Int. Conf. on Digital Audio Effects (DAFx)*, 2024.

[16] Mitcheltree, C., Tan, H. H., and Reiss, J. D., "Modulation Discovery with Differentiable Digital Signal Processing," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2025.

[17] Kumar, K., Kumar, R., de Boissiere, T., Gestin, L., Teoh, W. Z., Sotelo, J., de Brebisson, A., Bengio, Y., and Courville, A., "MelGAN: Generative Adversarial Networks for Conditional Waveform Synthesis," in *NeurIPS*, 2019.

[18] Engel, J., Hantrakul, L. H., Gu, C., and Roberts, A., "DDSP: Differentiable Digital Signal Processing," in *Int. Conf. on Learning Representations (ICLR)*, 2020.

[19] Steinmetz, C. J. and Reiss, J. D., "Randomized Overdrive Neural Networks," in *NeurIPS Workshop on Machine Learning for Creativity and Design*, 2020.

[20] Steinmetz, C. J. and Reiss, J. D., "Steerable Discovery of Neural Audio Effects," in *NeurIPS Workshop on Machine Learning for Creativity and Design*, 2021.

[21] Copet, J., Kreuk, F., Gat, I., Remez, T., Kant, D., Synnaeve, G., Adi, Y., and Défossez, A., "Simple and Controllable Music Generation," in *NeurIPS*, 2023.

[22] Barahona-Ríos, A. and Collins, T., "NoiseBandNet: Controllable Time-Varying Neural Synthesis of Sound Effects Using Filterbanks," *IEEE/ACM Transactions on Audio, Speech, and Language Processing (TASLP)*, 32, p. 1573–1585, 2024.

Audio Engineering Society

Conference Paper 33

Presented at the AES International Conference on Machine Learning and Artificial Intelligence for Audio
2025 September 8–10, London, UK

This paper was peer-reviewed as a complete manuscript for presentation at this conference. This paper is available in the AES E-Library (http://www.aes.org/e-lib), all rights reserved. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Journal of the Audio Engineering Society.

Adaptive Neural Audio Mixing with Human-in-the-Loop Feedback: A Reinforcement Learning Approach

Shanshan Zhu[1] and Mohammad Nasim[2]

[1]*University of Illinois Urbana-Champaign, Urbana, IL, USA*
[2]*Northwood University, Midland, MI, USA*

Correspondence should be addressed to Shanshan Zhu (szhu50@illinois.edu)

ABSTRACT

This work introduces a real-time, human-in-the-loop (HITL) audio mixer that integrates reinforcement learning from human feedback (RLHF) and neural source separation to achieve adaptive, personalized audio mixes. Existing automated mixers primarily address the optimization of technical parameters but often cannot understand the nuanced artistic preferences critical to high-quality audio production. In order to close this gap, the framework given combines ongoing user feedback with a reinforcement learning agent's decision-making process, where it is trained with Proximal Policy Optimization (PPO). By employing Demucs in neural source separation, the system obtains precise stem isolation, enhancing mixing controllability and flexibility. Explicit ratings as well as implicit behavioral signals are used as iterative user feedback, which guides the agent towards enhancing its mixing strategy with the passage of time. Experimental tests, using signal-to-distortion ratio (SDR) measurements and user satisfaction surveys, demonstrate significant technical quality and creative match enhancement. This method significantly bridges the gap between automation and artistic control and offers an adaptive and flexible solution for audio engineers and music producers seeking efficiency as well as individuality in audio mixing.

1 Introduction

Digital audio mixing has become an integral component in contemporary music recording, post-production for film, and multimedia applications. Audio engineers have played a significant role throughout history, making subtle adjustments by ear to level, EQ, and space location to provide a cohesive and engaging listening experience. Although skill-demanding, the work is time-consuming and demands careful attention to technical correctness as well as creative intention [1]. Commercial software such as LANDR and iZotope Neutron [2] have incorporated automated mixing capabilities, simplifying workflows and reducing labor. These tools optimize spectral balance but do not integrate real-time reinforcement learning feedback [3]. However, these methods are concerned with technical specifications—such as highest loudness and spectral balance—instead of the fine senses of artwork that human audio engineers naturally consider [4, 5]. Therefore, most systems that automate do not offer the level of creative autonomy and novelty that professional audio engineers require [6, 7].

Concerning these limitations, the integration of Human-in-the-Loop (HITL) methods in machine learning systems has gained popularity in various forms of arts. By

integrating user feedback into repetitive learning loops, HITL models enable machine learning systems to learn about user preferences and adapt accordingly [8, 9]. In audio production, HITL systems have been applied to music creation [4], sound adaptive design [9], and real-time audio effects modulation [10], demonstrating the capability of interactive, user-managed AI systems. The AdaptiveSound system, for example, successfully employed real-time user feedback to significantly enhance the accuracy of audio recognition models, validating the broader applicability of HITL paradigms in sound tasks [11, 12].

Alongside these developments, Reinforcement Learning (RL) has emerged as an effective approach to solving sequential decision-making problems across a broad variety of domains, as well as in audio processing [13, 3]. RL has been applied successfully to tasks such as real-time noise reduction [3, 2], dynamic range compression [13], and adaptive audio effects [10], enabling systems to learn to optimize processing parameters over time. Reinforcement Learning from Human Feedback (RLHF) is a very valuable extension of this method where ratings from the users can be used directly as input to the RL agent's reward function [14, 15]. This integration allows AI systems to be able to learn to communicate human aesthetic emotions, beyond merely objective optimization itself.

In light of these advances, this project introduces an integrated HITL audio mixing system that incorporates neural source separation [16, 17], interactive user interfaces, and RL-based optimization to improve the technical quality and creative coherence of AI-generated audio mixes. The main research question for this project is:

> *"Can reinforcement learning from human feedback (RLHF) improve the quality and tailoring of AI-generated audio mixes over time?"*

The program leverages Demucs [16] as the source separator and PPO [14] as the adaptation learner, benefiting from industry-top utilities to develop a secure, user-focused solution. From iterative inclusion of user feedback, the system not only improves audio clarity and balance but also learns the users' idiosyncratic aesthetic preferences, making it an adaptable and easy-to-use instrument for creative audio mixing available to both

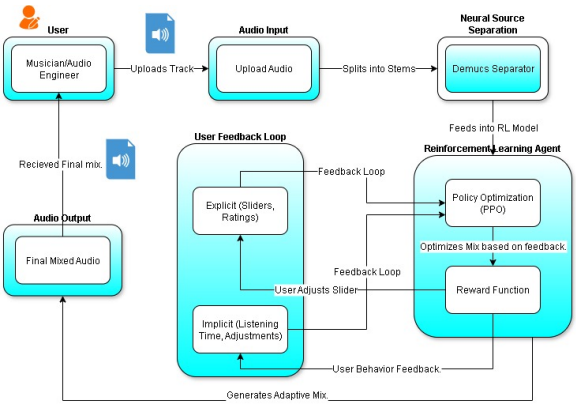


Fig. 1: The Human-in-the-Loop (HITL) Adaptive Audio Mixing System. The diagram outlines the interaction between user feedback, reinforcement learning, and neural source separation for adaptive mixing.

novice and professional engineers (see Figure 1 for a high-level overview).

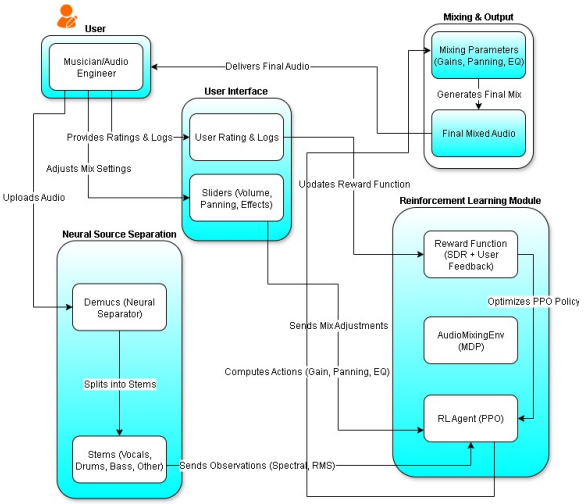
We test two hypotheses: (H1) RLHF significantly improves signal-to-distortion ratio (SDR) versus automated mixing; (H2) user satisfaction rises over repeated sessions as the system adapts to preferences. These capture RLHF's effect on both technical and perceptual outcomes.

2 Methodology

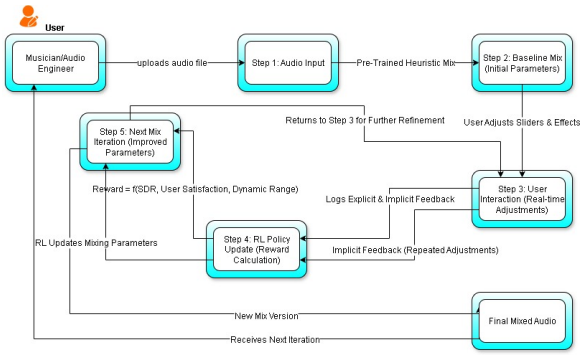
The following section provides a detailed explanation of the proposed human-in-the-loop (HITL) adaptive audio mixing architecture, such as the core system structure, its end-to-end pipeline, reinforcement learning (RL) approach, and most critically scalability features. The method focuses on combining neural source separation, real-time user feedback, and policy optimization in a loop to guarantee technical correctness and creative consistency in audio mixing.

2.1 Core System Architecture

The system (see Figure 2a for system architecture and Figure 2b for workflow) is constructed with three key modules—an AI-powered source separation and mixing module, a human feedback interface, and a reinforcement learning module—each of which is involved



(a) System Architecture: AI-Driven Source Separation & RL-Based Mixing.



(b) Iterative Workflow: Human-in-the-Loop Adaptive Mixing Process.

Fig. 2: Overview of the HITL Adaptive Audio Mixing System. (a) The system architecture integrates neural source separation, user feedback, and reinforcement learning for optimized audio mixing. (b) The iterative workflow depicts the user interaction and reinforcement learning process for continuous improvement.

in an adaptive process flow that progressively optimizes audio mixes.

In the first module, AI-Driven Mixing and Source Separation, we use Demucs [2, 16, 18], which is a deep neural network that is best suited to separating individual stems (vocals, drums, bass, and the rest) from a multi-track composite. This level of separation at the granular level gives the system high-granularity control of each stem's gain, pan, and EQ. Once the stems are extracted, initial mixing parameters are determined using pre-trained heuristics to create a default mix that balances loudness versus spatial location [7]. These heuristics take typical loudness levels and typical pan settings for an assortment of instruments as a starting point for eventual fine-tuning.

The second component, the Human Feedback Interface, is built using a real-time web application framework (e.g., Streamlit). This interface provides intuitive controls—volume and pan sliders, and effect switches like compression or reverb—that allow stems to be edited in real time. Similar interactive approaches have been explored in music information retrieval systems [19] and assistive technologies [12]. To support informed decision-making, the interface displays visual aids, including time-domain waveforms and spectrograms. During the user's interactions, every parameter change (explicit feedback) and repeated re-adjustment (implicit feedback) is continuously logged. This dual capture of user data ensures that the system can distinguish between direct preferences (e.g., volume slider set to 3 dB) and exploratory behaviors (e.g., frequent oscillation around a particular gain setting).

Finally, the Reinforcement Learning Module employs Proximal Policy Optimization (PPO) [14], one of the policy-gradient algorithms that is well known to be stable with comparatively low variance during training. Implemented via Stable-Baselines3, this module processes both audio features and user feedback signals, optimizing its policy to achieve maximum technical performance metrics (e.g., Signal-to-Distortion Ratio, or SDR) and user satisfaction [13]. The RL agent is executed in a specially crafted environment named *AudioMixingEnv*, which includes the mixing state, allowed actions, and reward computation. By passing user feedback on to the reward signal, the agent learns to produce mixes that simultaneously meet quantitative quality standards and agree with subjective aesthetic taste.

2.2 Workflow

As shown in Figure 2b, the adaptive HITL mixing process unfolds in the following sequence:

1. **Input and Source Separation:** Users upload either multi-track or stereo audio. For stereo-only input, Demucs isolates distinct stems (e.g., vocals, bass, drums), enabling fine-grained control.

2. **Baseline AI Mix:** The system initializes a mix using typical gain/pan heuristics, providing a neutral starting point that can be refined to meet creative aims.

3. **User Interaction and Feedback Logging:** Through an interactive interface, users adjust parameters such as volume, panning, EQ, and optional effects like reverb. These changes are automatically logged. Explicit feedback includes direct parameter settings (e.g., vocals to 4 dB), while implicit feedback arises from repeated or exploratory adjustments.

4. **Reinforcement Learning Adaptation:** Following each user interaction, the RL agent updates its policy. The reward function merges technical metrics (e.g., SDR, dynamic range) with user satisfaction, guiding the system to produce a revised mix that better matches evolving preferences.

5. **Iterative Improvement:** Over multiple rounds, the agent refines its policy, striving to enhance both clarity and artistic alignment. Each subsequent session benefits from prior learning, forming a feedback loop that continually optimizes the overall mixing strategy.

2.3 Reinforcement Learning Framework for Adaptive Mixing

At the core of this system's adaptation lies the *AudioMixingEnv*, which defines the audio mixing task as a Markov Decision Process (MDP). A Markov Decision Process provides a mathematical framework for decision-making in environments with stochastic outcomes, where future states depend only on the current state and action taken.

The reinforcement learning model optimizes mixing decisions using Proximal Policy Optimization (PPO), a policy-gradient algorithm that iteratively updates decision probabilities based on received rewards. PPO ensures stable learning by preventing drastic changes in policy, making it suitable for scenarios with noisy or evolving feedback.

Symbolically, let s_t be a state at time t, a_t an action of the RL agent at time t, and r_t a reward after each action. The transition dynamics (i.e., $p(s_{t+1} \mid s_t, a_t)$) preserve the audio mix differences as a result of the parameter variations. Here are the state, action, and reward spaces:

- **State Representation:** The observation vector s_t encodes time-varying features of each audio stem, such as root mean square (RMS) energy, spectral centroid, dynamic range indicators, and current panning settings. These features are updated every time the user makes a change or after a mixing iteration is completed.

- **Action Space:** The RL agent takes continuous-valued actions a_t that correspond to adjustments of volume, panning, EQ parameters, and optional effects controls. For instance, an action might incrementally increase the vocal stem's volume while slightly narrowing the stereo field of the drum stem.

2.4 Reward Function and Optimization

2.4.1 Reward Components

The reward function used to train the RL agent combines objective technical metrics with subjective human preferences. As defined in Equation 1, the total reward r_t at timestep t is a weighted sum of three components:

$$r_t = \alpha\,\mathrm{SDR}(t) + \beta\,\mathrm{UserScore}(t) + \gamma\mathrm{DynRange}(t) \tag{1}$$

- **SDR (Signal-to-Distortion Ratio)** quantifies the clarity and separation of audio stems, serving as an objective benchmark for technical mix quality.

- **UserScore** captures user preferences from both explicit Likert-scale ratings and implicit interaction behavior (e.g., repeated changes, time spent on adjustments).

- **Dynamic Range (DynRange)** penalizes over-compression and ensures the preservation of natural amplitude variation within the mix.

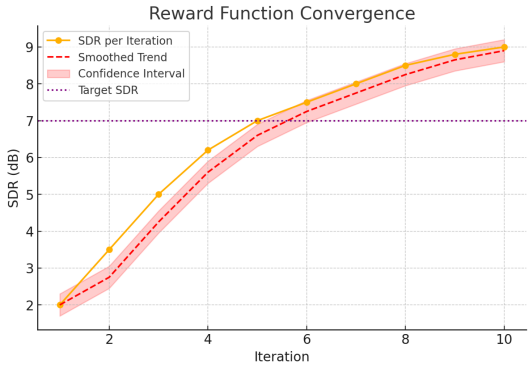


Fig. 3: Reward Function Convergence Plot. The reward score increases over training iterations, indicating successful policy optimization. The curve reflects cumulative progress in balancing Signal-to-Distortion Ratio (SDR), user satisfaction (UserScore), and dynamic range preservation across sessions.

2.4.2 Weight Optimization

To determine optimal weights for the reward components—Signal-to-Distortion Ratio (SDR), UserScore, and Dynamic Range—we conducted a grid search over the following candidate values:

The grid search was run over: $\alpha \in \{0.3, 0.5, 0.7\}$, $\beta \in \{0.2, 0.4, 0.6\}$, $\gamma \in \{0.1, 0.3, 0.5\}$.

Each parameter combination was evaluated via 3-fold cross-validation over 90 training episodes using a multi-objective criterion: the combined average of final SDR and user satisfaction scores. Table 1 summarizes the tested combinations and their corresponding performance.

Table 1: Grid Search Results for Reward Function Weights

α	β	γ	Avg. SDR (dB)	Avg. Satisfaction (/5)
0.3	0.2	0.1	6.9	3.7
0.5	0.4	0.3	**9.0**	**4.5**
0.7	0.2	0.1	9.2	3.2
0.5	0.6	0.1	8.7	4.0
0.3	0.4	0.5	7.1	4.4

The final selected configuration was:

$$\alpha = 0.5, \quad \beta = 0.4, \quad \gamma = 0.3$$

This setting achieved the best trade-off between objective mix quality (highest SDR) and user satisfaction (highest Likert scores). Notably, larger values of α (> 0.6) produced technically cleaner but emotionally sterile mixes, while overly high β (> 0.4) led to inconsistent outcomes due to subjective feedback noise. The chosen configuration avoids these extremes and aligns well with the system's goal of human-aligned technical refinement.

2.4.3 Convergence Criteria

The agent was trained using Proximal Policy Optimization (PPO) until the reward curve plateaued across sessions. Convergence was defined by:

- A moving average of total reward $\bar{r}_t$ over 5 sessions that changed by less than 0.01.
- Signal-to-Distortion Ratio (SDR) improvement of less than 0.2 dB between sessions n and $n+1$.
- UserScore stabilizing within 5%.

As shown in Figure 3, the reward function converged smoothly, with diminishing returns observed after session 7. This indicates the agent had sufficiently internalized both objective metrics and aesthetic cues.

2.5 Reinforcement Learning Algorithm Details

2.5.1 Proximal Policy Optimization (PPO) Implementation

Proximal Policy Optimization (PPO) iteratively improves the agent's policy $\pi_\theta(a \mid s)$ by sampling batches of trajectories from user interactions and optimizing a clipped objective function that limits overly large policy updates. This approach fosters stable learning even under potentially noisy or inconsistent user feedback. Each round of gradient updates nudges the mixing parameters toward states that yield higher rewards—i.e., improved SDR, maintained dynamic range, and higher user satisfaction.

The PPO algorithm ensures that the reinforcement learning model adapts smoothly by limiting drastic updates to policy parameters. Through this iterative approach, the system progressively aligns its mixing decisions with both technical metrics (SDR, dynamic range) and subjective user preferences, ultimately achieving a refined, user-tailored mix.

Algorithm 1 PPO-Based Policy Update per User Interaction

Input: π_θ (policy network with parameters θ), $\pi_{\theta_{old}}$ (old policy network, initially set to π_θ), *AudioMixingEnv* E (includes user feedback logging), T (number of timesteps per episode), Hyperparameters $\{\alpha, \beta, \gamma\}$ (weights for reward terms), ε (PPO clip parameter), K (number of training epochs)

for each user-interaction episode **do**
  Reset environment E; obtain initial state s_0
  for $t = 0$ to $T-1$ **do**
    Step 1: Sample action from current policy
    $a_t \sim \pi_\theta(a \mid s_t)$
    Step 2: Environment applies action & logs user feedback
    $s_{t+1}, r_t = E.step(a_t)$
    // r_t incorporates SDR, dynamic range, and user-derived feedback
    Store transition (s_t, a_t, r_t, s_{t+1}) in replay buffer
  end for
  Step 3: Compute advantages and update policy
  for each sample batch in replay buffer **do**
    Estimate advantages $\hat{A}_t$ using GAE or similar
    Update policy π_θ by maximizing PPO clipped objective:

$$L(\theta) = E_t\left[\min\left(r_t(\theta), clip(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\right) \times \hat{A}_t\right]$$

  end for
  Step 4: Update old policy $\pi_{\theta_{old}} \leftarrow \pi_\theta$
end for

2.6 Scalability and Performance Considerations

The system is optimized in several key areas to accommodate real-time interaction. A FastAPI backend handles concurrency and file operations, while GPU acceleration enables low-latency inference for high-resolution audio input. Containerized deployment (e.g., Docker) makes horizontal scaling possible, allowing additional instances of the service to be spun up as user traffic increases.

The system also supports user-specific profile caching. Each user's previous interactions, blending preferences, and most used parameter ranges are cached and reused in later sessions to reduce the agent's warm-up time. With user-specific states, the system can make more personal recommendations quicker and at lower computational cost, thereby supporting the workflow of repeat users.

All of these design elements are combined such that the proposed HITL audio mixing platform can comfortably handle a number of concurrent sessions without sacrificing responsiveness or quality. With increasing numbers of users, further optimization strategies such as distributed training of RL or more advanced load balancing can be incorporated to ensure real-time performance at scale.

2.7 Evaluation Metrics

To evaluate the technical and perceptual performance of the HITL audio mixing system, we employed four key metrics:

- **Signal-to-Distortion Ratio (SDR):** Measures the clarity and separation between audio stems. Higher SDR values indicate less distortion and better signal quality [2].
- **Dynamic Range:** Assesses the natural amplitude variation within a track. Excessive compression reduces dynamic range, while a balanced range preserves musical expressiveness [13].
- **Spectrogram Analysis:** Visualizes frequency energy distribution before and after mixing. It reveals whether the system successfully reduces masking or spectral overlap between stems.
- **User Satisfaction:** Captured through 5-point Likert-scale ratings and mapped to percentages (e.g., 3/5 = 60%). Combined with open-ended feedback, this metric assesses subjective alignment with users' creative preferences.

3 Experimental Setup

In order to assess both technical and subjective performance, we conducted controlled tests with 40 participants spanning various skill levels in audio engineering. Each participant undertook a series of ten iterative mixing sessions on musical excerpts (45–60 s) covering genres like pop, jazz, electronic, and orchestral.

3.1 Participant Groups and Setup

Participants were randomly divided into two cohorts: one using our HITL system powered by Reinforcement Learning from Human Feedback (RLHF), and another using a baseline commercial tool (iZotope Neutron) without adaptive learning. In the HITL group, an RL agent continuously updated its policy based on users' volume, panning, and EQ changes, along with explicit ratings and implicit behavioral signals [2, 3]. By contrast, baseline participants relied solely on predefined automated processes. Each mixing session required users to refine a short multi-track or stereo file; when only stereo was available, the system (in the HITL condition) automatically separated stems via Demucs [16] for granular adjustments.

3.2 Experimental Procedure

At the beginning of the study, participants received a brief tutorial on their assigned mixing interface. Those in the HITL group were shown how the reinforcement learning system would adapt to their interactions over multiple iterations, while the baseline group used a static automated mixer without any adaptive feedback. Each participant completed ten short mixing sessions (45–60 s of audio), adjusting parameters (e.g., volume, panning, effects) according to personal preferences.

After each session, participants rated the final mix on a 5-point Likert scale, also leaving qualitative comments about clarity, balance, and artistic appeal.

We used four key metrics to gauge mix quality: (1) Signal-to-Distortion Ratio (SDR) for measuring clarity and stem separation [2, 3, 17], (2) Dynamic Range to track how compression or gain adjustments affected musical expressiveness [13, 7], (3) Spectrogram Analysis to visualize how the system reduced unwanted overlaps in critical frequency bands [18, 19], and (4) User Satisfaction measured via a 5-point Likert scale, later mapped to 0–100%, reflecting subjective perceptions of clarity and creative alignment [6, 12, 8]. Combining these metrics offered a multifaceted view of mix quality, indicating that higher SDR and balanced amplitude distributions improved clarity without sacrificing expressiveness [7]. In parallel, user feedback and Likert-scale scores demonstrated the system's responsiveness to individual aesthetic needs, validating that the RLHF approach is both technically robust and perceptually appealing.

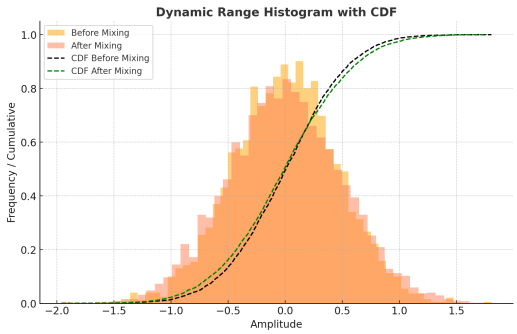


Fig. 4: Dynamic range histogram with CDF showing preservation of expressiveness without over-compression.

To evaluate the effectiveness of iterative reinforcement learning, we tracked reward convergence (Figure 3) and conducted a paired t-test to assess SDR improvements over the ten-session process. Since user satisfaction scores were based on a Likert scale, we applied a Wilcoxon signed-rank test to analyze changes in subjective ratings. Together, these objective and subjective measures offered a rigorous assessment of how effectively the RLHF approach balanced technical audio quality with user-specific aesthetic goals.

4 Results

4.1 Subjective and Objective Results

The HITL group demonstrated consistent improvements across both objective and subjective metrics.

4.1.1 Signal-to-Distortion Ratio (SDR)

SDR steadily increased from 6.8 dB to 9.0 dB over ten sessions (Table 2), indicating enhanced clarity from RLHF.

4.1.2 Dynamic Range Preservation

Figure 4 shows that dynamic range slightly improved (std. dev. increased from 0.46 to 0.50), avoiding over-compression while maintaining expressiveness. The cumulative distribution function (CDF) confirms that extremes were not clipped.

4.1.3 User Satisfaction Trends

User satisfaction in the HITL group rose from 60% (3.0/5) to 90% (4.5/5)(Table 2). The baseline group remained around 70%. Participants reported less manual correction needed and greater creative alignment in later sessions.

Table 2: Statistical Analysis of SDR Improvement and User Satisfaction Increase

Hypothesis	Pre-Mean (SD)	Post-Mean (SD)	Test Used	t/Z Value	p-Value
H1 (SDR Improvement)	6.8 (0.57) dB	9.0 (0.42) dB	Paired t-test	$t(9) = 6.23$	$p < 0.001$
H2 (User Satisfaction Increase)	3.0 (0.62) / 5	4.5 (0.55) / 5	Wilcoxon signed-rank test	$Z = 2.81$	$p = 0.002$

4.1.4 Statistical Validation

Table 2 summarizes the statistical results. A paired t-test confirmed the SDR improvement ($p < 0.001$), while a Wilcoxon signed-rank test verified the increase in user satisfaction ($p = 0.002$).

4.1.5 Between-Group Comparison

To directly assess the superiority of the HITL group over the baseline condition, we performed independent samples t-tests on the final-session values of SDR and user satisfaction. Table 3 reports the results, including p-values and effect sizes (Cohen's d).

Table 3: Between-Group Comparison of Final Performance Metrics

Metric	HITL (Mean SD)	Baseline (Mean SD)	p-value	Cohen's d
SDR (dB)	12.3 2.1	9.8 1.9	< 0.001	1.26
User Satisfaction (/5)	4.2 0.8	3.1 0.9	< 0.001	1.31

The 95% confidence interval for SDR difference between groups was [1.4, 3.2] dB, while the CI for user satisfaction was [0.7, 1.6] Likert points. Both comparisons yielded large effect sizes (Cohen's $d > 1.2$), indicating that the RLHF framework substantially outperformed the baseline in both technical accuracy and perceptual alignment.

4.1.6 Audio Examples and Spectrograms

To further illustrate the differences between the baseline and HITL RLHF-based mixing systems, we provide supplementary audio examples showcasing various genres and input conditions. A curated set of before/after audio samples is hosted at: **`https://adaptive-neural-audio-mixing.streamlit.app`**

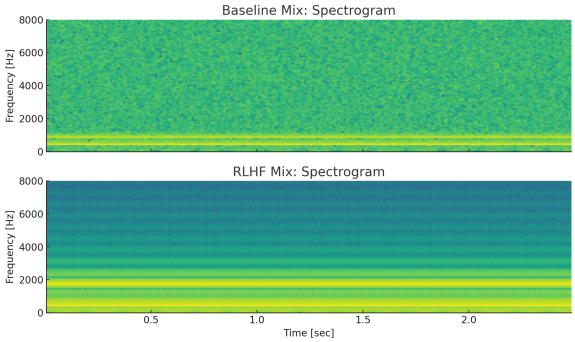


Fig. 5: Spectrogram comparison between baseline and HITL-RLHF mix. (Top) Baseline: notable masking in mid-frequency bands. (Bottom) RLHF-enhanced: improved separation and spectral clarity.

These examples allow listeners to directly compare the improvements in stem balance, clarity, and creative alignment across systems. Each example is paired with spectrogram visualizations to highlight changes in frequency distribution and dynamic range (see Figure 5).

5 Limitations

Although the proposed framework significantly improves mix clarity and user satisfaction, some limitations still remain. Firstly, *feedback variability* complicates learning since users occasionally enter exploratory or inconsistent input rather than deliberate, systematic changes [14, 3]. While the reward function encompasses explicit ratings and implicit behavior, future iterations could include *confidence-weighted feedback* or normalization for handling noisy signals [10, 20]. Second, *scalability* is a problem at high concurrency; while GPU acceleration reduces latency, performance is reduced if concurrent sessions are multiple. *Genre-specific biases* also came to the forefront, live jazz and orchestral proving harder to distinguish than pop or electronic, necessitating *genre-diverse training data* or personalized mixing profiles [16, 7]. Finally, existing measurements mostly deal with *SDR* and user enjoyment and potentially neglect psychoacoustic and emotional components. Imbedding perceptual tools like *PEAQ* could offer a more holistic perspective of the listening experience.

6 Future Work

Several advancements can calibrate and stretch such a HITL approach to audio mixing. First, *complex feedback processes* like eye-tracking or physiological feedback can better aid reinforcement learning updates, enabling user intent to be recorded in a more advanced manner [11, 8]. Incorporating *multi-objective RL* with methods like Bayesian reward shaping can maximize user desires, blend comprehensibility, and dynamic range preservation [13]. After scalability improvements, *multi-user collaboration* will facilitate simultaneous editing by multiple users, replicating commercial studio workflows. Meanwhile, *genre-specific mixing* profiles can be learned to handle stylistic subtleties in classical or jazz recording. Further additions to *mastering tasks* (e.g., final loudness normalisation, stereo imaging) can provide an end-to-end integrated solution. Lastly, researching *generative AI* in order to include creative effects or style variations can boost expressiveness and customize the ultimate mix to different artistic visions [21].

7 Discussion

Our experiments confirm that reinforcement learning from human feedback (RLHF) benefits both technical and subjective aspects of audio mixing [15, 8]. By iteratively capturing user intent, the system outperforms static, rule-based methods, delivering improved clarity (as evidenced by higher SDR) while allowing creative flexibility [6, 12]. Nonetheless, *inconsistent user input* and *reward design* remain pivotal challenges, indicating that further refinements—e.g., confidence-weighted or genre-informed feedback—may be necessary to achieve stable, high-quality mixes across diverse contexts [10, 20]. These observations highlight the broader potential of interactive, user-driven AI in audio production, where automation is not just a technical optimizer but an intelligent, adaptable collaborator.

8 Conclusion

In this study, we have presented a novel human-in-the-loop (HITL) audio mixing system that synthesizes *neural source separation*, *interactive user interfaces*, and *reinforcement learning from human feedback* [1, 19, 15]. Our experiments confirm that iterative user involvement significantly improves both objective metrics, such as SDR, and subjective user satisfaction. In uncovering artistic preferences in real time, the system shifts away from rigid rules-based automation and toward a more collaborative engine—a system that learns and refines its recommendations on the basis of user feedback. This approach is hopeful for bridging the creative potential of AI in music-making, from fine-grained aesthetic adjustments to fuller automation [22, 21]. As future work tackles multi-user environments, genre-specific optimizations, and new feedback modalities, AI-augmented mixing has the potential to become an even more expressive and intuitive tool for both content creators and audio engineers.

References

[1] Reiss, J. D. and Øyvind Brandtsegg, "Applications of Cross-Adaptive Audio Effects: Automatic Mixing, Live Performance, and Everything in Between," *Frontiers in Digital Humanities*, 5(17), pp. 1–16, 2018.

[2] Uhlich, S., Porcu, M., Giron, F., Enenkl, M., Kemp, T., Takahashi, N., and Mitsufuji, Y., "Improving Music Source Separation Based on Deep Neural Networks Through Data Augmentation and Network Blending," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 261–265, 2017.

[3] Luo, Y., Chen, Z., Hershey, J. R., Roux, J. L., and Mesgarani, N., "Deep Clustering and Conventional Networks for Music Separation: Stronger Together," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 2017, pp. 61–65, 2017.

[4] Sturm, B. L. and Ben-Tal, O., "Taking the Models back to Music Practice: Evaluating Generative Transcription Models built using Deep Learning," *Journal of Creative Music Systems*, 2(1), pp. 1–15, 2017.

[5] Pulkki, V., "Virtual Sound Source Positioning Using Vector Base Amplitude Panning," *Journal of Audio Engineering Society*, 45(6), pp. 456–466, 1997.

[6] Amershi, S., Cakmak, M., Knox, W. B., and Kulesza, T., "Power to the People: The Role

of Humans in Interactive Machine Learning," *AI Magazine*, 35(4), pp. 105–120, 2014.

[7] Pulkki, V., "Spatial Sound Reproduction with Directional Audio Coding," *Journal of Audio Engineering Society*, 55(6), pp. 503–516, 2007.

[8] Mosqueira-Rey, E., Hernández-Pereira, E., Alonso-Ríos, D., Pazos-López, S., and Iglesias, R., "Human-in-the-Loop Machine Learning: A State of the Art," *Artificial Intelligence Review*, 56(4), pp. 3005–3054, 2023.

[9] Do, H., Dang, Q., Huang, J. Z., and Jain, D., "AdaptiveSound: An Interactive Feedback-Loop System to Improve Sound Recognition for Deaf and Hard of Hearing Users," in *Proceedings of the 25th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '23)*, volume abs/1911.13254, pp. 18:1–18:12, 2023.

[10] Fails, J. A. and Jr., D. R. O., "Interactive Machine Learning," in *Proceedings of the 8th International Conference on Intelligent User Interfaces*, volume 8, pp. 39–45, 2003.

[11] Katan, S., Grierson, M., and Fiebrink, R., "Using Interactive Machine Learning to Support Interface Development Through Workshops with Disabled People," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*, 1, pp. 251–254, 2015.

[12] Yuksel, B. F., Fazli, P., Mathur, U., Bisht, V., Kim, S. J., Lee, J. J., Jin, S. J., Siu, Y.-T., Miele, J. A., and Yoon, I., "Human-in-the-Loop Machine Learning to Increase Video Accessibility for Visually Impaired and Blind Users," in *Proceedings of the 2020 ACM Designing Interactive Systems Conference (DIS '20)*, 1, pp. 47–60, 2020.

[13] Drossos, K., Virtanen, T., Schuller, G., and Mimilakis, S., "Deep Neural Networks for Dynamic Range Compression in Mastering Applications," in *Proceedings of the 2016 International Conference on Digital Audio Effects (DAFx-16)*, pp. 1–6, 2016.

[14] Lambert, N., "Reinforcement Learning from Human Feedback," *ArXiv preprint*, abs/2504.12501, 2025.

[15] Chai, Y., Sun, H., Fang, H., Wang, S., Sun, Y., and Wu, H., "MA-RLHF: Reinforcement Learning from Human Feedback with Macro Actions," 2025.

[16] D'efossez, A., Usunier, N., Bottou, L., and Bach, F. R., "Music Source Separation in the Waveform Domain," 2019.

[17] Grais, E. M. and Plumbley, M. D., "Single channel audio source separation using convolutional denoising autoencoders," in *2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pp. 1265–1269, 2017.

[18] Stoller, D., Ewert, S., and Dixon, S., "Wave-U-Net: A Multi-Scale Neural Network for End-to-End Audio Source Separation," 2018.

[19] Pons, J. and Serra, X., "musicnn: Pre-Trained Convolutional Neural Networks for Music Audio Tagging," 2019.

[20] Chen, Q. Z., Schnabel, T., Nushi, B., and Amershi, S., "HINT: Integration Testing for AI-Based Features with Humans in the Loop," in *Proceedings of the 27th International Conference on Intelligent User Interfaces (IUI '22)*, 1, pp. 1–17, 2022.

[21] Qian, J., Liu, X., Yu, Y., Jiang, Q., Wang, H., and Li, H., "Stripe-Transformer: Deep Stripe Feature Learning for Music Source Separation," *Journal of Audio, Speech, and Music Processing*, 2023(2), pp. 1–14, 2023.

[22] Justus, A. A., "Music Generation Using Human-In-The-Loop Reinforcement Learning," 2023.

Audio Engineering Society

Conference Paper 34

Presented at the AES International Conference on Machine Learning and Artificial Intelligence for Audio
2025 September 8–10, London, UK

This paper was peer-reviewed as a complete manuscript for presentation at this conference. This paper is available in the AES E-Library (http://www.aes.org/e-lib), all rights reserved. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Journal of the Audio Engineering Society.

NablAFx: A Framework for Differentiable Black-box and Gray-box Modeling of Audio Effects

Marco Comunità[1], Christian J. Steinmetz[1], and Joshua D. Reiss[1]

[1]*Centre for Digital Music, Queen Mary University of London, UK*

Correspondence should be addressed to Marco Comunità (m.comunita@qmul.ac.uk)

ABSTRACT

We present NablAFx, an open-source framework developed to support research in differentiable black-box and gray-box modeling of audio effects. Built in PyTorch, NablAFx offers a versatile ecosystem to configure, train, evaluate, and compare various architectural approaches. It includes classes to manage model architectures, datasets, and training, along with features to compute and log losses, metrics and media, and plotting functions to facilitate detailed analysis. It incorporates implementations of established black-box architectures and conditioning methods as well as differentiable DSP blocks and controllers, enabling the creation of both parametric and non-parametric gray-box signal chains. Beside established conditioning methods like concatenation, feature-wise linear modulation (FiLM) and temporal feature-wise linear modulation (TFiLM), we propose three further methods: time-varying concatenation (TVCond), tiny TFiLM (TTFiLM) and time-varying FiLM (TVFiLM), as efficient implementations of time-varying conditioning similar to TFiLM. We also propose the Static Rational Linearity as a flexible and efficient differentiable processor to learn nonlinear functions. The code is accessible at https://github.com/mcomunita/nablafx.

1 Introduction

Audio effects are central for engineers and musicians to shape timbre, dynamics, and spatialisation of sound [1]. Therefore, research related to audio effects, especially with the success of deep learning and differentiable digital signal processing (DDSP) [2], is a very active field [3]. This includes applications such as classification and identification [4], parameters estimation [5, 6], modeling [7, 8], style transfer [9, 10], automatic mixing [11, 12]. Audio effects modeling is one of the most active applications of differentiable approaches, with the majority of methods falling into black-box (i.e., neural networks) and gray-box (i.e., DDSP) paradigms. While black-box models achieve state-of-the-art accuracy [7, 13–15] there is interest in gray-box ones [16–20] due to interpretability and potential for efficiency.

Comparing modeling paradigms remains challenging due to significant variations in training and evaluation methods. In addition, the lack of standardized implementations for models and DDSP blocks further impedes reproducibility and performance assessment. There are a growing number of audio effect implementations available to researchers, however existing options remain limited in a number of ways (see Table 1).

While the `Spotify Pedalboard`[1] library offers Python implementations of common audio effects and allows to define signal chains, these are not differentiable. DDSP, introduced in [2], provides some differentiable blocks[2], though they are neither common nor

[1] github.com/spotify/pedalboard
[2] github.com/magenta/ddsp

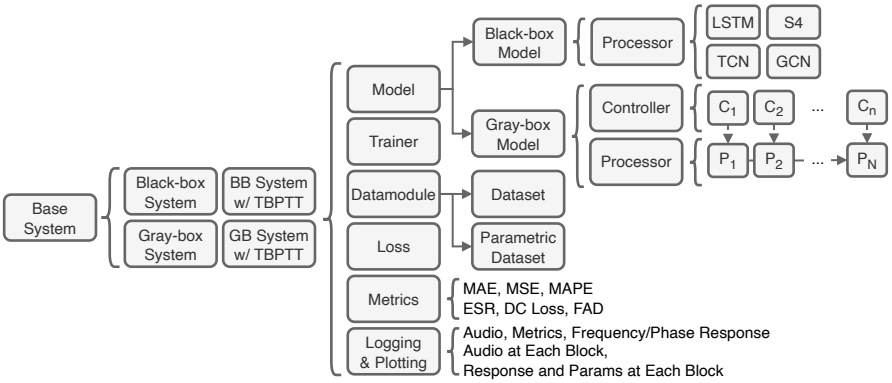


Fig. 1: Overview of the NablAFx framework for audio effects modeling

Table 1: Python libraries for processing/modeling applications. We show if: they include differentiable (Diff.) implementations, neural networks (NN), DSP processors (Proc.) and controllers (Contr.), they allow to define signal chains and include analysis tools.

Library	Diff.	NN	Proc.	Contr.	Chains	Analysis
`Pedalboard`	✗	✗	✓	✗	✓	✗
`DDSP`	✓	✗	✓	✗	✗	✗
`dasp`	✓	✗	✓	✗	✗	✗
`diffmoog`	✓	✗	✓	✗	✓	✗
`GRAFX`	✓	✗	✓	✗	✓	✗
`pyneuralfx`	✓	✓	✗	✗	✗	✓
`NablAFx`	✓	✓	✓	✓	✓	✓

easily reusable, since they are focused on specific applications within audio synthesis. `dasp`[3][9] includes differentiable implementations of common processing and mixing blocks, and while useful when imported into larger projects, the library in not meant to define signal chains. Interconnections of processors can be defined in `diffmoog`[4][21], although mainly focused on FM synthesis and not suitable for effects modeling. Also `GRAFX`[5] [22] enables complex interconnections, but lacks external control, limiting parametric, time-varying, and modulated signal chains for effects modeling.

`pyneuralfx`[6][23] is the only framework designed for modeling and, while it includes state-of-the-art neural networks, it focuses only on black-box approaches and does not include time-varying models [7]. Even though it provides functions for inference-time analysis, it lacks logging and plotting features during training and testing. Also, experiment configurations are hard to modularize and adapt to different datasets, models, or training procedures, limiting repeatability and comparison.

To address these limitations and advance differentiable audio effects modeling, we propose **NablAFx**, which provides:

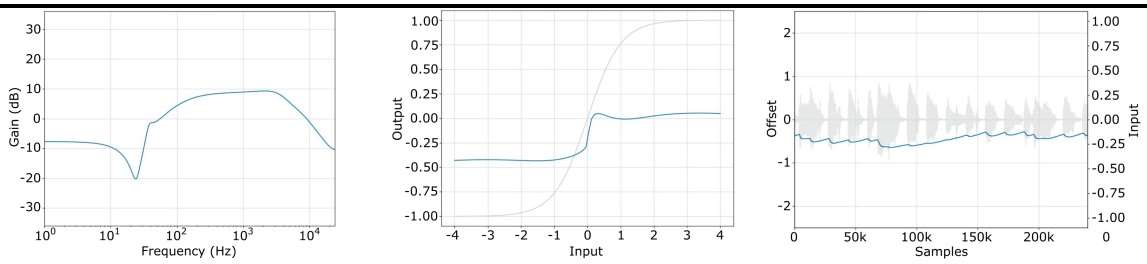
- Black-box architectures and condition methods: concatenation, FiLM, TFiLM, TVCond, TTFiLM, TVFiLM
- Gray-box architectures composed of differentiable processors, and differentiable controllers for parametric/non-parametric models.
- Modules to manage datasets, training, and loss functions.
- Tools to log metrics and media during training and testing.
- Plotting functions for analysis throughout training.

2 Framework

NablAFx is a framework for audio effects modeling that allows researchers to easily define, train, evaluate and compare differentiable black-box and gray-box models. As shown in Fig. 1, it integrates models, datasets, trainers, loss functions, metrics, and logging/plotting tools. Built with PyTorch Lightning[7], it leverages Weights&Biases[8] to log results and media.

System — In NablAFx all necessary functionalities are contained in an audio effects modeling system class. The *BaseSystem* class handles the initialization of loss functions, optimizers, learning rate scheduler, metrics, and includes shared methods to compute and log loss, metrics, audio and frequency/phase response. The *BaseSystem* is divided into *BlackBoxSystem* and *GrayBoxSystem*, which initialize black-box and gray-box

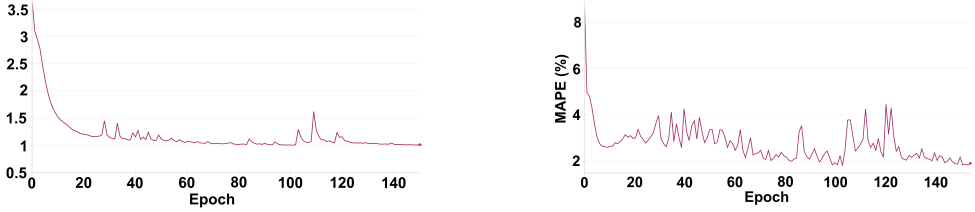
[3] `github.com/csteinmetz1/dasp-pytorch`
[4] `github.com/aisynth/diffmoog`
[5] `github.com/sh-lee97/grafx`
[6] `github.com/ytsrt66589/pyneuralfx`
[7] `lightning.ai/pytorch-lightning`
[8] `wandb.ai/site`



(a) Parametric EQ: frequency response learned during training

(b) Nonlinearity: amplitude response learned during training (blue) compared with *tanh* (gray)

(c) Time-varying DC offset: offset (blue) as a function of time and the input signal (gray)



(d) Loss: L1 + MR-STFT

(e) Metrics: MAPE

Fig. 2: Examples of plotting features included in NablAFx

models, respectively, and implement train, validation, and test steps. The *GrayBoxSystem* adds methods to log audio output, plot/log frequency and time responses, and parameters values for each stage of the signal chain. Both systems are extended with *WithTBPTT* classes, which implement truncated backpropagation through time to enable faster training of recurrent networks[13].

Model — In our framework, black-box models can be any neural network - with outputs defined as a function of input and controls $y = f(x,c)$ - represented by the *Processor* class in *BlackBoxModel*. Gray-box models comprise interconnected differentiable blocks, forming a function composition: $y = (f_1 \circ f_2 \circ \ldots \circ f_N)(x,c)$, and the *Processor* class defines a chain of processors. A *Controller* class defines a chain of controllers, each associated with a processor, allowing the definition of parametric and time-varying models that are a function of both input audio and controls.

Data — The *DataModule* class takes care of initializing the dataset and dataloaders for train, validation and testing. *AudioEffectDataset* and *ParametricAudioEffectDataset* classes are used to manage data for non-parametric and parametric models.

Metrics — Metrics are computed with: *torchmetrics*[9] for mean absolute error (MAE), mean squared error (MSE) and mean absolute percentage error (MAPE); *auraloss*[10][24] for error-to-signal ratio (ESR) and DC loss; and the *frechet-audio-distance*[11] package for Frechét Audio Distance (FAD) [25].

Plotting — In addition to logging losses, metrics and audio examples, we provide methods to plot and log frequency/phase response for the whole system, as well as frequency/time response and parameters values for each DDSP block in a gray-box system. We offer two methods to compute the frequency and phase response: one using an exponential sine sweep[12] [26], suitable for linear and mildly nonlinear systems, and a custom method designed for nonlinear systems. The latter measures the system's response in steps, using sinusoidal inputs at exponentially spaced frequencies. To ensure reliable measurements, each sinusoid lasts several seconds for the system to reach steady state, with magnitude/phase response computed only from the final segment as a function of the minimum frequency.

$$x = x[-T \cdot \lfloor f_s/f_1 \rfloor :]$$
$$y = y[-T \cdot \lfloor f_s/f_1 \rfloor :]$$

where T is the signal duration (e.g. 5 s), f_s is the sample rate and f_1 is the minimum frequency of the stepped sweep (e.g., 10 Hz). Fig. 2a shows the frequency response of a Parametric EQ block, while Fig. 2b and 2c display examples of learned nonlinearity (vs. *tanh*, light gray) and time-varying DC offset (vs. input signal, light gray). In Fig. 2d and 2e we also show examples of plots for loss (L1 + MR-STFT) and metrics (MAPE) during training, while in Fig. 3 we show and example of frequency and phase responses measured after training using our proposed method for highly nonlinear effects.

[9] lightning.ai/docs/torchmetrics/stable/
[10] github.com/csteinmetz1/auraloss
[11] github.com/gudgud96/frechet-audio-distance
[12] ant-novak.com/pages/sss/

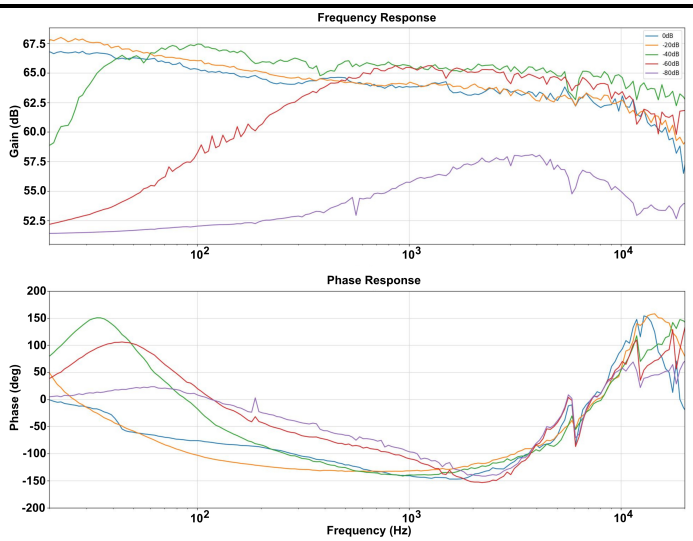


Fig. 3: Example of frequency and phase response measurement after training

2.1 Differentiable Black-box Models

This section provides an overview of state-of-the-art neural network architectures and conditioning methods included in NablAFx.

LSTM — The recurrent neural network architecture we implement is widely adopted for nonlinear effects (e.g., overdrive, distortion, guitar amps) [13, 27], nonlinear time-varying effects (e.g., fuzz, compressor) [7, 14], and modulation effects (e.g., phaser, flanger) [6, 28]. As shown in Fig. 4a, it consists of a single LSTM layer, a linear layer, and a *tanh* activation. For parametric models, a conditioning block processes control values and optionally the input sequence.

TCN — Temporal Convolution Networks (TCNs), introduced in [29] and shown to outperform recurrent architectures [30] on a variety of tasks, were proposed for audio effects modeling [14, 31, 32] and applied to linear (EQ, reverb) and nonlinear time-varying (compressor) effects. The architecture (Fig. 4b) consists of a series of residual blocks (Fig. 4c) made of 1-dimensional convolutions with increasing dilation factors, optionally followed by batch normalization and conditioning block, and an activation function (here *tanh*). A linear layer matches the output channels to the input size.

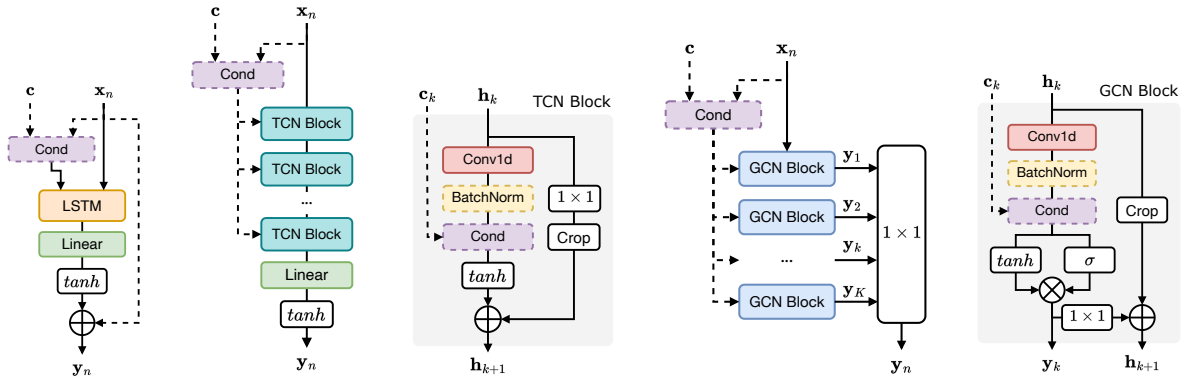
GCN — Gated Convolution Networks (GCNs), introduced in [33] as a feed-forward WaveNet, are a special case of TCNs with gated convolutions. GCNs have been used in [27, 34] for nonlinear audio effects (guitar amp, overdrive, distortion) and in [7] for nonlinear time-varying effects (compressor, fuzz). Beside the activation function at each block (Fig.4e), a GCN (Fig.4d) differs from a TCN in that its output is a linear combination of the activation features at each block.

S4 — Structured state space sequence models (S4) were introduced in [35] as a general sequence modeling architecture and shown to outperform recurrent, convolutional and Transformer architectures on a variety of tasks. An S4 layer is a differentiable implementation of an infinite impulse response (IIR) system in state-space form, with a theoretically infinite receptive field, similar to recurrent networks. Based on these observations state-space models were adopted for non-linear time-varying (compressor) effects modeling [36, 37].

The architecture in our framework, based on [36] (Fig. 5a), consists of S4 blocks. Unlike standard convolutional ones, S4 layers are not combined or mixed across data channels, this explains the use of a linear layer and activation function (tanh) at the input of each S4 block (Fig. 5b) for affine transformations along the channel dimension. These are followed by an S4D layer [38], which uses diagonal matrices for a parameter-efficient implementation, optional batch normalization and conditioning block, followed by an activation function (*tanh* in this case). Linear layers are used at the start and end to adjust the channel count to match the input data.

2.1.1 Conditioning for Black-box Models

Conditioning mechanisms for black-box models have been explored for different purposes: to include parametric control [8, 14, 36], to capture long-range dependencies [7] or for modulation in LFO-driven effects [6, 28]. While concatenation and feature-wise linear modulation (FiLM) [6, 8, 14, 36] remain the most common methods, temporal FiLM (TFiLM) has been adopted to capture time-varying behavior [7]. Beside these established methods, in this work we also propose three further conditioning methods: time-varying concatenation (TVCond), tiny TFiLM (TTFiLM) and time-varying FiLM (TVFiLM), as efficient implementations of time-varying conditioning similar to TFiLM. Concatenating control values ($\mathbf{c}$) to the input sequence ($\mathbf{x_n}$)



(a) LSTM **(b)** TCN **(c)** TCN block **(d)** GCN **(e)** GCN block

Fig. 4: Black-box architectures included in NablAFx

along the channel dimension is a simple, parameter-efficient conditioning method. It serves as a baseline for recurrent networks and has been used for parametric control in, e.g., compression [14] and overdrive [15].

Equally common is FiLM conditioning, mainly when using TCN [14, 15], GCN [15] or S4 [36, 37] backbones, with works adopting it for compressors [8, 14, 37] and overdrive [8, 15] modeling. Introduced in [39] as a general-purpose conditioning method, FiLM modulates a neural network's intermediate features using a conditioning vector $\mathbf{c}$. It learns functions f and g to generate scaling ($\gamma_{k,c} = f(\mathbf{c})$) and bias ($\beta_{k,c} = g(\mathbf{c})$) parameters for each layer k and channel c, which are used to modulate the activations at each layer $\mathbf{h}_{k,c}$, via a feature-wise affine transformation:

$$\text{FiLM}(\mathbf{h}_{k,c}, \gamma_{k,c}, \beta_{k,c}) = \gamma_{k,c} \cdot \mathbf{h}_{k,c} + \beta_{k,c}. \quad (1)$$

In practice, f and g are neural networks (Fig. 5c) that learn a latent representation $\mathbf{z}$ of the conditioning vector $\mathbf{c}$; then, a linear layer uses the latent representation to generate scaling and bias parameters for each block of the main network (Fig. 5d).

TFiLM [40] enhances network expressivity by using recurrent networks to modulate intermediate features over time as a function of layer activations $\mathbf{h}_k$ and optionally a conditioning vector $\mathbf{c}$ (Fig. 6a). Given a sequence of activations $\mathbf{h}_k$ from the k-th block of a network, the sequence is split into T blocks of B samples $\mathbf{h}_{k,b_1:b_T}$ along the sequence dimension. For each block $\mathbf{h}_{k,b_t}$, 1-dimensional max pooling downsamples the signal by a factor of B. To include the conditioning vector $\mathbf{c}$, it is repeated T times and concatenated with the downsampled activations. Then, an LSTM generates scaling $\gamma_{k,b_1:b_T,c}$ and bias $\beta_{k,b_1:b_T,c}$ parameters for each channel c, which are used to modulate the activations in each block via an affine transformation:

$$\text{TFiLM}(\mathbf{h}_{k,b_1:b_T,c}, \gamma_{k,b_1:b_T,c}, \beta_{k,b_1:b_T,c}) = \gamma_{k,b_1:b_T,c} \cdot \mathbf{h}_{k,b_1:b_T,c} + \beta_{k,b_1:b_T,c}.$$

In its standard formulation, TFiLM conditioning adds a recurrent network for each block in the main neural network, which can lead to a significant increase in parameters due to the number of blocks (typically 5-10) and channels (typically 16-32).

To retain TFiLM's expressivity while reducing parameters and computational cost, we propose two methods: TTFiLM and TVFiLM. TTFiLM (Fig.6b) is structurally similar to TFiLM, and reduces the computational complexity by using fewer channels in the recurrent network, achieved through a linear layer before it. The output is then scaled up to the required number of scaling $\gamma_{k,b_1:b_T,c}$ and bias $\beta_{k,b_1:b_T,c}$ channels using a small MLP. TVFiLM is a time-varying extension of FiLM conditioning. It replaces the MLP in the FiLM controller (Fig. 5c) with a recurrent network (Fig. 6c), creating a time-dependent latent representation $\mathbf{z}_{n,b1:b_T}$ shared across the main network's blocks. Modulation sequences are then generated at each block via a linear layer (Fig.6d), similarly to standard FiLM (Fig.5d).

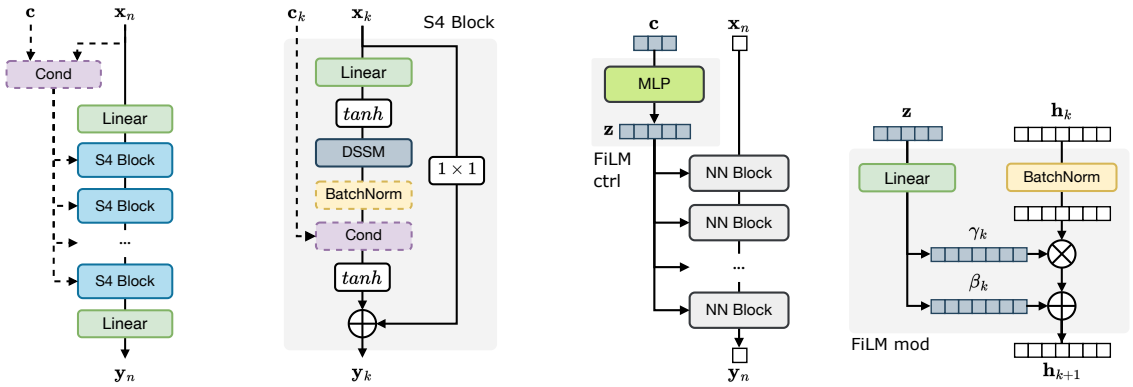
We also implement time-varying concatenation (TV-Cond) for recurrent models by using a TVFiLM controller to generate a time-dependent conditioning sequence, which is concatenated to the input for greater expressivity compared to standard concatenation.

2.2 Differentiable Gray-box Models

As described in Sec. 2 we define a gray-box model as a sequence of differentiable processors, each with an associated controller which generates the control parameters that dictate the behavior of the processor.

2.2.1 Differentiable Audio Processors

For our application, we define three types of audio processors: basic (e.g., phase inversion, gain), filters (e.g., EQ, shelving), and nonlinearities (e.g., tanh, MLP).



(a) S4 **(b)** S4 block **(c)** FiLM controller **(d)** FiLM modulator

Fig. 5: Black-box architectures and conditioning methods included in NablAFx

Most processors can be controlled by any of the controllers in Sec. 2.2.2, enabling parametric and time-varying configurations. All filters are implemented as differentiable biquads [41], with the transfer function:

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{a_0 + a_1 z^{-1} + a_2 z^{-2}}.$$

Biquad coefficients are calculated based on center/cutoff frequency (Hz), gain (dB), and Q factor, following Robert Bristow-Johnson's method[13]. To implement N^{th} order filters (e.g., Parametric EQ) we follow the common practice of using K cascaded second order sections:

$$H(z) = \prod_{k=0}^{K} H_k(z) \tag{2}$$

The frequency response is computed evaluating the transfer function along the unit circle in the complex plane and taking the magnitude:

$$\left|H(e^{j\omega})\right| = \left|\prod_{k=0}^{K} H_k(e^{j\omega})\right|.$$

For efficiency, during training we adopt the frequency sampling method, which approximates a cascade of second order IIR filters by computing the frequency response as in Eq. 2, applying the convolution in the frequency domain via multiplication and using the inverse FFT to transform the signal back to the time domain:

$$y[n] = F^{-1}[Y(e^{j\omega})] = F^{-1}[X(e^{j\omega})H(e^{j\omega})].$$

In the following paragraph we describe each processor. Generally, all processors' parameters can depend on the input $\mathbf{x}$ and/or controls $\mathbf{c}$.

Phase Inversion — Invert the phase of the input.

Gain — Multiply input by a gain value in dB.

DC Offset — Add a constant value to the input.

Lowpass/Highpass — Second order lowpass/highpass implemented as a single biquad section. Each filter is defined by 2 parameters: cutoff frequency and Q factor.

Low/High Shelf — Second order low/high shelving filter implemented as a single biquad section. Each filter is defined by 3 parameters: gain, cutoff frequency, Q factor.

Peak/Notch — Second order peak/notch filter implemented as a single biquad section. Each filter is defined by 3 parameters: gain, center frequency, Q factor.

Parametric EQ — We define a Parametric EQ as a chain of 5 filters: low shelf, three peak/notch filters and a high shelf. Each Parametric EQ has 15 parameters.

Shelving EQ — We define a Shelving EQ as a chain of 4 filters: highpass, low shelf, high shelf, lowpass. Each Shelving EQ is defined by a total of 10 parameters.

Static FIR Filter — We define a Static FIR Filter using a SIREN layer [42], which stores the tap values of an N^{th}-order impulse response. The network can be initialized with a pre-trained response (e.g., loudspeaker) and its hyperparameters (hidden dimension and layers count) to be customized.

Tanh Nonlinearity — Standard hyperbolic tangent.

Static MLP Nonlinearity — MLP Nonlinearity implemented with SIREN layer, initialized by default with a pre-trained model approximating a *tanh*.

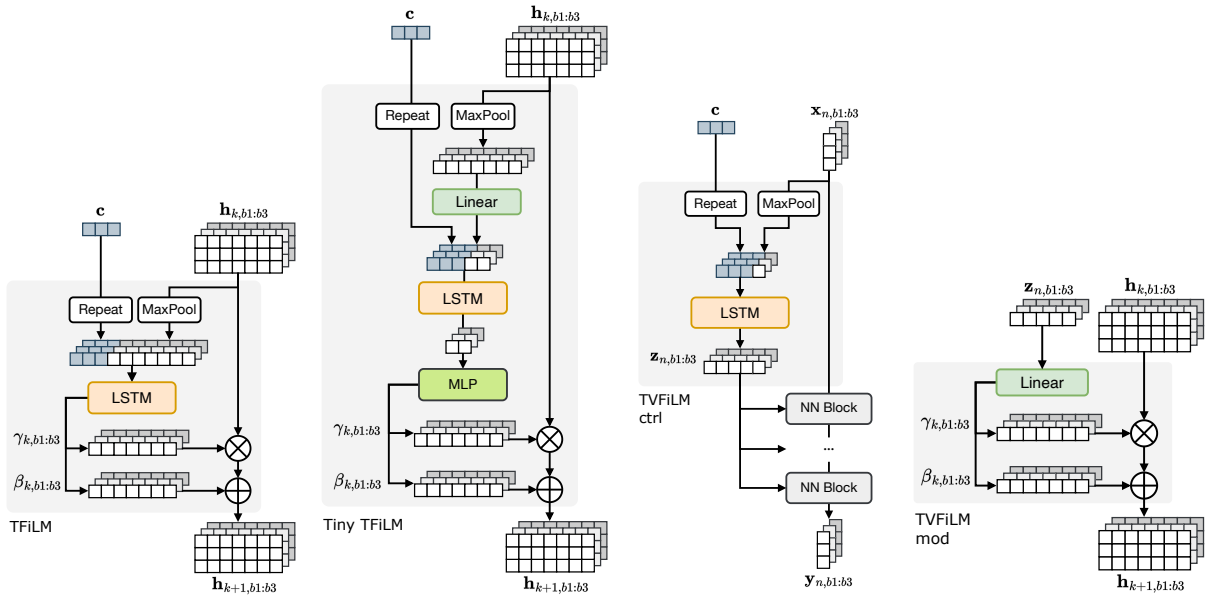
Static Rational Nonlinearity — A Padé approximant [43] is a rational function of order $[m/n]$ that best approximates a function $f(x)$ near a specific point, with $m \geq 0$ and $n \geq 1$:

$$R(x) = \frac{a_0 + a_1 x + a_2 x^2 + \cdots + a_m x^m}{1 + b_1 x + b_2 x^2 + \cdots + b_n x^n}$$

which agrees with $f(x)$ to the highest possible order. Learnable Padé approximants[14] [44] enable flexible ra-

[13] www.musicdsp.org/en/latest/Filters/197-rbj-audio-eq-cookbook.html

[14] github.com/ml-research/rational_activations



(a) TFiLM **(b)** TTFiLM **(c)** TVFiLM controller **(d)** TVFiLM modulator

Fig. 6: Black-box architectures and conditioning methods included in NablAFx

tional activation functions with few weights (numerator and denominator coefficients). We define a learnable Static Rational Nonlinearity using a single rational activation layer, initialized by default to a *tanh* approximation of order $[6,5]$.

2.2.2 Differentiable Controllers

We define five types of differentiable controllers (Fig. 7) to generate control parameters for an audio processor.
Dummy — A Dummy controller is a placeholder for processors that don't require control parameters (e.g., Phase Inversion, Static FIR Filter, Static Nonlinearity).
Static — A Static controller is a tensor of trainable controls $\mathbf{b}$ - one for each control parameter in the respective processor - followed by a sigmoid function to limit the output to the [0,1] range: $g = \sigma(\mathbf{b})$.
Static Conditional — A Static Conditional controller uses an MLP with a sigmoid activation to adjust control parameters based on audio effects controls (or some other fixed values): $g = f(\mathbf{c})$. Hyperparameters include number of input controls and output control parameters, number of layers, and hidden dimensions.
Dynamic — A Dynamic controller is used to adjust control parameters over time based on another signal, oftentimes the input audio: $g[n] = f(x[n])$. The control signal is downsampled (default downsampling factor is 128), processed through an LSTM, a sigmoid activation, and upsampled to output a control parameters sequence $\mathbf{g}_n$ at the original rate. Hyperparameters include block size (i.e., downsampling factor) and number of recurrent layers, while the hidden size is set by the number of control parameters for each processor.
Dynamic Conditional — A Dynamic Conditional controller adjusts control parameters based on both fixed values (typ., audio effect controls) and a time-varying control (typ., input signal): $g[n] = f(x[n], c)$. The signal is downsampled while the controls are upsampled and concatenated, the sequence processed by an LSTM, and after sigmoid activation and upsampling, the control parameters sequence $\mathbf{g}_n$ is returned at the original rate.

Although the control parameters sequences are at sampling rate, the block size (i.e., downsampling rate) is used internally in each processors to downsample the sequence so that the coefficients are recomputed once per block. This is not a limitation, as setting the block size to 1 provides sequences at audio rate. No interpolation methods have been implemented for smooth control sequences at the time of writing.

3 Audio Effects Modeling

To showcase our audio effects modeling framework and evaluate the proposed conditioning methods we train parametric black- and gray-box models of the Multidrive Pedal Pro F-Fuzz—a digital emulation of the Dallas Arbiter Fuzz Face—while for a complete study on differentiable black-box and gray-box modelling of audio effects that uses NablAFx, we point the reader to [45]. Table 2 shows all models configurations. We select TCN and S4 models and evaluate all conditioning methods available (i.e., FiLM, TFiLM, TTFiLM, TVFiLM). The table shows how TTFiLM and TVFiLM enable implementation of time-varying conditioning with a small overhead w.r.t. FiLM.

Table 3: Test loss for parametric models trained on Multidrive Pedal Pro F-Fuzz. Best model for each architecture shown in **bold**.

Model	Tot.	$L1$	MR-STFT
TCN-F-45-S-16	0.7095	0.0217	0.6878
TCN-TF-45-S-16	**0.4886**	0.0077	0.4809
TCN-TTF-45-S-16	0.5324	0.0102	0.5223
TCN-TVF-45-S-16	0.5356	0.0115	0.5241
S4-F-S-16	0.7687	0.0243	0.7444
S4-TF-S-16	0.4034	0.0075	0.3959
S4-TTF-S-16	0.3816	0.0066	0.3749
S4-TVF-S-16	**0.3354**	0.0058	0.3296
GB-C-DIST-MLP	1.2104	0.0611	1.1492
GB-C-DIST-RNL	1.2531	0.0672	1.1858
GB-C-FUZZ-MLP	**0.9303**	0.0345	0.8958
GB-C-FUZZ-RNL	0.9395	0.0355	0.9040

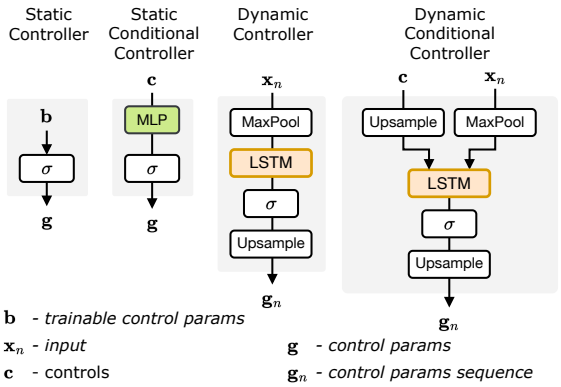


Fig. 7: Controllers included in NablAFx

We propose two gray-box architectures (GB-DIST and GB-FUZZ) that are extensions of the typical Weiner-Hammerstein model [16] adopted for distortion modeling, which includes a memoryless nonlinearity in between pre-emphasis and de-emphasis linear time-invariant filters. We test two nonlinearities: Static MLP (MLP) and Static Rational Nonlinearity (RNL). While GB-DIST models only use Static Conditional controllers, in GB-FUZZ we opt for a Dynamic Conditional controller for the Offset block, to capture the characteristic dynamic bias shift of fuzz effects.

Models are trained for a maximum of 15k steps using a weighted sum of $L1$ and MR-STFT [24] losses and the results shown in Table 3. For TCN models, TTF and TVF conditioning perform on par with TF; while for S4 models TTF and TVF outperform TF. For GB models, regardless of the nonlinearity type, GB-FUZZ achieves better results than GB-DIST, proving the Dynamic controller useful. Also, RNL in shown to be an effective and efficient alternative to the MLP nonlinearity.

4 Conclusion

In this work we presented NablAFx, an open-source framework developed to support research in differentiable black-box and gray-box audio effects modeling. Its modular design enables easy configuration of experiments with different architectures, datasets, training settings, and loss functions. With logging, plotting, and performance metrics, it simplifies experiment analysis and comparison. We consider gray-box models as a series connection of DDSP blocks, but this could be generalized using a graph representation. While black-box models are currently single networks, they could be extended to interconnected networks. Hybrid models could be introduced to combine black- and gray-box processors, allowing DDSP blocks with known functions alongside neural networks for learning functions. Moreover, community contributions could help expand our framework in various ways, including new architectures, loss functions, metrics, and more.

5 Acknowledgments

Funded by UKRI and EPSRC - grant EP/S022694/1.

References

[1] Wilmering, T., Moffat, D., Milo, A., and Sandler, M. B., "A history of audio effects," *Applied Sciences*, 10(3), 2020.

[2] Engel, J., Hantrakul, L., Gu, C., and Roberts, A., "DDSP: Differentiable digital signal processing," *arXiv preprint arXiv:2001.04643*, 2020.

[3] Comunita, M. and Reiss, J. D., "AFxResearch: a repository and website of audio effects research," in *DMRN+ 19: Digital Music Research Network One-day Workshop 2024*, 2024.

[4] Comunità, M., Stowell, D., and Reiss, J. D., "Guitar Effects Recognition and Parameter Estimation With Convolutional Neural Networks," *Journal of the Audio Engineering Society*, 69(7/8), 2021.

[5] Colonel, J. T., Steinmetz, C. J., Michelen, M., and Reiss, J. D., "Direct design of biquad filter cascades with deep learning by sampling random polynomials," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022.

[6] Mitcheltree, C., Steinmetz, C. J., Comunità, M., and Reiss, J. D., "Modulation extraction for LFO-driven audio effects," *arXiv preprint arXiv:2305.13262*, 2023.

[7] Comunità, M., Steinmetz, C. J., Phan, H., and Reiss, J. D., "Modelling black-box audio effects with time-varying feature modulation," in *2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023.

Table 2: Parametric models included in the experiments. Cond. = conditioning method, R.F. = receptive field in samples. PEQ = Parametric EQ, G = Gain, O = Offset, MLP = Multilayer Perceptron, RNL = Rational Non Linearity. Controllers: .s = static, .d = dynamic, .sc = static conditional, .dc = dynamic conditional

Model	Cond.	R.F.	Blocks	Kernel	Dilation	Channels	# Params	FLOP/s	MAC/s
TCN-F-45-S-16	FiLM	2047	5	7	4	16	15.0k	736.5M	364.3M
TCN-TF-45-S-16	TFiLM	2047	5	7	4	16	42.0k	762.8M	364.2M
TCN-TTF-45-S-16	TTFiLM	2047	5	7	4	16	17.3k	744.0M	367.4M
TCN-TVF-45-S-16	TVFiLM	2047	5	7	4	16	17.7k	740.4M	366.2M
Model	Cond.	R.F.	Blocks	State Dimension		Channels	# Params	FLOP/s	MAC/s
S4-F-S-16	FiLM	-	4	4		16	8.9k	135.2M	53.8M
S4-TF-S-16	TFiLM	-	4	4		16	30.0k	155.6M	53.8M
S4-TTF-S-16	TTFiLM	-	4	4		16	10.2k	141.0M	56.3M
S4-TVF-S-16	TVFiLM	-	4	4		16	11.6k	138.9M	55.3M
Model	Signal Chain						# Params	FLOP/s	MAC/s
GB-C-DIST-MLP	PEQ.sc → G.sc → O.sc → MLP → G.sc → PEQ.sc						4.5k	202.8M	101.4M
GB-C-DIST-RNL	PEQ.sc → G.sc → O.sc → RNL → G.sc → PEQ.sc						2.3k	920.5k	4.3k
GB-C-FUZZ-MLP	PEQ.sc → G.sc → O.dc → MLP → G.sc → PEQ.sc						4.2k	202.8M	101.4M
GB-C-FUZZ-RNL	PEQ.sc → G.sc → O.dc → RNL → G.sc → PEQ.sc						2.0k	988.9k	3.6k

[8] Simionato, R. and Fasciani, S., "Comparative Study of Recurrent Neural Networks for Virtual Analog Audio Effects Modeling," *arXiv preprint arXiv:2405.04124*, 2024.

[9] Steinmetz, C. J., Bryan, N. J., and Reiss, J. D., "Style Transfer of Audio Effects with Differentiable Signal Processing," *Journal of the Audio Engineering Society*, 70(9), 2022.

[10] Steinmetz, C. J., Singh, S., Comunità, M., Ibnyahya, I., Yuan, S., Benetos, E., and Reiss, J. D., "St-ito: Controlling audio effects for style transfer with inference-time optimization," *arXiv preprint arXiv:2410.21233*, 2024.

[11] Steinmetz, C. J., Pons, J., Pascual, S., and Serra, J., "Automatic multitrack mixing with a differentiable mixing console of neural audio effects," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021.

[12] Sai Vanka, S., Safi, M., Rolland, J.-B., and Fazekas, G., "Adoption of AI Technology in Music Mixing Workflow: An Investigation," in *Audio Engineering Society Convention 154*, 2023.

[13] Wright, A., Damskägg, E.-P., and Välimäki, V., "Real-time black-box modelling with recurrent neural networks," in *Proc. of the Int. Conf. on Digital Audio Effects (DAFx-19)*, 2019.

[14] Steinmetz, C. J. and Reiss, J. D., "Efficient neural networks for real-time modeling of analog dynamic range compression," in *Audio Engineering Society Convention 152*, 2022.

[15] Yeh, Y.-T., Hsiao, W.-Y., and Yang, Y.-H., "Hyper recurrent neural network: Condition mechanisms for black-box audio effect modeling," *arXiv preprint arXiv:2408.04829*, 2024.

[16] Colonel, J. T., Comunità, M., and Reiss, J., "Reverse engineering memoryless distortion effects with differentiable waveshapers," in *Audio Engineering Society Convention 153*, 2022.

[17] Wright, A. and Valimaki, V., "Grey-box modelling of dynamic range compression," in *Proc. of the Int. Conf. on Digital Audio Effects (DAFx-20in22)*, 2022.

[18] Carson, A., King, S., Botinhao, C. V., and Bilbao, S., "Differentiable grey-box modelling of phaser effects using frame-based spectral processing," in *Proceedings of the 26th International Conference on Digital Audio Effects*, 2023.

[19] Miklanek, S., Wright, A., Välimäki, V., and Schimmel, J., "Neural grey-box guitar amplifier modelling with limited data," in *Proc. of the Int. Conf. on Digital Audio Effects (DAFx-23)*, 2023.

[20] Yeh, Y.-T., Chen, Y.-H., Cheng, Y.-C., Wu, J.-T., Fu, J.-J., Yeh, Y.-F., and Yang, Y.-H., "DDSP Guitar Amp: Interpretable Guitar Amplifier Modeling," *arXiv preprint arXiv:2408.11405*, 2024.

[21] Uzrad, N., Barkan, O., Elharar, A., Shvartzman, S., Laufer, M., Wolf, L., and Koenigstein, N., "DiffMoog: a Differentiable Modular Synthesizer for Sound Matching," *arXiv preprint arXiv:2401.12570*, 2024.

[22] Lee, S., Martínez-Ramírez, M., Liao, W.-H., Uhlich, S., Fabbro, G., Lee, K., and Mitsufuji,

Y., "GRAFX: an open-source library for audio processing graphs in PyTorch," *arXiv preprint arXiv:2408.03204*, 2024.

[23] Yeh, Y.-T., Hsiao, W.-Y., and Yang, Y.-H., "PyNeuralFx: A Python Package for Neural Audio Effect Modeling," *arXiv preprint arXiv:2408.06053*, 2024.

[24] Steinmetz, C. J. and Reiss, J. D., "auraloss: Audio focused loss functions in PyTorch," in *Digital music research network one-day workshop (DMRN+ 15)*, 2020.

[25] Kilgour, K., Zuluaga, M., Roblek, D., and Sharifi, M., "Fr\'echet audio distance: A metric for evaluating music enhancement algorithms," *arXiv preprint arXiv:1812.08466*, 2018.

[26] Novak, A., Simon, L., Kadlec, F., and Lotton, P., "Nonlinear system identification using exponential swept-sine signal," *IEEE Transactions on Instrumentation and Measurement*, 59(8), 2009.

[27] Wright, A., Damskägg, E.-P., Juvela, L., and Välimäki, V., "Real-time guitar amplifier emulation with deep learning," *Applied Sciences*, 10(3), 2020.

[28] Wright, A. and Valimaki, V., "Neural modeling of phaser and flanging effects," *Journal of the Audio Engineering Society*, 69(7/8), 2021.

[29] Lea, C., Vidal, R., Reiter, A., and Hager, G. D., "Temporal convolutional networks: A unified approach to action segmentation," in *Computer Vision–ECCV Workshops*, Springer, 2016.

[30] Bai, S., Kolter, J. Z., and Koltun, V., "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv preprint arXiv:1803.01271*, 2018.

[31] Steinmetz, C. J., "Learning to mix with neural audio effects in the waveform domain," *MS thesis*, 2020.

[32] Stein, M., Abeßer, J., Dittmar, C., and Schuller, G., "Automatic detection of audio effects in guitar and bass recordings," in *Audio Engineering Society Convention 128*, 2010.

[33] Rethage, D., Pons, J., and Serra, X., "A wavenet for speech denoising," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.

[34] Damskägg, E.-P., Juvela, L., and Välimäki, V., "Real-time modeling of audio distortion circuits with deep learning," in *Sound and music computing conference*, 2019.

[35] Gu, A., Goel, K., and Ré, C., "Efficiently modeling long sequences with structured state spaces," *arXiv preprint arXiv:2111.00396*, 2021.

[36] Yin, H., Cheng, G., Steinmetz, C. J., Yuan, R., Stern, R. M., and Dannenberg, R. B., "Modeling Analog Dynamic Range Compressors using Deep Learning and State-space Models," *arXiv preprint arXiv:2403.16331*, 2024.

[37] Simionato, R. and Fasciani, S., "Modeling Time-Variant Responses of Optical Compressors with Selective State Space Models," 2024.

[38] Gupta, A., Gu, A., and Berant, J., "Diagonal state spaces are as effective as structured state spaces," *Advances in Neural Information Processing Systems*, 35, 2022.

[39] Perez, E., Strub, F., De Vries, H., Dumoulin, V., and Courville, A., "Film: Visual reasoning with a general conditioning layer," in *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[40] Birnbaum, S., Kuleshov, V., Enam, Z., Koh, P. W. W., and Ermon, S., "Temporal FiLM: Capturing long-range sequence dependencies with feature-wise modulations." *Advances in Neural Information Processing Systems*, 32, 2019.

[41] Nercessian, S., "Neural parametric equalizer matching using differentiable biquads," 2020.

[42] Sitzmann, V., Martel, J., Bergman, A., Lindell, D., and Wetzstein, G., "Implicit neural representations with periodic activation functions," *Advances in neural information processing systems*, 33, 2020.

[43] Baker Jr, G. A. and Gammel, J. L., "The Padé approximant," *Journal of Mathematical Analysis and Applications*, 2(1), 1961.

[44] Molina, A., Schramowski, P., and Kersting, K., "Padé activation units: End-to-end learning of flexible activation functions in deep networks," *arXiv preprint arXiv:1907.06732*, 2019.

[45] Comunità, M., Steinmetz, C. J., and Reiss, J. D., "Differentiable black-box and gray-box modeling of nonlinear audio effects," *arXiv preprint arXiv:2502.14405*, 2025.

Late Breaking Demo Papers

Audio Engineering Society

Late Breaking Demo Paper

Presented at the AES International Conference on
Artificial Intelligence and Machine Learning for Audio
2025 September 8-10, London, UK

This Late Breaking Demo Paper was selected after a minimal screening process and was not peer reviewed. This Paper has been reproduced from the author's advance manuscript without editing, corrections, or consideration by the Review Board. The AES takes no responsibility for the contents. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Audio Engineering Society.

OBSIDIAN-Neural: Real-Time AI Music Generation Plugin for Live Performance

Anthony Charretier
Independent Developer, France
anthony.charretier@etik.com

ABSTRACT

OBSIDIAN-Neural introduces an approach to human-AI musical collaboration through a Virtual Studio Technology version 3 plugin implementing real-time neural audio generation directly within Digital Audio Workstations (DAW). Unlike external neural audio generation workflows that break creative momentum, OBSIDIAN-Neural enables seamless live performance where artificial intelligence (AI) becomes a collaborative partner rather than a replacement for human creativity

1 Introduction

Contemporary neural audio generation models commonly require complex external workflows that interrupt the creative process. Musicians must stop their session, generate content in separate applications, and manually import results back into their DAW . This paper presents OBSIDIAN-Neural, a plugin that eliminates these interruptions by implementing real-time neural audio generation directly within the DAW environment.

While previous plugins like VroomAI from the 2023 Neural Audio Plugin Competition [4] demonstrated text-to-audio generation, these solutions typically provide single-sample generation without performance-oriented features.

2 Technical architecture

The system implements a hybrid architecture combining local Large Language Models with neural audio generation models, creating an unprecedented real-time workflow for music production. Unlike existing AI sample generators such as VroomAI [4], which provides basic text-to-audio generation with manual export workflows, OBSIDIAN-Neural integrates generation directly within a comprehensive performance instrument featuring multi-track sampling, Musical Instrument Digital Interface triggering, and real-time controls.

Figure 1: OBSIDIAN-Neural interface showing multi-track sampler, sequencer, and real-time generation controls

The core technical components include:

Local Large Language Model Integration: A Gemma-3-4B-IT model (GGUF format via Unsloth) [6] running on CPU processes simple user keywords and contextual information including DAW tempo, key signature, and generation history to generate optimized prompts for audio synthesis.

Neural Audio Generation: Stable Audio Open [5] models create 4-10 second audio samples with automatic tempo synchronization to maintain coherence with existing DAW projects, eliminating manual drag-and-drop workflows.

Multi-track Architecture: An 8-track sampler with individual outputs, built-in 16-step sequencer, and comprehensive Musical Instrument Digital Interface Learn functionality enables hardware controller integration for live performance scenarios.

Real-time Synchronization: Automatic alignment with DAW transport, tempo, and sample rate ensures immediate integration. Waveform display with loop point editing and pitch controls enable precise sample manipulation and one-shot isolation without external file management.

3 Workflow Innovation

OBSIDIAN-Neural eliminates disruptive external workflows through seamless DAW integration. Musicians assign generation triggers to hardware controllers via Musical Instrument Digital Interface Learn, enabling background generation during live performance without interrupting playback.

4 Performance Analysis

Live performance testing demonstrates generation times of 30-40 seconds to 2-3 minutes on consumer hardware configurations. The system maintains musical coherence through conversational memory, allowing the Large Language Model to generate contextually appropriate samples that complement previously generated elements. The system addresses critical challenges in real-time AI music generation including latency management through efficient model loading strategies, hardware accessibility through optimized local inference, and cross-platform compatibility across multiple DAW environments.

5 Community adoption

Since June 2025, OBSIDIAN-Neural has gained international recognition through open-source development [1][2][3]. User feedback reveals enthusiastic adoption by producers seeking compositional building blocks, with installation complexity remaining the primary barrier.

6 Conclusions

OBSIDIAN-Neural demonstrates that AI music tools can enhance rather than replace human creativity, establishing AI as a new instrument category enabling extended improvisation sessions where skilled prompting allows controlled unpredictability in live performance. This opens possibilities for DJs, live electronic performers, and musicians seeking enhanced improvisational capabilities. The live demonstration will showcase real-time generation triggered by text prompts, integration with hardware synthesizers, and multi-track layering with Musical Instrument Digital Interface controller interaction.

References

[1] Charretier, A. "OBSIDIAN Neural: When AI Becomes Your Jam Partner," Medium, June 2025. Available: https://medium.com/@innermost47/obsidian-neural-when-ai-becomes-your-jam-partner-5203726a3840 [Accessed: July 2025].

[2] Bedroom Producers Blog, "OBSIDIAN Neural Sound Engine," June 2025. Available: https://bedroomproducersblog.com/2025/06/06/obsidian-neural-sound-engine/ [Accessed: July 2025].

[3] Charretier, A. "OBSIDIAN-Neural Open Source Repository," GitHub, 2025. Available: https://github.com/innermost47/ai-dj [Accessed: July 2025].

[4] Anderson, M., Hill, B. "VroomAI: Generate sounds from words directly in your DAW," 2023 Neural Audio Plugin Competition, 2023. Available: https://github.com/vroomai/vst [Accessed: July 2025].

[5] Stability AI, "Stable Audio Open," Hugging Face, 2024. Available: https://huggingface.co/stabilityai/stable-audio-open-1.0 [Accessed: July 2025].

[6] Google, "Gemma-3-4B-IT," Hugging Face, 2024. Available: https://huggingface.co/google/gemma-3-4b-it [Accessed: July 2025].

Audio Engineering Society

Late Breaking Demo Paper

Presented at the AES International Conference on
Artificial Intelligence and Machine Learning for Audio
2025 September 8–10, London, UK

This Late Breaking Demo Paper was selected after a minimal screening process and was not peer reviewed. This Paper has been reproduced from the author's advance manuscript without editing, corrections, or consideration by the Review Board. The AES takes no responsibility for the contents. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Audio Engineering Society.

Bias Research on Generated Music from around the World

Ahmet Solak[1], Florian Grötschla[1], Luca A. Lanzendörfer[1], and Roger Wattenhofer[1]

[1]*ETH Zurich, Switzerland*

Correspondence should be addressed to Florian Grötschla (fgroetschla@ethz.ch)

ABSTRACT

While recent years have seen remarkable progress in music generation models, research on their biases across countries, languages, cultures, and musical genres remains underexplored. In this work, we present preliminary research on a large-scale dataset containing real and generated music from around the world. We find disparities in music quality and distribution alignment compared to reference music between high- and low-resource regions, as well as between various genres across cultures and geographical regions.

1 Introduction

Music generation has seen remarkable progress in recent years [1]. Current commercial models have demonstrated exceptional performance, even outperforming real music in large-scale human evaluation studies [2]. This rapid progress highlights the need for approaches that allow the evaluation of potential biases and weaknesses in these models, especially those that go beyond broad measures of musical preference.

While music holds a unique and universal role across cultures worldwide, there is still a significant lack of intercultural and multilingual datasets in music generation research. This lack of diversity in cultural, linguistic, and genre representation means that many musical traditions, languages, and styles remain underrepresented or entirely absent from the research landscape. These gaps limit the ability of current models to generalize beyond the narrow distributions they are typically trained on.

2 Methodology

To evaluate the distribution misalignment, we generate 40k tracks with musical style information and synthetic lyrics for 10k artists from 79 countries across 5 continents with state-of-the-art commercial music generation models (Suno v4 and Mureka v6). We consider between 5 and 10 reference tracks for each artist used to generate music. To analyze the distributions of generated and reference audio we use the Frechet Audio Distance (FAD) [3], which has been widely used to estimate the perceptual quality of music [2]. To compute the distribution similarity with FAD we use various embedding models. We embed our generated and reference tracks with the audio embedding models PANN [4], LAION-CLAP [5] and MUQ-MULAN [6].

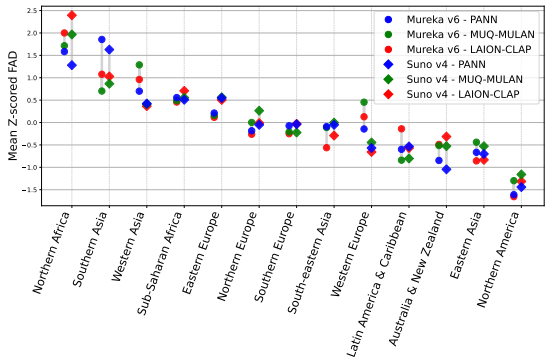


Fig. 1: Mean Z-score FAD for Suno and Mureka.

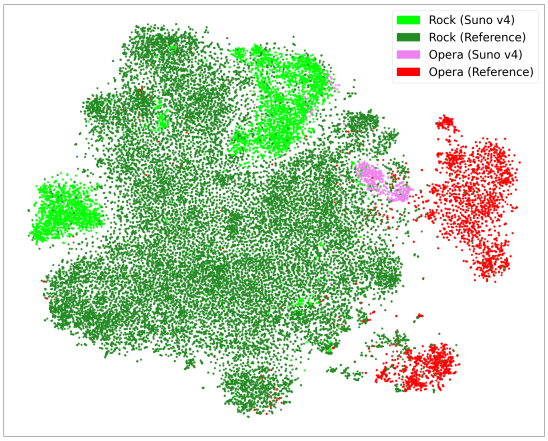


Fig. 2: t-SNE Plot for PANN embeddings of generated (Suno v4) and reference Rock and Opera tracks.

3 Results

For each embedding model, FAD scores are computed for all countries and then standardized (Z-scored) across countries within that model. These FAD scores are subsequently averaged within each world subregion. We report the resulting mean FAD scores in Fig. 1. The results show marked differences between regions, where all considered generative models are able to approximate music from high-resource regions like Nothern America and Oceania, while low-resource regions such as Southern Asia and Northern Africa prove more challenging for the commercial models considered. Furthermore, genres have a significant effect on the ability of state-of-the-art music generation models to approximate the styles of real music. In Fig. 2 we present a t-SNE clustering of reference and generated tracks, qualitatively comparing the genres Rock and Opera, which are both prevalent in higher resource regions such as Western Europe. This visualization suggests that music generation models are able to produce Rock music that is in-distribution with real Rock tracks, whereas generated Opera music appears to lie outside the distribution of real Opera tracks. To support this qualitative assessment with objective metrics, we observe that generated and reference Rock tracks have a considerably lower FAD score of 18.6 as compared to 73.4 for Opera, and even more interestingly, generated Opera tracks are more similar to reference Rock tracks with an FAD score of 35.1. These preliminary findings indicate that commercial models struggle to capture the diversity found in certain genres and regions, which warrants further investigation.

References

[1] Evans, Z. et al., "Stable audio open," in *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5, IEEE, 2025.

[2] Grötschla, F. et al., "Benchmarking Music Generation Models and Metrics via Human Preference Studies," in *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5, IEEE, 2025.

[3] Roblek, D. et al., "Frechet Audio Distance: A Reference-free Metric for Evaluating Music Enhancement Algorithms," in *Proc. Interspeech*, pp. 2350–2354, 2019.

[4] Kong, Q. et al., "PANNs: Large-scale pretrained audio neural networks for audio pattern recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28, pp. 2880–2894, 2020.

[5] Wu, Y. et al., "Large-scale Contrastive Language-Audio Pretraining with Feature Fusion and Keyword-to-Caption Augmentation," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, 2023.

[6] Zhu, H. et al., "MuQ: Self-Supervised Music Representation Learning with Mel Residual Vector Quantization," *CoRR*, 2025.

Audio Engineering Society

Late Breaking Demo Paper

Presented at the AES International Conference on
Artificial Intelligence and Machine Learning for Audio
2025 September 8–10, London, UK

This Late Breaking Demo Paper was selected after a minimal screening process and was not peer reviewed. This Paper has been reproduced from the author's advance manuscript without editing, corrections, or consideration by the Review Board. The AES takes no responsibility for the contents. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Audio Engineering Society.

Real-time low latency audio features with Resonate

Alexandre R.J. François[1]

[1]*Interactions Intelligence*

Correspondence should be addressed to Alexandre R.J. François (alex@interactionsintelligence.org)

ABSTRACT

This demonstration aims to bring to the attention of the audio ML community the recently introduced *Resonate*, a low latency, low memory footprint, and low computational cost algorithm to evaluate perceptually relevant spectral information from audio signals. *Resonate* offers potential cost and performance benefits wherever spectrograms are used or relevant in the processing of audio signal, including for training ML models and real-time online inference. The demonstration features live real-time computation of linear, logarithmic and mel frequency scale spectrograms, chromagrams and Mel Frequency Cepstral Coefficients (MFCCs).

1 Resonate Overview

Resonate [1] builds on a resonator model that accumulates the signal contribution around its resonant frequency in the time domain using the Exponentially Weighted Moving Average (EWMA), also known as a low-pass filter in signal processing. Consistently with on-line perceptual signal analysis, the EWMA gives more weight to recent input values, whereas the contributions of older values decay exponentially. A compact, iterative formulation of the model affords computing an update at each signal input sample, requiring no buffering and involving only a handful of arithmetic operations.

Banks of resonators, independently tuned to perceptually relevant frequency scales, compute an instantaneous, perceptually relevant estimate of the spectral content of an input signal in real-time. Both memory and per-sample computational complexity of such a bank are linear in the number of resonators, and independent of the number of input samples processed, or duration of processed signal.

As computations on a given input sample depend on the computation of the previous sample, batch processing cannot be vectorized across time, a common computational optimization for training in ML systems. However, all *per sample* computations can be parallelized across resonators and leverage hardware support such as SIMD architectures. In an offline processing context, the cumulative computational cost for a given duration increases linearly with the number of input samples processed.

Compared to traditional algorithms (FFT-based), these properties could result in significant cost reductions in

memory and computation when training ML systems on large datasets, in addition to making possible online, real-time spectral information computation, a necessary enabling step towards real-time inference.

Audio data used in ML systems is often down-sampled to reduce memory and computation/time requirements. *Resonate* offers lower memory and computation requirements than FFT based techniques, potentially freeing resources for higher resolution input. Where higher temporal or frequency resolution is not desirable, explicit control over the dynamics of the computation and complete freedom over resonator frequency distribution and tuning could lead to higher quality learned features and analysis results.

2 Resonate spectrograms

The resonators in a bank are independent so that there is no constraint on the tuning of their resonant frequencies or time constants. Where traditional methods first compute an STFT (linear frequency scale) and then map the result to a perceptually relevant frequency scale, such as a mel-frequency scale [2] or a log-frequency scale via a Constant-Q Transform (CQT) [3], a *Resonate* oscillator bank with adequately tuned resonators computes an arbitrary frequency scale spectrogram directly and efficiently, with more relevant frequency resolution and much higher temporal resolution (a value is available for each input sample).

3 Chroma and chromagrams

Humans perceive audio frequencies on a logarithmic scale [4]. For this reason, music and speech applications often favor mapping the frequency axis to correspond to the geometrically spaced frequencies of the equal-tempered scale (log-frequency spectrogram). Chroma features project the entire power spectrum to a single musical octave divided in 12 semitones (or chroma), and thus capture a pitch class profile of the signal. A chromagram is a plot of chroma information as a function of time, analogous to a spectrogram.

Chroma are important audio features in many Music Information Retrieval systems, for such tasks as audio and score alignment, and artist identification. For real-time score alignment tasks, the high temporal resolution of *Resonate* chroma may enable more precise results, with lower latency.

4 Mel Frequency Cepstral Coefficients

MFCCs are audio signal features that have been successfully used in speech recognition and speaker identification. In music, they characterize timbre, and have been used in Music Information Retrieval tasks such as genre classification and similarity estimation [5]. MFCCs are the amplitudes of the spectrum of the mel frequency spectrogram of the audio signal, typically obtained by applying a Direct Cosine Transform (DCT).

5 Resources

Available resources aim to promote and facilitate the adoption of Resonate in the community. The open source python module `noFFT` provides python and C++ implementations of *Resonate* functions and Jupyter notebooks illustrating their use in offline settings. The `Oscillators` Swift package contains reference implementations in Swift and C++. The Oscillators app demonstrates real-time spectrograms and derived audio features. Links are available on the project website: www.alexandrefrancois.org/Resonate

References

[1] François, A. R., "Resonate: Efficient Low Latency Spectral Analysis of Audio Signals," in *Proceedings of the International Conference on Computer Music*, pp. 251–258, Boston, MA, USA, 2025.

[2] Stevens, S. and Volkmann, J., "The Relation of Pitch to Frequency; A Revised Scale," *The American Journal of Psychology*, 53, pp. 329–353, 1940, doi:10.2307/1417526.

[3] Brown, J. C., "Calculation of a Constant Q Spectral Transform," *Journal of the Acoustical Society of America*, 89, pp. 425–434, 1991, doi:10.1121/1.400476.

[4] Rasch, R. and Plomp, R., "1 - The Perception of Musical Tones," in D. Deutsch, editor, *Psychology of Music*, Cognition and Perception, pp. 1–24, Academic Press, New York, 1982, ISBN 978-0-12-213562-0.

[5] Müller, M., *Information Retrieval for Music and Motion*, Springer Berlin Heidelberg, 2007, ISBN 9783540740476.

Audio Engineering Society

Late Breaking Demo Paper

Presented at the AES International Conference on
Artificial Intelligence and Machine Learning for Audio
2025 September 8–10, London, UK

This Late Breaking Demo Paper was selected after a minimal screening process and was not peer reviewed. This Paper has been reproduced from the author's advance manuscript without editing, corrections, or consideration by the Review Board. The AES takes no responsibility for the contents. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Audio Engineering Society.

Real-Time Audio Monitoring Pipeline with Edge Inference and IoT Supervision via ThingsBoard

Javier Naranjo-Alcazar[1], Jordi Grau-Haro[1], Ruben Ribes-Serrano[1], and Pedro Zuccarello[1]

[1]*Instituto Tecnologico de Informatica, Paterna, Spain*

Correspondence should be addressed to Javier Naranjo-Alcazar (jnaranjo@iti.es)

ABSTRACT

This paper presents a complete real-time audio monitoring pipeline that combines edge inference and remote supervision through the ThingsBoard platform. The system uses a Raspberry Pi 4B running models from the PANNs (Pretrained Audio Neural Networks) family to perform continuous audio classification at the edge. Inference results and system health data are transmitted to ThingsBoard for centralized visualization, analysis, and device management. This cost-effective, scalable solution enables intelligent audio sensing in Internet of Things (IoT) environments, supporting applications such as environmental monitoring, smart cities, and security. A demonstration is deployed at the Port of Valencia (Spain), where it helps identify acoustic events related to logistics and environmental noise pollution.

1 Introduction

This work presents a real-time audio monitoring pipeline that integrates edge inference with remote supervision through the ThingsBoard platform. The system uses a Raspberry Pi 4B running PANNs-based models (Pretrained Audio Neural Networks) [1] to perform continuous sound classification, while also collecting device health metrics such as temperature and CPU load. All data is transmitted to ThingsBoard for centralized visualization and management, enabling a low-cost, scalable, and autonomous solution for intelligent audio sensing in industrial and IoT settings.

2 Methods

The acoustic inference node (see Fig. 1) is enclosed in a rugged PVC IP65-rated case, designed to withstand harsh outdoor and industrial conditions. The core of the system is a Raspberry Pi 4B single-board computer, which performs on-device audio processing and manages data transmission. The node can be powered through a 12V DC input, a solar panel, or an internal battery, all managed by an uninterruptible power supply (UPS) module that ensures continuous operation during power fluctuations or outages.

Inference is performed using models from the PANNs family. For deployment on the Raspberry Pi, each

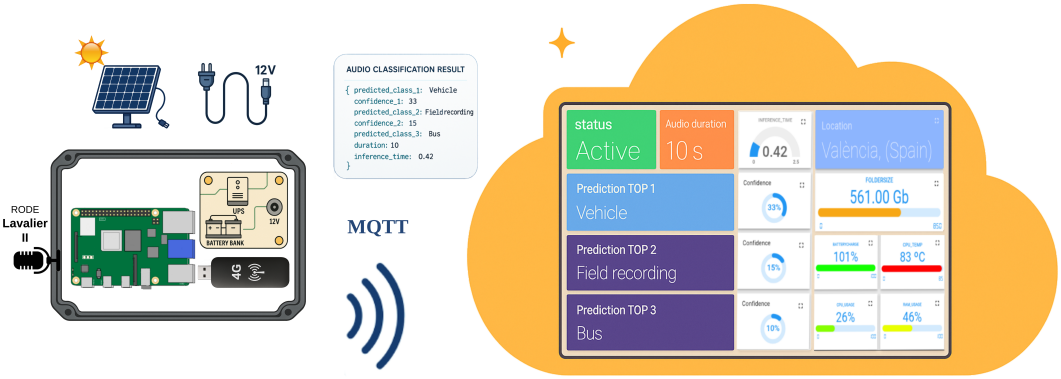


Fig. 1: Full system diagram of the edge audio inference node with 4G connectivity and remote monitoring via ThingsBoard.

model is converted to ONNX format with a fixed input size corresponding to 10 seconds of audio at a 32 kHz sampling rate (i.e., 320,000 samples). This duration was chosen based on the typical length of acoustic events observed in industrial environments, such as the Port of Valencia. Audio is initially recorded at 44.1 kHz and resampled to 32 kHz before inference. A 50% overlap is applied during the inference process, resulting in predictions every 5 seconds. All PANNs models were evaluated; the most efficient was MobileNetV2, achieving inference in 0.23 seconds, while the most computationally intensive was Res1dNet51, with an inference time of 2.6 seconds. The CNN14 model, identified as the best-performing architecture in the original work, has an inference time of 1.56 seconds.

The communication architecture of the proposed inference node leverages Message Queuing Telemetry Transport (MQTT) over a 4G mobile network to enable reliable data transmission in the absence of fixed connectivity. Each device integrates a USB router equipped with a SIM card, providing mobile broadband access and ensuring autonomous operation in distributed environments. After each inference cycle, the node publishes both classification outputs (Top 3 predictions with their probabilities) and telemetry data—including CPU temperature, resource usage, inference duration, audio duration, storage availability, and location—to a ThingsBoard instance (see Fig. 1). These updates are visualized through a custom dashboard that presents real-time and historical views of the node's activity and health metrics. This setup ensures lightweight, low-latency communication and facilitates large-scale monitoring of geographically dispersed nodes without reliance on local infrastructure, reinforcing the system's suitability for remote and industrial use cases.

3 Results

This paper presents the application of a real-time audio inference node in an industrial setting, leveraging PANNs models for continuous sound classification. It describes the designed Raspberry Pi-based node, the inference pipeline, and the adaptation of the models for operation in a resource-constrained environment. Additionally, it details the integration of the node with a cloud platform that enables both system monitoring and real-time awareness of acoustic events. The demo is currently deployed in an open industrial environment—the Port of Valencia (Spain)—where it supports the optimization of logistics operations and the identification of noise sources.

Acknowledgements

This research was partially funded by the Valencian Institute for Business Competitiveness (LIASound: IMDEEA/2024/110), and by the Ministry of Science, Innovation and Universities of Spain and the EU (STARRING-NEURO: PID2022-137048OA-C44).

References

[1] Kong, Q., Cao, Y., Iqbal, T., Wang, Y., Wang, W., and Plumbley, M. D., "Panns: Large-scale pretrained audio neural networks for audio pattern recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28, pp. 2880–2894, 2020.

Audio Engineering Society

Late Breaking Demo Paper

Presented at the AES International Conference on
Artificial Intelligence and Machine Learning for Audio
2025 September 8–10, London, UK

This Late Breaking Demo Paper was selected after a minimal screening process and was not peer reviewed. This Paper has been reproduced from the author's advance manuscript without editing, corrections, or consideration by the Review Board. The AES takes no responsibility for the contents. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Audio Engineering Society.

Class-Aware Hybrid Ensemble for Query-by-Vocal Imitation

Rahul Peter[1] and Vivek Mohan[1]

[1]*Aalto University*

Correspondence should be addressed to Rahul Peter (rahul.peter@aalto.fi)

ABSTRACT

We present a Query-by-Vocal Imitation (QBV) system that ensembles four pretrained audio encoders (MobileNetV3, PANNs, PaSST, BEATs), each contrastively fine-tuned on VimSketch pairs. Two inference fusions are compared: **System 1**, a novel *class-aware* mixture-of-experts using per-class Mean Reciprocal Rank (MRR) for soft weights; and **System 2**, *global* fusion (weighted average) without class information. System 1 attains the best MRR (0.3191) and class-wise metrics, outperforming System 2 and single-model baselines. *Code* is available for reproducibility.

1 Introduction

Query-by-Vocal Imitation (QBV) involves predicting a reference audio file from a vocal imitation of it, circumventing the need for precise text metadata. The QVIM challenge [1] has highlighted the difficulties of this task in which human vocal imitations exhibit wide timbral variation, class distributions are often imbalanced, and mapping between vocal renditions and diverse sound classes requires robust representations [2]. In our evaluations, we report Mean Reciprocal Rank (MRR) [3], a metric that reflects the average of the reciprocal ranks of the first correct result. Recent dual-encoder CNNs such as MobileNetV3 [4], with contrastive losses, improved MRR scores. However, single-encoder models still struggle across heterogeneous sound categories [2]. We hypothesize that ensembling diverse encoders, combined with *class-specific fusion* or *global fusion methods (weighted average)*, can better accommodate the varied acoustic characteristics and differences across classes inherent in the QVIM task.

2 Methods

2.1 Encoders & Training

We fine-tune four AudioSet-pretrained encoders: MobileNetV3, PANNs [5], PaSST [6], and BEATs [7] using InfoNCE [8] to align vocal-imitation and target-sound embeddings in a dual-encoder setup on VimSketch [9] Pairs. For MobileNet, we apply pitch shift, Gaussian noise, and spectrogram mixing [10] augmentations.

2.2 Fusion at Inference

System 1 (Class-Aware). For each class c, we compute per-class validation MRRs for all encoders and, at inference, set

$$w_{i,c} = \frac{\exp\left(\mathrm{MRR}_{i,c}/T\right)}{\sum_{j=1}^{N}\exp\left(\mathrm{MRR}_{j,c}/T\right)}, \tag{1}$$

with T=0.05 (empirically selected) controlling the sharpness of the softmax across the encoders. If the top MRR

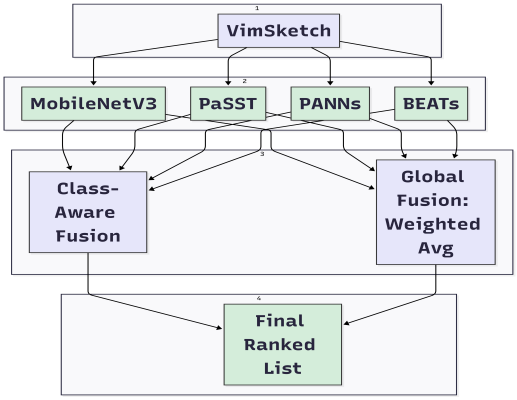


Fig. 1: System overview

Table 1: Retrieval performance (higher is better).

Model	MRR	c-MRR	c-NDCG
System 1 (class-aware)	**0.3191**	**0.5564**	**0.6912**
System 2 (global)	0.3026	0.5090	0.6601
MobileNetV3 only	0.2726	0.4683	0.6463

exceeds the runner-up by ≥ 0.03 (empirically chosen), we select that encoder; otherwise, we apply the softmax-weighted fusion.

System 2 (Global). Without using class information, we apply simple score averaging with global fusion weights:

$$S_{\text{final}} = \sum_{i=1}^{N} w_i S_i, \tag{2}$$

where S_i denotes the retrieval score from encoder i, and w_i is its corresponding global fusion weight, with $\sum_{i=1}^{N} w_i = 1$.

where the weights are empirically defined as:

$$(w_{\text{MNetV3}}, w_{\text{BEATs}}, w_{\text{PaSST}}, w_{\text{PANNs}}) = (0.4, 0.3, 0.2, 0.1). \tag{3}$$

3 Results

We evaluate on the DEV dataset [1], reporting Mean Reciprocal Rank (MRR), class-wise MRR (c-MRR), and class-wise Normalized Discounted Cumulative Gain (c-NDCG). Table 1 summarizes retrieval performance.

System 1 yields a 5.5% absolute MRR improvement over the best global fusion and, in a MUSHRA [11] listening evaluation for the QVIM challenge, achieved a subjective MRR of 0.382 and a Mean Opinion Score of 54.7.

4 Discussion

The class-aware fusion consistently boosts MRR for difficult classes by leveraging encoder specialization. Future work includes optimizing the temperature (T) and dominance threshold hyperparameters in System 1 for improved encoder selection.

5 Summary

We present a concise 2-page LBDP where an ensemble of four contrastively fine-tuned encoders is discussed. System 1 (class-aware fusion) improves MRR by ~17% relative over the best single model.

References

[1] Audio Engineering Society, "Query-by-Vocal Imitation (QVIM) Challenge," `https://qvim-aes.github.io/`, 2025.

[2] Greif, J. and Schmid, F. e. a., "Improving query-by-vocal imitation with contrastive learning and audio pretraining," 2024.

[3] "On Search Engine Evaluation Metrics - Section 4.2," https://arxiv.org/pdf/1302.2318, 2012.

[4] Howard, A. and Sandler, M. e. a., "Searching for MobileNetV3," ICCV, 2019.

[5] Kong, Q. and Cao, Y. e. a., "PANNs: Large-Scale pretrained audio neural networks for audio pattern recognition," IEEE TASLP, 2019.

[6] Koutini, K., Schlüter, J., Eghbal-zadeh, H., and Widmer, G., "Efficient Training of Audio Transformers with Patchout," in *Interspeech 2022*, pp. 2753–2757, ISCA, 2022.

[7] Chen, S., Wu, Y., and Wang, C. e. a., "BEATs: Audio pre-training with acoustic tokenizers," ICML, 2022.

[8] Oord, A. v. d., Li, Y., and Vinyals, O., "Representation learning with contrastive predictive coding," 2018.

[9] Kim, B., Cartwright, M., Pishdadian, F., and Pardo, B., "VimSketch Dataset," 2019.

[10] Yun, S., Han, D., Oh, S. J., Chun, S., Choe, J., and Yoo, Y., "CutMix: Regularization strategy to train strong classifiers with localizable features," 2019.

[11] "B. Series, 'Method for the subjective assessment of intermediate quality level of audio systems,' International Telecommunication Union Radiocommunication Assembly, vol. 2, 2014." 2014.

Audio Engineering Society

Late Breaking Demo Paper

Presented at the AES International Conference on
Artificial Intelligence and Machine Learning for Audio
2025 September 8–10, London, UK

This Late Breaking Demo Paper was selected after a minimal screening process and was not peer reviewed. This Paper has been reproduced from the author's advance manuscript without editing, corrections, or consideration by the Review Board. The AES takes no responsibility for the contents. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Audio Engineering Society.

Investigating the use of Deep Convolutional Neural Networks for Direction-of-Arrival Estimation on Raw Stereo Audio

Samuel Hobern[1], Alan Archer-Boyd[2], and Damian Murphy[1]

[1]*AudioLab, School of Physics, Engineering and Technology, University of York, York, UK*
[2]*BBC Research and Development, The Lighthouse, White City Place, 201 Wood Lane, London, W12 7TQ*

Correspondence should be addressed to Samuel Hobern (sam.hobern@york.ac.uk)

ABSTRACT

This paper outlines the use of raw stereo audio as the input to a Deep Convolutional Neural Network for the purpose of Direction-of-Arrival (DOA) estimation based on source localisation over the frontal hemisphere of a 50-point Lebedev loudspeaker array using a pair of spaced stereo microphones. Results show the model is capable of a high classification accuracy with evidence of some ability to generalise to unseen data highlighting the benefits of raw audio as the input feature to models.

1 Introduction

Sound Source Localisation (SSL) is a challenging task with numerous applications including robotics, fault detection in industrial machinery, and audio upmixing [1]. This challenge can be solved by estimating the Direction-of-Arrival (DOA). SSL is typically tackled using Machine Learning (ML) based methods due to their high accuracy and robustness to noise [1].

2 Methods

In this work, ML based DOA estimation employed a classification approach using a modified version of the *SampleCNN* architecture [2]. In order to avoid confusion, the model created in this paper shall be referred to as *SampleDOA*.

Many models for this task use empirically-derived features such as the Generalised Cross Correlation Phase Transform (GCC-PHAT) or Mel-Spectrograms. However, these can inadvertently bias the model by limiting the information the model has to train on. Instead, the models were trained on raw, stereo audio. Impulse responses were recorded within the University of York AudioLab's 50 loudspeaker Lebedev sampled sphere for numerous stereo microphone configurations including a 40cm spaced pair of AKG C414s set to the omnidirectional mode and denoted as 'AB_Omni_40' [3]. The impulses responses were convolved with the NIGENS database [4]. The 8 splits allocated by NIGENS were used with 6 for training, 1 for validation, and 1 for testing.

A model was created, named *SampleDOA_5_6* with

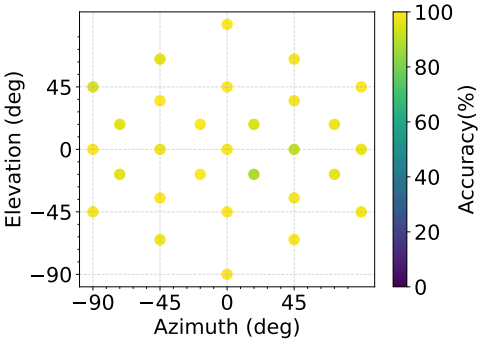


Fig. 1: *SampleDOA*_5_6 Azimuth Accuracy on the AB_Omni_40 Dataset

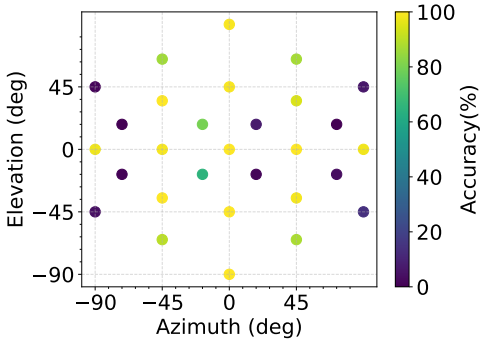


Fig. 2: *SampleDOA*_5_6 Azimuth Accuracy on the AB_Omni_30 Dataset

6 convolutional layers each with a convolutional filter size of 5. With an input of 78125 samples (9.77s at an 8kHz sampling rate), the model is capable of extracting sample level features. Each layer consists of a 1D Convolution, followed by Batch Normalisation, ReLU activation, and a Max Pooling operation. The *SampleDOA* model was tasked with estimating the azimuth of sound sources in the frontal hemisphere only; the azimuth (θ) and elevation (ϕ) were both bounded between $[-90°, 90°]$. The Adam optimiser was used with a learning rate of $1e-5$, and a batch size of 8.

3 Results

After 1000 epochs, the SampleDOA_5_6 model achieved an accuracy of 97.6% as shown for each loudspeaker position in Figure 1. The azimuth estimation accuracy is largely consistent across loudspeaker positions with slight deviations potentially caused by having fewer samples of those loudspeaker positions being included in the training data. To evaluate the performance of the model, it was tested against the 'AB_Omni_30' dataset. This dataset follows the same microphone configuration as 'AB_Omni_40' except the spacing between the microphones is 30cm instead of 40cm. As seen in Figure 2, the classification accuracy of the model decreases to approximately 62.5%. However, the model begins to show evidence of generalisation. At an elevation of 0° the model maintains the same level of accuracy along all loudspeakers as it did for the AB_Omni_40 dataset. This is also true for the loudspeakers positioned along an azimuth of 0°. Performance suffered most significantly at loudspeaker positions at the boundaries of the hemisphere as denoted by the darker spots in Figure 2.

4 Discussion

The results indicate the *SampleDOA* architecture is capable of performing Direction-of-Arrival estimation on raw stereo audio at a high degree of accuracy. The performance on the AB_Omni_30 dataset suggests evidence of the model beginning to generalise. However, further investigation is required. Training the model on multiple datasets of differing microphone configurations may improve its ability to generalise and will be investigated in future work.

References

[1] Khan, A., Waqar, A., Kim, B., and Park, D., "A review on recent advances in sound source localization techniques, challenges, and applications," *Sensors and Actuators Reports*, 9, p. 100313, 2025, ISSN 2666-0539, doi:10.1016/j.snr.2025.100313.

[2] Lee, J., Park, J., Kim, K. L., and Nam, J., "SampleCNN: End-to-End Deep Convolutional Neural Networks Using Very Small Filters for Music Classification," *Applied Sciences*, 8(1), p. 150, 2018, ISSN 2076-3417, doi:10.3390/app8010150, number: 1 Publisher: Multidisciplinary Digital Publishing Institute.

[3] Turner, D. and Murphy, D., "Dataset of stereo and multi-channel IRs for a 50-point Lebedev quadrature." 2023, doi:10.5281/zenodo.7990195.

[4] Trowitzsch, I., Taghia, J., Kashef, Y., and Obermayer, K., "The NIGENS General Sound Events Database," 2020, doi:10.48550/arXiv.1902.08314, arXiv:1902.08314 [cs].

Audio Engineering Society

Late Breaking Demo Paper

Presented at the AES International Conference on
Artificial Intelligence and Machine Learning for Audio
2025 September 8–10, London, UK

This Late Breaking Demo Paper was selected after a minimal screening process and was not peer reviewed. This Paper has been reproduced from the author's advance manuscript without editing, corrections, or consideration by the Review Board. The AES takes no responsibility for the contents. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Audio Engineering Society.

Beat-Based Rhythm Quantization of MIDI Performances

Maximilian Wachter[1], Sebastian Murgul[1,2], and Michael Heizmann[2]

[1]*Klangio GmbH, Karlsruhe, Germany*
[2]*Institute of Industrial Information Technology, Karlsruhe Institute of Technology, Karlsruhe, Germany*

Correspondence should be addressed to Maximilian Wachter (max.wachter@klang.io)

ABSTRACT

We propose a transformer-based rhythm quantization model that incorporates beat and downbeat information to quantize MIDI performances into metrically-aligned, human-readable scores. We propose a beat-based preprocessing method that transfers score and performance data into a unified token representation. We optimize our model architecture and data representation and train on piano and guitar performances. Our model exceeds state-of-the-art performance based on the MUSTER metric.

1 Introduction

This paper introduces a novel transformer-based approach for beat-based rhythm quantization of symbolic MIDI performances. Rhythm quantization aims to recover the intended notated rhythm from expressive performances, a fundamental task in music information retrieval and automatic music trancription. Our method uniquely integrates beat information and performance timing into a unified tokenized input representation, enabling the model to learn rhythmic structure by modeling timing deviations relative to an underlying metrical grid. Unlike most quantization models, our model is capable of leveraging metronome information, which entails the possibility of completely eliminating the uncertainty of beat estimations.

2 Methods

To support diverse musical contexts, we designed a flexible preprocessing pipeline that uses beat estimations or ground truth beats as a priori information and adapts to multiple time signatures without requiring explicit time signature tokens. This enables the model to generalize and successfully quantize rhythms in time signatures unseen during training. We propose a confusion-based evaluation metric tailored to beat-aligned quantization, incorporating both onset accuracy and note value correctness, which is better suited to evaluating timing on a score-level than frame-based metrics.

We used the T5 transformer for our model architecture, which we subsequently optimized in terms of efficiency and quality. For training our model we used the ASAP dataset [1], a piano dataset containing performance

MIDI files with accompanying scores as well as beat and downbeat annotations. We trained our model on N measure sequences, assuming a one-to-one correspondence. Due to the sometimes poor alignment between scores and performances we filtered the dataset based on how well input and target sequences matched. We subsequently optimized the model based on input sequence length, note ordering, and data augmentation techniques such as pitch transposition, note deletion, and duration noise in order to maximize musical note onset accuracy.

3 Experiments

Our optimized checkpoint achieves an onset F1 score of 97.3% on the ASAP dataset and a note value accuracy of 83.3%. Extending beyond piano, we adapted the model to guitar performances using the Leduc dataset, demonstrating that instrument-specific training yields improved quantization results. This highlights the importance of capturing instrument-dependent rhythmic interpretation characteristics for precise quantization.

Finally, we benchmarked our approach against state-of-the-art methods using the test and training splits defined in the ACPAS dataset for computing the MUSTER metric [2]. In Table 1 we compared the onset and offset error rates ε_{onset} and ε_{offset} to other state-of-the-art probabilistic and deep learning-based approaches. We showed that our model is able to surpass state-of-the-art models in ε_{onset} by using beat annotations, while getting surpassed in ε_{offset} only by [3].

Table 1: Comparison of onset-time (ε_{onset}) and offset-time (ε_{offset}) error rates using the MUSTER metric.

Method	ε_{onset}	ε_{offset}
Neural Beat Tracking [4]	68.28	54.11
End-to-End PM2S [3]	15.55	**23.84**
HMMs (J-Pop) [5]	25.02	29.21
HMMs (classical) [5]	22.58	29.84
Our Model	**12.30**	28.30

4 Discussion

Our results confirm that explicit beat information can yield significant improvements in onset quantization performance. However, offset accuracy remains challenging due to our model's current inability to detect 32nd notes. Compared to similar models, ours excels in scenarios where explicit beat information is available like in cases where performances are recorded to a metronome. If metronome information is not available the performance is however highly dependent on beat estimation quality.

5 Summary

Our findings suggest that transformer-based models with beat-aware tokenization offer a powerful framework for expressive rhythm quantization across instruments and meters. Future work will explore incorporating explicit time signature modeling, and expanding to irregular meters and note values, aiming for even greater generalization and musical fidelity.

References

[1] Foscarin, F., Mcleod, A., Rigaux, P., Jacquemard, F., and Sakai, M., "ASAP: a dataset of aligned scores and performances for piano transcription," in *Proceedings of the 21st International Society for Music Information Retrieval Conference (ISMIR)*, 2020.

[2] Nakamura, E., Benetos, E., Yoshii, K., and Dixon, S., "Towards Complete Polyphonic Music Transcription: Integrating Multi-Pitch Detection and Rhythm Quantization," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.

[3] Beyer, T. and Dai, A., "End-to-end Piano Performance-MIDI to Score Conversion with Transformers," in *Proceedings of the 25th International Society for Music Information Retrieval Conference (ISMIR)*, 2024.

[4] Liu, L., Kong, Q., Morfi, G., Benetos, E., et al., "Performance MIDI-to-score conversion by neural beat tracking," in *Proceedings of the 23rd International Society for Music Information Retrieval Conference (ISMIR)*, 2022.

[5] Shibata, K., Nakamura, E., and Yoshii, K., "Non-local musical statistics as guides for audio-to-score piano transcription," *Information Sciences*, 566, pp. 262–280, 2021.

Audio Engineering Society

Late Breaking Demo Paper

Presented at the AES International Conference on
Artificial Intelligence and Machine Learning for Audio
2025 September 8–10, London, UK

This Late Breaking Demo Paper was selected after a minimal screening process and was not peer reviewed. This Paper has been reproduced from the author's advance manuscript without editing, corrections, or consideration by the Review Board. The AES takes no responsibility for the contents. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Audio Engineering Society.

Decoding Melodic Acoustic Features from Neural Data

Zorka Bozilovic[1] and Iran R. Roman[1]

[1]*Queen Mary University of London*

Correspondence should be addressed to Zorka Bozilovic (zorka.bozilovic@gmail.com)

ABSTRACT

We decoded acoustic features of music from EEG using machine learning, focusing on envelopes and onsets and comparing musicians and non-musicians. Results showed higher accuracy when decoding from musicians, especially in the theta band. These findings highlight measurable differences in neural representation of music and suggest potential applications in neural audio engineering, assistive technologies, and rehabilitation.

1 Introduction

This late-breaking paper investigates how auditory stimuli relate to brain activity by decoding acoustic features of music from EEG. This deepens understanding of how music transforms into neural signals and suggests directions for neural audio engineering (e.g., reconstructing imagined sounds) which may benefit assistive technologies and neurological rehabilitation [1]. Brain encoding and decoding have both been studied with audio [2]: encoding predicts EEG, while decoding reconstructs stimuli. A recent study examined musical feature encoding across EEG frequency bands [3], but did not explore decoding. We applied machine learning to a public EEG dataset recorded as participants listened to melodies. **Our paper's contributions are:** (1) Decoded melodic envelope and onsets from EEG; (2) Evaluated decoding across musicians vs. non-musicians and EEG bands; (3) Open-sourced the code.[1]

[1] github/ZorkaBozilovic/audio-brain-decoding

2 Methods

Dataset. We used the dataset from [4] (analyzed in [3]), which includes EEG (64ch) from 10 non-musicians and 10 pianists. Each individual passively listened to 10 melodies while EEG was recorded. This was repeated three times, yielding 30 trials per participant. EEG was re-referenced to the mean of the two mastoid channels.

Preprocessing. EEG was filtered into delta (1–4 Hz), theta (4–8 Hz), alpha (8–12 Hz), and beta (12–30 Hz) bands[2]. We extracted the envelopes and onset features from the dataset's melodies.

Decoding acoustic features from neural data. We used the temporal response function (TRF, a linear model mapping neural responses to stimulus features via convolution [2]). When decoding, EEG is the input and musical features are the output. One model was fit

[2] Gamma was excluded due to the sampling rate in the pipeline [2].

per participant, with separate models for envelope and onsets. We used `eelbrain`'s boosting function with a 300ms window and evaluated decoding accuracy using Pearson's r between predicted and actual features.

3 Results

Figure 1 shows decoding performance across frequency bands. Correlation coefficients were significantly higher for musicians than non-musicians, with the theta band (4–8 Hz) yielding best decoding overall. Envelope decoding generally outperformed onsets decoding.

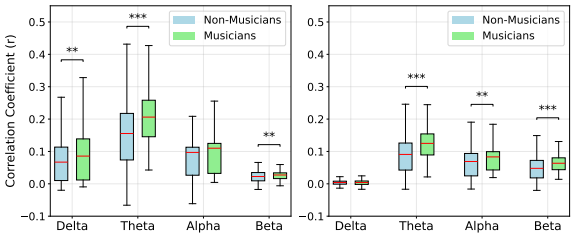


Fig. 1: Decoding correlation coefficients across EEG bands for envelopes (left) & onsets (right). The mean is shown in red. T-tests show significant differences across non-musicians & musicians (***: $p < 0.001$, **: $p < 0.01$, *: $p < 0.05$).

Figure 2 shows that musicians had stronger filter amplitudes and stronger voltage field, particularly over right-lateralized channels. This aligns with prior findings on hemispheric lateralization for musicians [5].

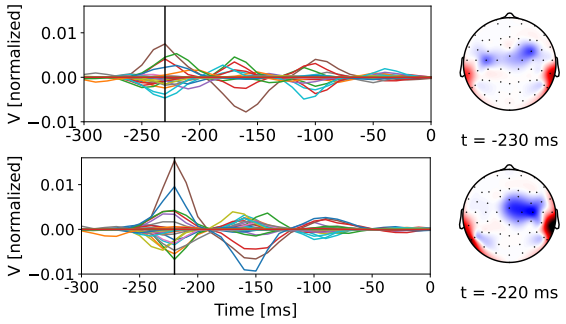


Fig. 2: Mean filters learned to decode musical envelopes from theta-band. Left: butterfly plots show the filter for each EEG channel. Right: topographic maps at peak latencies show spatial distribution of voltage field strength. Top: non-musicians (N=10); bottom: musicians (N=10).

Figure 3 shows three representative decoding examples. The top (who is a musician) shows closer alignment in shape and timing between decoded and true envelope, particularly toward the end. In contrast, non-musicians (middle and bottom) show reduced synchrony.

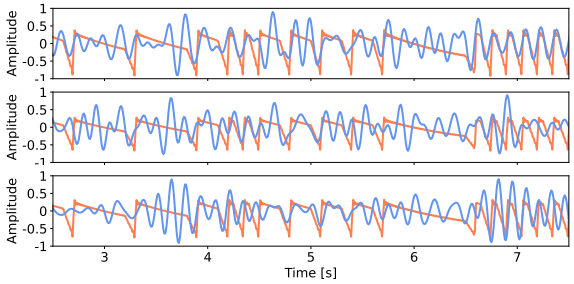


Fig. 3: Examples of decoded (blue) vs. true (orange) envelopes in the theta band. Top: high correlation ($r = 0.43$, musician); middle: average performance ($r = 0.18$, non-musician); bottom: poor decoding ($r = -0.02$, non-musician).

4 Conclusion

Musical expertise enables more accurate decoding of musical features from EEG. Performance was highest in the theta band. Our findings suggest that musical training yields measurable differences in brain responses to music, offering valuable insights for future neural audio engineering interfaces.

References

[1] Moses, D. A. et al., "Neuroprosthesis for decoding speech in a paralyzed person with anarthria," *New England Journal of Medicine*, 385(3), 2021.

[2] Brodbeck, C. et al., "Eelbrain, a Python toolkit for time-continuous analysis with temporal response functions," *eLife*, 12, 2023.

[3] Galeano-Otálvaro, J.-D. et al., "Neural encoding of melodic expectations in music across EEG frequency bands," *European Journal of Neuroscience*, 60(11), pp. 6734–6749, 2024.

[4] Di Liberto, G. M. et al., "Cortical encoding of melodic expectations in human temporal cortex," *Elife*, 9, p. e51784, 2020.

[5] Koelsch, S. et al., "Effects of musical expertise on the early right anterior negativity: an event-related brain potential study," *Psychophysiology*, 2002.

Audio Engineering Society

Late Breaking Demo Paper

Presented at the AES International Conference on
Artificial Intelligence and Machine Learning for Audio
2025 September 8–10, London, UK

This Late Breaking Demo Paper was selected after a minimal screening process and was not peer reviewed. This Paper has been reproduced from the author's advance manuscript without editing, corrections, or consideration by the Review Board. The AES takes no responsibility for the contents. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Audio Engineering Society.

Perceptual Probing of Deep Audio Representations

Etienne Bost[1], Richard Kronland-Martinet[2], Mitsuko Aramaki[2], Sølvi Ystad[2], Thierry Artières[1], and Thomas Schatz[1]

[1]*Aix Marseille Univ, Centrale Marseille, CNRS, LIS, Marseille, France*
[2]*Aix Marseille Univ, Centrale Marseille, CNRS, PRISM, Marseille, France*

Correspondence should be addressed to Etienne Bost (etienne.bost@lis-lab.fr)

Introduction: Deep audio representation models offer promising capabilities for applications such as sound synthesis, editing, and classification among others. A growing number of state-of-the-art models have been developed for general-purpose audio representation, and most perform comparably on standard classification benchmarks. However, their underlying similarities, differences, and limitations remain poorly understood beyond these coarse-grained metrics. In this study, we assess the extent to which these models capture perceptually meaningful structure by evaluating their alignment with human judgments along three high-level dimensions identified in ecological theories of sound perception: material, hollowness, and spatial extension. Understanding how well model representations reflect such perceptual dimensions is essential for enabling controllable and perceptually grounded applications in audio generation and manipulation such as audio editing or synthesis.

Methods: We conduct a fine-grained evaluation of popular self-supervised and supervised audio models using a controlled dataset of impact sounds rated by human listeners along key perceptual dimensions. Unlike standard benchmarks such as AudioSet [1], which produce a single performance score over diverse and often loosely defined categories, our approach directly assesses the structure of model representations in relation to validated perceptual judgments. For instance, if one seeks a representation suitable for making a hollow object sound full, the AudioSet score offers no relevant insight. To compare how well model representations align with human perception, we adopt a representational similarity analysis (RSA) framework. Human data consists of perceptual judgments by 17 participants across several dimensions (material, hollowness, spatial extension), while model representations are high-dimensional activation patterns. For each, we compute how dissimilar different sounds are from one another, forming pairwise similarity structures. We then quantify the alignment between human and model similarity spaces using Centered Kernel Alignment (CKA) [2], a robust correlation metric between 0 and 1 that quantifies alignment between the two spaces, regardless of their scale or dimensionality. Higher CKA values indicate stronger correspondence with human perceptual structure.

Results and perspective: Our results (Figure 1) show that, despite comparable classification performance on AudioSet (mean average precision scores ranging from 0.47 to 0.506, except for Yamnet) the alignment of model representations with human perceptual judgments varies substantially. The difference between the best- and worst-performing models is nearly twofold. CLAP, trained with a contrastive audio-text objective, consistently achieves the highest alignment across all tested perceptual dimensions. Audio-only models such

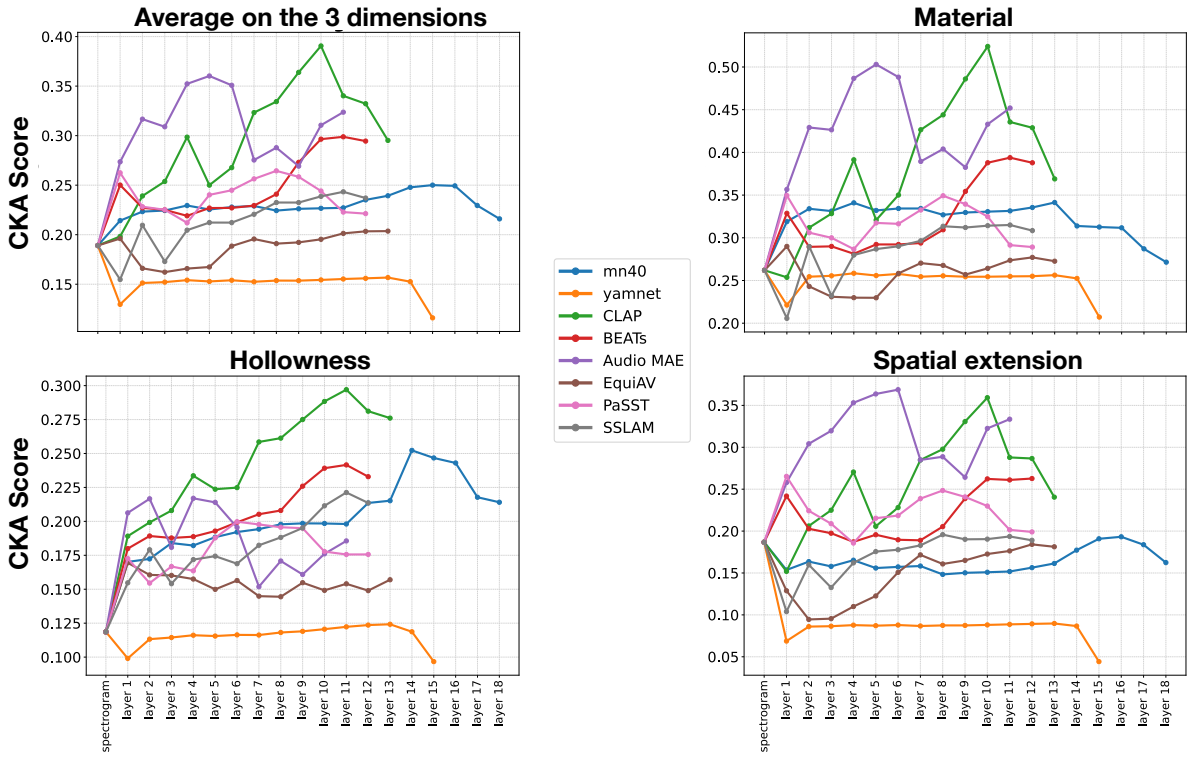


Fig. 1: Correlation between model representations and human judgments.

as AudioMAE [3], SSLAM [4], PaSST [5], and MN40 [6] perform relatively well on material and spatial extension, but generally show weaker alignment on the hollowness dimension. As expected, Yamnet, an earlier convolutional model with lower classification accuracy, also ranks lowest in perceptual alignment, confirming that weaker representations result in poorer performance on our metrics. Interestingly, EquiAV [7], an audiovisual model with strong classification results, exhibits the weakest alignment with perception among the recent state-of-the-art models, suggesting that high benchmark accuracy does not necessarily reflect perceptually meaningful internal structure. Finally, we observe that in models such as BEATs [8] and CLAP [9], perceptual alignment increases consistently across layers, with later (deeper) representations showing the highest correspondence with human judgments supporting the view that deeper layers encode more abstract, perceptually relevant information. Future work will aim to investigate the underlying causes of these differences in perceptual alignment. In particular, we seek to understand whether CLAP's superior performance and EquiAV's weaker alignment are primarily driven by their multimodal training, or whether other factors such as model size also play a role. Expanding the analysis to include perceptual dimensions related to actions (e.g., scraping, brushing) is another important direction. Moreover, we intend to assess whether the observed differences in representation structure translate into practical advantages in downstream applications. For instance, can CLAP representations facilitate more effective sound transformations such as converting a "hollow" sound into a "full" one compared to other models? Confirming such use-case relevance would further validate the utility of perceptually grounded evaluations.

References

[1] Gemmeke et al., "Audio set: An ontology and human-labeled dataset for audio events," in *ICASSP*, 2017.
[2] Williams, "Equivalence between representational similarity analysis, centered kernel alignment, and canonical correlations analysis," in *UniReps*, 2024.
[3] Huang et al., "Masked Autoencoders that Listen," 2023.
[4] Alex et al., "SSLAM: Enhancing Self-Supervised Models with Audio Mixtures for Polyphonic Soundscapes," in *ICLR*, 2025.
[5] Koutini et al., "Efficient training of audio transformers with patchout," *Interspeech*, 2021.
[6] Schmid et al., "Efficient Large-Scale Audio Tagging Via Transformer-to-CNN Knowledge Distillation," in *ICASSP*, 2023.
[7] Kim et al., "Equiav: Leveraging equivariance for audio-visual contrastive learning," 2024.
[8] Chen et al., "Beats: Audio pre-training with acoustic tokenizers," 2022.
[9] Elizalde et al., "CLAP: learning audio concepts from natural language supervision," in *ICASSP*, 2023.

Audio Engineering Society

Late Breaking Demo Paper

Presented at the AES International Conference on
Artificial Intelligence and Machine Learning for Audio
2025 September 8-10, London, UK

This Late Breaking Demo Paper was selected after a minimal screening process and was not peer reviewed. This Paper has been reproduced from the author's advance manuscript without editing, corrections, or consideration by the Review Board. The AES takes no responsibility for the contents. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Journal of the Audio Engineering Society.

Black-Box Modeling of Immersive Reverberation with Convolutional Wavenet Audio Effects Modeling Network

Yavuz Buruk[1] and Taylan Özdemir[1]

[1]Istanbul Technical University

Correspondence should be addressed to Yavuz Buruk (yavuzburuk@gmail.com)

ABSTRACT

Recent advances in black-box modeling using deep neural networks have revolutionized the emulation of nonlinear audio effects. This paper investigates the application of Convolutional WaveNet Audio Effects Modeling Networks (CWAFx) of reverberation in immersive audio, such as Dolby Atmos environments. Results reveal the promise of multi-output networks in modeling complex, multi-channel reverberant responses with acceptable perceptual fidelity.

1 Introduction

Deep learning approaches—especially black-box models using convolutional and recurrent neural networks—have enabled end-to-end modeling of audio effects using only input-output waveform pairs. Immersive sound design and mixing have rapidly evolved with the advent of spatial audio formats like Dolby Atmos. This study applies black-box modeling to immersive reverberation using the Convolutional Wavenet Audio Effects Modeling Network (CWAFx), trained on a custom dataset rendered across 9 spatial channels.

2 Background

Black-box modeling has emerged as a practical alternative to white- and gray-box techniques for simulating nonlinear effects such as guitar amplifiers [1], distortion [2] and dynamic compressors [3]. Architectures like Convolutional Audio Effects Modeling Network (CAFx), Convolutional Recurrent Audio Effects Modeling Network (CRAFx), WaveNet and CWAFx have been used for short-term and long-term memory effects [4].

3 Methodology

In this study, the IDMT-SMT-Audio-Effects database [5] was utilized, focusing exclusively on electric guitar samples without any audio effects. The NoFX dataset comprises 624 monophonic WAV files, each 2 seconds in duration, recorded at a sampling rate of 44.1 kHz and a bit depth of 16 bits. For consistency with the immersive audio processing, all samples were resampled to 48 kHz.

Our objective is to develop a black-box model for reverberation that takes a stereo reverberant waveform as input and produces an immersive reverberation as output. To construct the stereo and

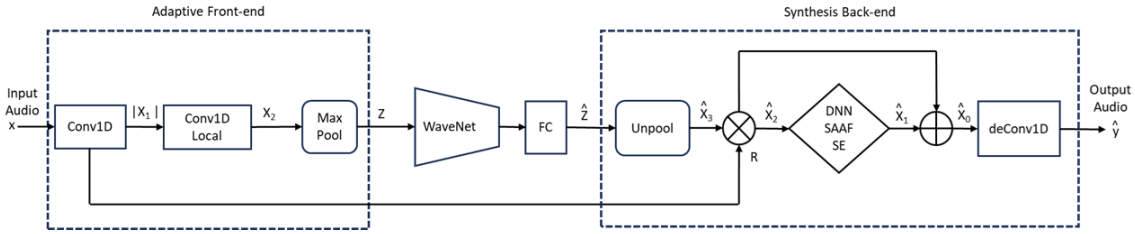


Figure 1. Convolutional Wavenet Audio Effects Modeling Network (CWAFx).

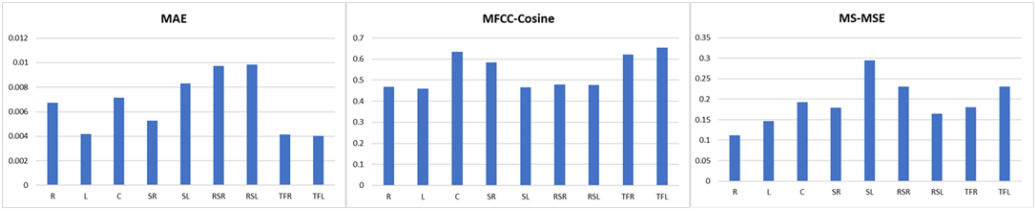


Figure 2. MAE, MFCC-Cosine and MS-MSE values. (R: Right, L: Left, C: Center, SR/SL: Surround R/L, RSR/RSL: Rear Surround R/L, TFR/TFL: Top Front R/L)

Atmos reverberant datasets for training, samples from NoFX dataset were rendered using high-end spatial reverberation plugins, FabFilter Pro-R2, Paragon and Cinematic Rooms Pro, with an additional 2-second tail. The proposed CWAFx model is trained on these paired datasets, consisting of 2-channel stereo inputs and 9-channel Atmos outputs.

4 Experiments

The rendered stereo and Atmos waveforms are amplitude normalized. The test and validation samples correspond to 20% of the stereo and Atmos datasets each. Input and output waveforms are downsampled to 16 kHz for training and outcomes are resampled back to 48 kHz. The model was trained over 250 epochs using the Adam optimizer and MAE (Mean Absolute Error) for all 9 immersive channels for the selected presets.

5 Results

Figure 2 shows the MAE, MFCC-Cosine Similarity, and MS-MSE metrics of the proposed black-box model, confirming its measurable performance. While some metallic and noisy artifacts remain, the results provide a solid foundation for future improvements through parameter optimization and extended training.

6 Conclusions

These findings offer preliminary support for the potential of neural networks in modeling immersive reverberation. The majority of the existing reverberation tools are limited to producing only stereo outputs. If the model to be developed and refined successfully generates high-fidelity, high resolution immersive audio, it may serve as a basis for converting existing stereo reverb plugins into immersive plugins.

References

[1] Wright, A., Damskägg, E. P., Juvela, L., & Välimäki, V. (2020). Real-time guitar amplifier emulation with deep learning. *Applied Sciences*, 10(3), 766.

[2] Damskägg, E. P., Juvela, L., & Välimäki, V. (2019). Real-time Modeling of Audio Distortion Circuits with Deep Learning. *In Sound and music computing conference* (pp. 332-339).

[3] Steinmetz, C. J., & Reiss, J. D. (2021). Efficient Neural Networks for Real-time Modeling of Analog Dynamic Range Compression. *arXiv preprint arXiv: 2102.06200*.

[4] Martínez Ramírez, M. A., Benetos, E., & Reiss, J. D. (2020). Deep Learning for Black-Box Modeling of Audio Effects. *Applied Sciences*, 10(2), 638.

[5] Stein, M. IDMT-SMT-Audio-Effects Dataset. Zenodo. DOI: 10.5281/zenodo.7544032. Available at: https://doiorg/105281/zenodo7544032

Audio Engineering Society

Late Breaking Demo Paper

Presented at the AES International Conference on
Artificial Intelligence and Machine Learning for Audio
2025 September 8–10, London, UK

This Late Breaking Demo Paper was selected after a minimal screening process and was not peer reviewed. This Paper has been reproduced from the author's advance manuscript without editing, corrections, or consideration by the Review Board. The AES takes no responsibility for the contents. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Audio Engineering Society.

Transformer-Based Sustain Pedal Reconstruction for Expressive Piano Performance MIDI

Wenhao Liu[1], György Fazekas[1], and Jingjing Tang[1]

[1]*Center for Digital Music, Queen Mary University of London, London, United Kingdom*

Correspondence should be addressed to Wenhao Liu (w.liu@se20.qmul.ac.uk)

ABSTRACT

Although expressive performance rendering (EPR) has been widely explored using various deep learning models, most existing studies focus primarily on note-level features in MIDI data, while overlooking the reconstruction of pedal information. We propose a transformer-based approach for reconstructing sustain pedal control in expressive piano performance MIDI, enabling accurate prediction of pedal onset time and duration.

1 Introduction

The sustain pedal plays a vital role in piano performance, serving as a key component of musical expressiveness by enabling smooth legato articulation, extended resonance, and enhanced harmonic continuity. In expressive performance rendering (EPR), the task of generating human-like performances from musical inputs such as scores or transcribed MIDI the pedal is essential for capturing the nuances of expressive intent. However, existing EPR systems either overlook pedal information [1, 2] or generate pedalling along with note features [3], achieving limited performance. Unlike discrete note events, pedals function as continuous controls that often span multiple notes, making joint prediction challenging.

We adapt a transformer-based framework [1] for reconstructing sustain-pedal controls in expressive piano performances. The performance MIDI files are tokenised using REMI [4] and converted into a structured multi-stream representation of note and pedal attributes for training. Preliminary results demonstrate effective pedal reconstruction across different task settings, suggesting the model's potential compatibility with EPR systems to enhance pedalling generation[1].

2 Proposed Method

We first convert the input MIDI file into structured multi-stream sequences of note attributes (pitch, velocity, onset position, duration), pedal controls (onset position and duration), and structural markers such as bar positions. These features are then tokenised using a REMI-style vocabulary [4], which allows explicit

[1]Code: https://github.com/liulizhizi/Pedal_genbymidibert

masking of pedal information while preserving other note features. We adopt duration rather than offset, as duration requires only one token for representation while offset needs two features (position and bar), making the encoding more compact and intuitive.

Two task settings are considered: (**1**) *partial reconstruction*, where the model predicts pedal durations given onset times, and (**2**) *full reconstruction*, where both pedal onset times and durations are predicted. This setup guides the model to reconstruct musically plausible sustain behaviour under varying levels of input degradation.

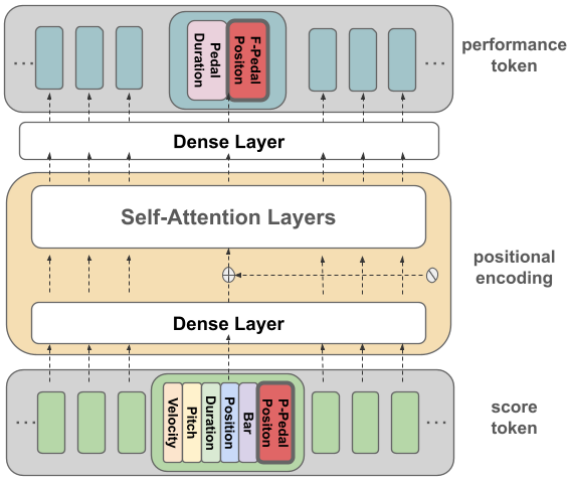


Fig. 1: Model architecture adapted from [1]. **P-Pedal Position** appears only in the *partial reconstruction* task, and **F-Pedal Position** appears only in the *full reconstruction* task.

As shown in Figure 1, our model builds on the architecture [1], with 8 transformer layers, 8 attention heads, and a hidden size of 512. The input embedding encodes note features in all settings and pedal position only in the *partial reconstruction* setting. We retain the encoder-only structure and adapt it to predict target pedal token sequences.

3 Preliminary Results

We evaluated the performance of our model under both *partial* and *full reconstruction* settings. We measured whether each note was covered by a pedal and computed the F1 score by comparing predictions with ground truth. In addition, we calculated the mean absolute error (MAE) and the root mean square error (RMSE) for pedal features in milliseconds. The results show higher accuracy in the *partial reconstruction* setting, with lower MAE and RMSE values and a higher F1-score compared to the *full reconstruction* setting.

Table 1: Objective evaluation metrics for pedal duration (Dur.) and onset time under *partial* and *full* reconstruction settings.

Settings	F1-score	MAE (ms)	RMSE (ms)
Partial	0.65	232.1	513.5
Full (Dur.)	0.54	466.2	758.8
Full (Onset)		428.7	573.3

4 Conclusion

We presented a Transformer-based system to reconstruct sustain pedal controls in expressive piano performance MIDIs, supporting both *partial* and *full* pedal reconstruction. Future work includes perceptual listening evaluation and evaluation on compatibility with existing EPR models.

5 Acknowledgement

This work is supported the UKRI Centre for Doctoral Training in Artificial Intelligence and Music (Grant no. EP/S022694/1), and China Scholarship Council [grant number 202008440382].

References

[1] Tang, J., Wiggins, G., and Fazekas, G., "Reconstructing Human Expressiveness in Piano Performances with a Transformer Network," *The 16th International Symposium on Computer Music Multidisciplinary Research*, 2023.

[2] Borovik, I. and Viro, V., "ScorePerformer: Expressive Piano Performance Rendering With Fine-Grained Control." in *ISMIR*, pp. 588–596, 2023.

[3] Jeong, D., Kwon, T., Kim, Y., Lee, K., and Nam, J., "VirtuosoNet: A Hierarchical RNN-based System for Modeling Expressive Piano Performance." in *ISMIR*, pp. 908–915, 2019.

[4] Huang, Y.-S. and Yang, Y.-H., "Pop Music Transformer: Beat-Based Modeling and Generation of Expressive Pop Piano Compositions," MM '20, p. 1180–1188, Association for Computing Machinery, New York, NY, USA, 2020.

Audio Engineering Society

Late Breaking Demo Paper

Presented at the AES International Conference on
Artificial Intelligence and Machine Learning for Audio
2025 September 8–10, London, UK

This Late Breaking Demo Paper was selected after a minimal screening process and was not peer reviewed. This Paper has been reproduced from the author's advance manuscript without editing, corrections, or consideration by the Review Board. The AES takes no responsibility for the contents. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Audio Engineering Society.

Latent rhythm transformation of drum recordings

Jason Hockman[1] and Jake Drysdale[2]

[1]*School of Digital Arts, Manchester Metropolitan University, Manchester, United Kingdom*
[2]*Beathoven.ai, Bengaluru, Karnataka, India*

Correspondence should be addressed to Jason Hockman (j.hockman@mmu.ac.uk)

ABSTRACT

A method is proposed for rhythm style transfer of multitimbral drum recordings via conditioning a VAE on rhythm and timbral features. Modulation and estimation of latent parameters and a novel resequencing process for reconstruction loss result in an end-to-end transformation circumventing manual segmentation and alignment.

1 Introduction

Drums play a crucial role in shaping the rhythmic and timbral identity of many forms of music. In multitimbral drum recordings, overlapping events, expressive timing, and timbral subtlety make rhythmic structure difficult to isolate and manipulate. Many professional studio workflows rely on *redrumming*, a technique that replaces or layers recorded drums with alternative recordings while preserving the original timing of a target performance [1]. Dedicated programs such as Recycle and modern DAWs (Logic, Ableton) provide manual or semi-automated workflows for time-based slicing and manipulation of waveforms; however, success of such processes is limited by spectral overlap of drums, requiring time-consuming manual intervention. In this paper, we extend the well-known RAVE method [2] to transformation of the rhythmic characteristics of a source drum recording to match the timing and drum classes present in a target recording.

2 Method

An overview of the proposed system is presented in Figure 1. The system operates in two stages: (1) VAE training and (2) transformer rhythm resequencing. For Stage 1, we adopt the RAVE [2] encoder E and generator G, training them on 3-sec drum recordings x_s to produce a latent representation z from the encoder outputs—mean μ and standard deviation σ. Stage 2 discards σ and introduces a lightweight transformer stack that performs style transfer on μ, yielding $\hat{\mu}$ and a subsequent estimation of $\hat{\sigma}$. Attention key k and value v are learned projections of μ, and query q is derived from conditional features c_t obtained from 5-class drum transcription (ADT) probabilities [3], $r \in R^{B\times 5\times T}$, and intermediate encoder activations $a \in R^{B\times C\times T}$ from target drum recordings x_t. These are projected into embeddings r_e and a_e via Conv1D sub-networks with LeakyReLU activations, to provide non-prescriptive attention guidance. Both r_e and a_e use four-layer Conv1D stacks

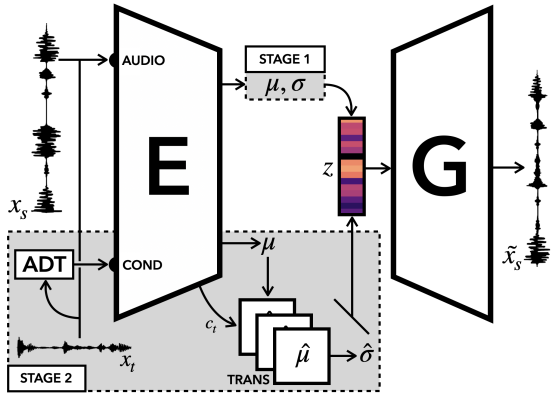


Fig. 1: Proposed model for rhythm transformation.

with LeakyReLU, with r_e kernel size 5, a_e size 3, each mapped to $2D$ channels ($D = 128$). Outputs are merged via gated fusion and projected to form q:

$$\lambda = \phi(W_g[r_e; a_e]), \qquad q = \lambda r_e + (1-\lambda)a_e. \quad (1)$$

$[\cdot;\cdot]$ denotes channel-wise concatenation, W_g is a learnable linear layer, and ϕ is the sigmoid function. The resulting query q is passed to a stack of L=3 transformer blocks with $H = 4$ attention heads each. Layer-wise cross-attention is applied using projections of the current $\mu^{(l)}$ in k and v:

$$\text{Attn}(q,k,v) = \text{softmax}\left(\frac{qk^\top}{\sqrt{d}} + \gamma B_{rel} + \delta B_{rhythm}\right) v, \quad (2)$$

where $\gamma, \delta \in R^H$ are learned per-head scaling factors, $B_{rel} \in R^{B\times H\times T\times T}$ is a learned relative positional bias, $B_{rhythm} \in R^{B\times H\times T\times T}$ is a rhythm bias derived from a linear projection of ADT probabilities and d is per-head dimensionality ($d = \frac{D}{H}$). $\hat{\mu}$ is updated at each layer l via linear interpolation using a learned scalar coefficient $\alpha^{(l)} \in [0,1]$. After the final layer, $\hat{\sigma}$ is estimated from $\hat{\mu}$ via convolutional projection f_σ:

$$\hat{\sigma} = \text{softplus}\left(f_\sigma * \hat{\mu}^{(L)}\right) + \varepsilon, \quad (3)$$

where $*$ is 1D convolution and f_σ consists of two 1×1 Conv1D layers with LeakyReLU activation. The resulting $\hat{\mu}$ and $\hat{\sigma}$ parameterise a Gaussian posterior from which z is sampled and passed to G yielding $\tilde{x}_s$.

3 Training

Stage 1 follows the representation learning and adversarial training procedure in [2] with 20,000 3-sec segments of real and synthesized drum recordings. Stage 2 guides rhythm transform learning while concurrently *re*-regulating the latent space over 6000 steps. The model is trained using Adam ($LR = 2e^{-4}$) with batch size 8; fuser networks are trained with a reduced rate of $1e^{-4}$ to mitigate timbre leakage. As the transform modifies source rhythmic–timbral layout, standard reconstruction loss (i.e., comparing $\tilde{x}_s$ to x_s) is unsuitable. To produce pseudo-targets for training reconstruction losses, Stage 2 employs a 10,000-segment synthetic dataset created from 100 kits constructed from real and electronic oneshots with pitch and gain augmentation and 100 rhythms across various styles. Temporal expressivity is added via tempo scaling, swing, microtiming, and event-timing dropouts. Kits are stored per-segment with pitch and gain settings, enabling re-sequenced reconstruction loss pseudo-targets by applying augmented oneshots used in assembling x_s to drum class events and timing determined by x_t. Cycle consistency and attention entropy losses respectively promote invertibility and head diversity during training. Following Stage 2, the trained system performs transformations directly on real drum recordings.

4 Examples and Summary

Examples of the transformation are presented on the supporting website.[1] System output sounds coherent, with the rhythm of the target applied as intended to the source timbres. In cases where the timbre is not convincingly achieved in the generations, traces of the source are heard with reduced high frequency detail; however, improved fidelity is expected through further experimentation with loss balancing. Future work will explore an interactive sequencing interface to allow direct control over target rhythm events within x_s.

References

[1] López-Serrano, P., Davies, M. E., Hockman, J., Dittmar, C., and Muller, M., "Break-informed audio decomposition for interactive redrumming," in *ISMIR*, 2018.

[2] Caillon, A. and Esling, P., "RAVE: A variational autoencoder for fast and high-quality neural audio synthesis," arXiv:2111.05011, 2021.

[3] Zehren, M., Alunno, M., and Bientinesi, P., "High-Quality and Reproducible Automatic Drum Transcription from Crowdsourced Data," *Signals*, 4(4), 2023.

[1] https://jhockman.github.io/

Audio Engineering Society

Late Breaking Demo Paper

Presented at the AES International Conference on
Artificial Intelligence and Machine Learning for Audio
2025 September 8–10, London, UK

This Late Breaking Demo Paper was selected after a minimal screening process and was not peer reviewed. This Paper has been reproduced from the author's advance manuscript without editing, corrections, or consideration by the Review Board. The AES takes no responsibility for the contents. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Audio Engineering Society.

FLARE: An Open-Source Library for RIR Synthesis and Analysis in PyTorch

Gloria Dal Santo[1], Karolina Prawda[2], Sebastian J. Schlecht[3], and Vesa Välimäki[1]

[1]*Acoustics Lab, Dpt. of Information and Communications Engineering, Aalto University, Espoo, Finland*
[2]*AudioLab, School of Physics, Engineering, and Technology, University of York, York, United Kingdom*
[3]*Multimedia Communications and Signal Processing, Friedrich-Alexander-Universität (FAU), Erlangen, Germany*

Correspondence should be addressed to Gloria Dal Santo (gloria.dalsanto@aalto.fi)

ABSTRACT

We present FLARE, a PyTorch-based library for synthesizing and analyzing room impulse responses (RIRs) using feedback delay networks (FDNs). It provides a range of configurable FDN designs with parameters that can be inferred from reference data or sampled from a domain-aware distribution. FLARE is designed to support deep learning workflows by facilitating the efficient generation of diverse, metadata-rich synthetic RIR datasets.

1 Introduction

The FDN is an efficient recursive structure for synthesizing RIR [1]. In its simplest form, the FDN consists of N delay lines $\boldsymbol{m}$, a set of input and output gains $\boldsymbol{b}$ and $\boldsymbol{c}$, and a feedback matrix through which the delay outputs are coupled to the delay inputs. The transfer function of the FDN is

$$H(z) = \boldsymbol{c}^\top \left[\boldsymbol{D_m}(z)^{-1} - \boldsymbol{A}\right]^{-1} \boldsymbol{b} + d \qquad (1)$$

where $\boldsymbol{A}$ is the $N \times N$ feedback matrix, $\boldsymbol{D_m}(z)$ is the diagonal delay matrix, and d is the direct gain. The operator $(\cdot)^\top$ denotes the transpose. A common design choice is to start with a lossless FDN prototype with a smooth response. This can be achieved by using an orthogonal matrix $\boldsymbol{A}$ [2] optimized for spectral flatness and temporal density [3, 4]. Attenuation filters can be combined with the feedback matrix to control frequency-dependent T_{60} [5], designed to give the same decay characteristics to all delays, and the echo density can be increased by introducing delays in the feedback matrix [6]. We denote the resulting filter matrix with $\boldsymbol{A}(z) = \boldsymbol{U}(z)\boldsymbol{\Gamma}(z)$ where $\boldsymbol{U}(z)$ is a paraunitary FIR filter matrix, and $\boldsymbol{\Gamma}(z)$ is the diagonal attenuation matrix.

FLAMO is a Python library for trainable differentiable digital signal processing (DDSP) modules based on frequency sampling and the PyTorch framework for automatic differentiation [7]. The library provides all the DDSP components, signal paths, and operations needed to implement differentiable FDNs.

Contribution — This paper introduces FLARE[1], an open-source PyTorch-based package for the synthesis

and analysis of RIRs. The library implements various classes of FDNs derived from FLAMO's modules, with flexible parameterization options based on the user's input or sampled randomly from a domain-aware distribution. FLARE supports deep learning workflows by enabling efficient data generation for applications such as dereverberation, style transfer, and blind estimation. These tasks often require large volumes of RIR data, both recorded and synthetic. This work contributes to the field by expanding available resources through the generation of synthetic RIRs with detailed acoustic metadata.

2 Feedback Delay Network Designs

The reverb module in FLARE offers several common FDN designs listed below. The scalar coefficients of the FDNs can be optimized to reduce coloration, which is prominent for low N, and improve temporal density using the method in [4]. The attenuation $\Gamma(z)$ is designed from provided or randomly generated $T_{60}(\omega)$, and early reflections can be inserted by using an FIR filter in the direct path, in place of d. Each RIR is saved alongside the FDN parameters used to generate it and the derived acoustic parameters.

Scalar FM — The simplest form of the FDN follows (1), where $\boldsymbol{U}$ is a scalar orthogonal matrix. Common scalar orthogonal matrices include: random orthogonal, Hadamard, Householder, and rotation.

FIR FM — To increase echo density, $\boldsymbol{U}$ can be generalized as a FIR filter matrix $\boldsymbol{U}(z) = U_0 + z^{-1}U_1 + z^{-2}U_2 + \cdots + z^{-L}U_L$ with adjustable density [6]. The filters can be designed either as velvet noise sequences or as random sequences with arbitrary density.

Grouped FDN — To model coupled room spaces, $\boldsymbol{U}$ can be designed as a block matrix in which the diagonal sub-matrices control the mixing in the individual rooms, whereas the off-diagonal sub-matrices control the coupling between the rooms via a coupling coefficient [8]. The decay characteristics are controlled by groups of delay lines to model multi-stage decay [8].

3 Acoustic Parameters

The library includes a module for estimating acoustic parameters of RIRs, such as the energy decay curve, energy decay relief, normalized echo density profile, reverberation time (estimated via linear regression), clarity index, and definition. These are among the most commonly used acoustic parameters and can be computed directly from the generated RIRs, providing easy access when constructing a dataset with corresponding metadata.

4 Distribution

The source code is available online[1] and on the Python package index[2]. We plan to continuously expand the FLARE library to support more FDN designs.

References

[1] Jot, J.-M. and Chaigne, A., "Digital delay networks for designing artificial reverberators," in *AES Convention 90*, 1991.

[2] Stautner, J. and Puckette, M., "Designing multichannel reverberators," *Computer Music Journal*, 6(1), pp. 52–65, 1982.

[3] Dal Santo, G., Prawda, K., Schlecht, S. J., and Välimäki, V., "Differentiable Feedback Delay Network For Colorless Reverberation," in *Proc. of DAFx*, pp. 244–251, Copenhagen, Denmark, 2023.

[4] Dal Santo, G., Prawda, K., Schlecht, S. J., and Välimäki, V., "Optimizing tiny colorless feedback delay networks," *EURASIP Journal on ASMP*, 2025(1), p. 13, 2025.

[5] Välimäki, V., Prawda, K., and Schlecht, S., "Two-Stage Attenuation Filter for Artificial Reverberation," *IEEE Signal Process. Lett.*, 31, pp. 391–395, 2024.

[6] Schlecht, S. J. and Habets, E. A., "Scattering in feedback delay networks," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, 28, pp. 1915–1924, 2020.

[7] Dal Santo, G., De Bortoli, G. M., Prawda, K., Schlecht, S. J., and Välimäki, V., "FLAMO: An Open-Source Library for Frequency-Domain Differentiable Audio Processing," in *Proc. of ICASSP*, pp. 1–5, IEEE, 2025.

[8] Das, O. and Abel, J. S., "Grouped feedback delay networks for modeling of coupled spaces," *Journal of the AES*, 69(7/8), pp. 486–496, 2021.

[1] https://github.com/gdalsanto/flare
[2] https://pypi.org/project/flareverb

Audio Engineering Society

Late Breaking Demo Paper

Presented at the AES International Conference on
Artificial Intelligence and Machine Learning for Audio
2025 September 8–10, London, UK

This Late Breaking Demo Paper was selected after a minimal screening process and was not peer reviewed. This Paper has been reproduced from the author's advance manuscript without editing, corrections, or consideration by the Review Board. The AES takes no responsibility for the contents. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Audio Engineering Society.

Towards Leveraging Machine Learning for Multichannel Audio Upmix

Jean-Marc Jot[1], Gordon Wichern[2], Thaddeus Paez[3], and Sunil Bharitkar[4]

[1]*Virtuel Works LLC, USA*
[2]*Mitsubishi Electric Research Laboratories (MERL), USA*
[3]*Samsung Research Tijuana, Samsung Electronics, Mexico*
[4]*Digital Media Solutions Group-Audio Lab, Samsung Research America, CA, USA*

Correspondence should be addressed to Jean-Marc Jot (jmmjot@gmail.com)

ABSTRACT

The demand for the production, distribution, and playback of audio content in surround or immersive multichannel formats and flexible playback setups has continued to grow in the cinema, broadcast, and music industries. This trend motivates the continued development of signal processing methods for the conversion of recordings between different spatial audio formats. Leveraging Machine Learning (ML) techniques presents opportunities to improve algorithm performance: (a) adaptation to content type (such as treating speech-dialog, effects, and music differently), and (b) spatial signal decomposition and analysis (including primary-ambient decomposition). This paper seeks more particularly to encourage future research tackling the problem of ML-based primary-ambient decomposition and spatial analysis of two-channel recordings, within a modular framework, referred to as *Acoustic Source Decomposition* (ASD), that leverages ML-based audio signal processing methods and approaches developed for audio source separation, noise reduction, or dereverberation.

1 Towards ML-Based Upmix of Audio Signals

ML-based spatial format conversion of audio signals is generally introduced in [1] (section 3.4). In the present paper, we focus more particularly on *upmixing*, the conversion of mono or stereo audio content to address consumer systems, venues or distribution formats comprising a higher channel count.

Modern audio upmixing methods commonly employ frequency-domain primary-ambient decomposition (PAD) of the source signal, so that ambient signal portions may be rendered selectively to reproduce a controlled spatial diffuseness in the target multichannel format. Primary components, on the other hand, call for rendering strategies that maximize spatial discrimination and reproduce the apparent direction of arrival of each time-frequency component [2, 3].

In practical applications, like TV with mixed-source streaming services, preserving artistic intent may require that upmix algorithm behavior be dependent on content type. Accordingly, in [4], a ML-based input classifier is inserted to identify source content category (movie, music, advertisement, news, sports, etc.).

The development of ML-based spatial audio decomposition and analysis methods to improve the performance of upmix algorithms is hindered by infrastructure and psychoacoustic modeling limitations:
– training such algorithms may involve the construction of datasets comprising audio recordings stored in several possible multichannel formats, thus placing extra requirements on storage and computation capacity;
– the subjective attributes that underlie spatial audio perception are not readily encompassed in a differentiable loss function suitable for training supervised or unsupervised ML models.

2 Acoustic Source Decomposition (ASD)

In order to overcome these limitations, we propose dividing the problem into signal analysis and decomposition sub-tasks for which algorithm design and optimization metrics can be defined with focused scope. A ML-based realization of this approach, which we name *Acoustic Source Decomposition* (Figure 1), involves minimizing cost functions associated with three successive operations, independently or jointly:

(a) Source Decomposition. In this first stage, the two-channel input signal is decomposed into a mixture of two-channel sources (or stems). This operation can readily leverage existing ML-based source separation methods (e.g., [5]), and may be skipped if the source content is provided in separated stem-based form.

(b) Acoustic Decomposition. Each source (or stem) is decomposed into direct and reflected components, the latter possibly decomposed into early and late reflections. This task can employ variants of ML-based methods developed independently for noise reduction and dereverberation of audio signals [6].

(c) Spatial Analysis evaluates spatial properties of each acoustic component of each source in the time-frequency domain, which inform re-rendering in the target multichannel audio format [3]. For a direct component, these properties may include estimated direction of arrival. For a spatially diffuse reflected component, they may represent directional emphasis.

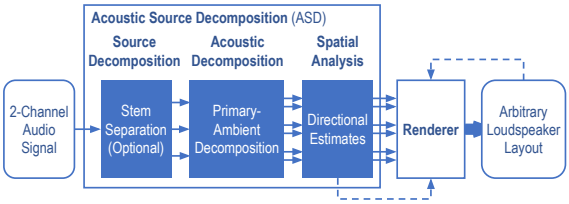


Fig. 1: Acoustic Source Decomposition (ASD) of a two-channel audio signal.

Acoustic Source Decomposition (ASD) offers a methodological framework that can facilitate the development of novel ML-based audio upmix methods, by incorporating tractable sub-tasks synergistic with independent prior and ongoing data-based or model-based audio processing algorithm innovation. Furthermore, the Source Decomposition stage (a) can advantageously incorporate content-dependent upmix processing.

References

[1] Steinmetz, C., Uhle, C., Everardo, F., Mitcheltree, C., McElveen, J. K., Jot, J.-M., and Wichern, G., "Audio Signal Processing in the Artificial Intelligence Era: Challenges and Directions," *J. Audio Eng. Soc.*, 73, 2025.

[2] Ibrahim, K. M. and Allam, M., "Primary-Ambient Source Separation for Upmixing to Surround Sound Systems," in *Proceedings of IEEE Int. Conf. Acou., Speech and Sig. Proc. (ICASSP)*, 2018.

[3] Goodwin, M. and Jot, J.-M., "Spatial Audio Scene Coding," in *Proceedings of 125th Convention Audio Eng. Soc.*, 2008.

[4] Bharitkar, S. and Paez, T., "Edge-Optimized Model for Multimedia Classification using Linguistic Metadata," in *Proceedings of IEEE Int. Conf. Acou., Speech and Sig. Proc. (ICASSP)*, 2024.

[5] Luo, Y. and Yu, J., "Music Source Separation With Band-Split RNN," *IEEE/ACM Trans. Audio, Speech, and Lang. Proc.*, 31, 2023.

[6] Wang, Z.-Q., Wichern, G., and Le Roux, J., "Convolutive prediction for monaural speech dereverberation and noisy-reverberant speaker separation," *IEEE/ACM Trans. on Audio, Speech, and Lang. Proc.*, 29, 2021.

Audio Engineering Society

Late Breaking Demo Paper

Presented at the AES International Conference on
Artificial Intelligence and Machine Learning for Audio
2025 September 8–10, London, UK

This Late Breaking Demo Paper was selected after a minimal screening process and was not peer reviewed. This Paper has been reproduced from the author's advance manuscript without editing, corrections, or consideration by the Review Board. The AES takes no responsibility for the contents. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Audio Engineering Society.

Deep Localization Refinement for Bat Abundance Estimation

Roberto Alessandri[1, 2, 3], Lia Gilmour[3], Jin Jack Tan[2], Alejandro Osses[2], and Dan Stowell[1, 4]

[1]*Tilburg School of Humanities and Digital Sciences, Tilburg University Warandelaan 2, 5037 AB Tilburg, The Netherlands*
[2]*Sorama B.V., Achtseweg Zuid 153H, 5651 GW Eindhoven, The Netherlands*
[3]*Bat Conservation Trust (BCT), 8 Battersea Park Road, London, SW8 4BG, United Kingdom*
[4]*Naturalis Biodiversity Center (NBC), Darwinweg 2, 2333 CR Leiden, The Netherlands*

Correspondence should be addressed to Roberto Alessandri (roberto.alessandri@sorama.eu)

ABSTRACT

Monitoring bat populations through acoustic sensing is vital for biodiversity and conservation research. However, most current approaches rely on single-channel audio, lacking the spatial resolution needed to separate overlapping calls or track multiple animals. We propose a prototype that effectively combines deep learning with spatial signal processing. First detection is performed in time-frequency. Detected events are passed through the beamformer to produce preliminary spatial heatmaps before using a U-Net, which refines the maps by producing sharper localisation and suppressing noise. The network is conditioned on species identity via feature embeddings extracted from the detection model. Training supervision is provided by synthetic ground truth heatmaps, generated from beamforming maxima and encoded as one-hot-like coordinate maps. The model is trained and overfitted on early field recordings captured using a 64-channel microphone array with a Sunflower configuration. The system outputs clean spatial heatmaps that are compatible with downstream tracking and abundance estimation. While preliminary and overfitted, the demo shows the capacity to improve localisation sharpness and species separation. Designed for real-time and edge deployment, this work demonstrates how acoustic sensing and deep neural inference can be combined to improve spatial reasoning in ecological monitoring applications.

1 Introduction

Monitoring bat species abundance is vital for biodiversity assessment [1]. Traditional acoustic approaches often rely on hand-crafted signal descriptors and single-channel recordings, which lack spatial resolution and struggle in noisy environments [2]. Advances in Deep Learning (DL) for bat echolocation call detection, such as BatDetect2 [3], have improved classification performance, yet spatial localisation remains a key challenge. This project introduces an integrated approach that combines automatic detection, array-based localisation, and data-driven refinement. Echolocation events are first identified, then spatialised using time-and-frequency-selective Delay-and-Sum (DAS) beamforming, followed by refinement via a U-Net model [4]. The

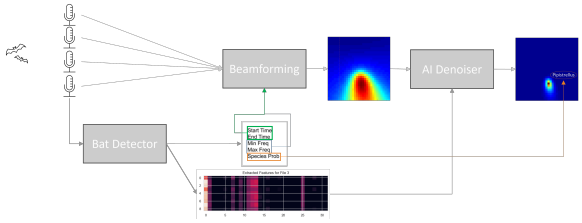


Fig. 1: Schematic representation of the training procedure.

resulting heatmaps are designed for use in multi-bat tracking and species-aware abundance estimation.This pipeline addresses a central limitation in current methods: the difficulty of resolving multiple sources under realistic conditions.

2 Methods

Our system is composed of the following steps, as shown in Fig. 1:

- Data collection using the Sorama 64-microphone array CAMiV64s
- Echolocation call detection via BatDetect2
- Optimized (DAS) beamforming
- Per-frame heatmap refinement using a U-Net
- (Future) Multi-target particle filtering for tracking

Ground Truth Heatmaps are constructed from the maxima of initial DAS beamformed maps, where each map represents a frame of $0.017s$. The (x,y) coordinates of the energy peak are converted to one-hot-encoded spatial targets. During training, species-specific feature embeddings from BatDetect2's decoder are injected at the U-Net bottleneck, enabling the model to filter out blobs not associated with echolocating bats. This allows the model to condition its refinement on species, helping it learn to denoise and enhance beamformed maps while suppressing irrelevant sources.

3 Preliminary Results

Due to limited field data, we trained the U-Net by overfitting it to a small set of real recordings. Qualitative visualisations suggest sharper localisation. While not yet suitable for generalisation, the prototype confirms that learning-based refinement can meaningfully enhance spatial output quality.

4 Discussion and Future Work

This demo highlights the feasibility of combining time-and-frequency-selective beamforming with deep learning for wildlife localisation. Ongoing work focuses on expanding real-world datasets, implementing tracking, and optimising the system for edge deployment. A current limitation is that training maps are derived assuming at most one bat per frame. In practice, the high frame rate means calls from different individuals are usually separated across consecutive frames, so multi-target events are partly addressed. Future work will explore synthetic superpositions of single-source examples to assess robustness when multiple calls overlap, and investigate robustness in high-noise environments.

Acknowledgements

Roberto Alessandri is funded by the BioAcousticAI Doctoral Network (European Union's Marie Skłodowska-Curie Action under grant agreement No 101116715). We are also grateful to the Bat Conservation Trust for their collaboration and assistance in field data collection, and to the developers of BatDetect2, whose work formed the foundation of our detection module.

References

[1] Lang, L. et al., "Autonomous Bat Echolocation Monitoring System with Machine Learning for Real-Time Pest and Biodiversity Management," in *2024 International Conference on Decision Aid Sciences and Applications (DASA)*, pp. 1–5, Manama, Bahrain, 2024, doi:10.1109/DASA63652.2024.10836283.

[2] Mac Aodha, O. et al., "Bat Detective: Deep learning tools for bat acoustic signal detection," *PLoS Computational Biology*, 14(3), p. e1005995, 2018, doi:10.1371/journal.pcbi.1005995.

[3] Mac Aodha, O. et al., "Towards a General Approach for Bat Echolocation Detection and Classification," *bioRxiv*, 2022, doi:/10.1101/2022.12.14.520490.

[4] Ronneberger, O., Fischer, P., and Brox, T., "U-Net: Convolutional Networks for Biomedical Image Segmentation," *arXiv preprint arXiv:1505.04597*, 2015, doi:10.48550/arXiv.1505.04597.

Audio Engineering Society

Late Breaking Demo Paper

Presented at the AES International Conference on
Artificial Intelligence and Machine Learning for Audio
2025 September 8–10, London, UK

This Late Breaking Demo Paper was selected after a minimal screening process and was not peer reviewed. This Paper has been reproduced from the author's advance manuscript without editing, corrections, or consideration by the Review Board. The AES takes no responsibility for the contents. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Audio Engineering Society.

Denoising Marine Recordings of Humpback Whales

Toby Gifford[1]

[1] *University of the Sunshine Coast*

Correspondence should be addressed to Toby Gifford (tgifford@usc.edu.au)

ABSTRACT

The research presents a prototype denoising algorithm intended to extract humpback whale vocalisations from noisy marine recordings at a specific physical location. It leverages a pre-trained humpback whale vocalisation classification model for fine-tuning a CNN audio denoiser. The acoustic environment of interest has a rich soundscape including snapping shrimp and frequent engine noise. Informal perceptual evaluation suggests the denoiser is successful at removing other marine sounds, compared to a bandpass filter, although introduces artefacts.

1 Introduction

Humpback whale vocalisations have an enduring popular appeal. Researchers from our *Creative Ecologies Research Centre* at the University of the Sunshine Coast, Australia, have used live hydrophone feeds of humpback whale vocalisations to create immersive audio-visual artworks designed to engage the local community with their marine ecosystem [1]. In partnership with state government, shire council, and a local whale-watching tourism operator, we have embarked on a project to install a permanent live streaming hydrophone in the ocean off the coast of Mooloolabah, Queensland. The live stream will be available on the internet, and also will be played in the tourism operator's storefront/dock at the marina.

This location is on a humpback whale migration route, and from June - October each year whale-watching is a popular tourist activity. During whale season, humpback vocalisations can often be heard through a hydrophone in the ocean. The permanent hydrophone stream will be deployed near a reef. The acoustic environment in this location contains a rich soundscape, often dominated by the sound of crustaceans, along with fish sonifications, dolphin calls, and frequent engine noise from motor boats and jet-skis.

At times the vocalisations of humpback whales can be obscured by the richness of this marine soundscape. Whilst it is interesting and educational to hear the acoustic environment as it is, at times it may be desirable to extract (or at least emphasise) whale calls from this background soundscape.

This paper presents a prototype humpback whale 'denoiser', intended specifically to extract humpback whale calls from the typical acoustic environment of this marine location.

2 Denoising Algorithm

The approach taken in creating this denoising algorithm was to leverage an existing humpback whale classification model from NOAA & Google [2]. The idea of this denoiser was to use the ability of this model to detect 'whaliness' to guide the training of a CNN audio denoiser to preserve whale sounds whilst removing other sounds.

The denoiser was implemented as a lightweight convolutional neural network trained to suppress non-whale components from underwater recordings while preserving humpback whale vocalisations. The system was trained in three stages (see Fig. 1):

1. Denoiser pre-training. Segments of audio with relatively clean whale calls were band-pass filtered (200–800 Hz), and mixed with synthetic noise at random SNRs (10–25 dB), then used to train the denoiser network with mean squared error (MSE) against the clean target. All clips were 3.92 s at 10 kHz, matching the input format of the NOAA & Google model.

2. ScoreNet training. A surrogate loss network was trained to predict the scalar 'whale score' of the NOAA & Google classifier from raw audio, providing a differentiable measure of 'whaliness' for fine-tuning. While the original classifier is differentiable in principle, the model is provided for TensorFlow, whilst our denoising pipeline is created in PyTorch, making fine-tuning the denoiser with the original model awkward.

3. Denoiser Fine-tuning. The denoiser was fine-tuned with a loss that combines a negative whale score and a small MSE regularizer against the noisy input (to maintain stability):

$$L = -\lambda_{score} \cdot ScoreNet(\hat{x}) + \lambda_{mse} \cdot \mathrm{MSE}(\hat{x}, x_{noisy})$$

where λ_{score} weights the whale-awareness term, encouraging preservation of classifier-salient features.

At inference, audio was band-pass filtered, denoised, and post-processed to remove DC offset, lightly high-pass (10 Hz), and peak-normalised.

3 Results

Around 1000 segments of noisy audio containing quiet or distant whale calls were denoised using this algorithm. These examples can be heard at https://whales.offig.com, where each is presented against the original noisy segment, and a bandpassed version of that segment for comparison.

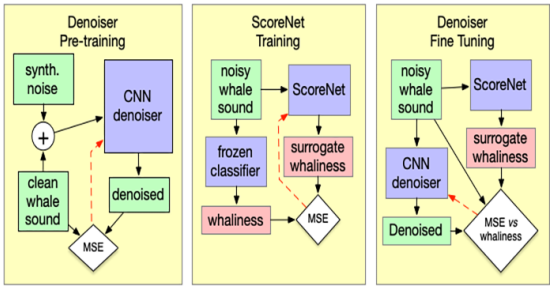


Fig. 1: Overview of the denoising architecture.

Our informal perceptual evaluation suggests this algorithm does subtract non-whale sounds from the audio, more than a simple bandpass filter, at the expense of introducing some other audio artefacts. The structured sounds of the marine ecosystem are replaced with a whiter background noise, whilst leaving the whale sounds relatively intact.

4 Discussion

This research has created a prototype humpback whale denoising algorithm. The intended use case is for isolating whale vocalisations from a noisy marine recording from a known specific geographical location. It has trained a bespoke denoiser using clean and noisy recordings from this location, that also leverages an existing humpback whale audio classification model.

References

[1] Barclay, L., King, T., Gifford, T., Williams, M., Grimm, M., Fuller, J., Wegener, F., Beaumont, G., and Burgess, J., "Creative Ecologies Research Cluster: A model for place-based creative practice, community engagement and reciprocity in Australia," in *Proceedings of the 30th International Symposium on Electronic Arts*, pp. 985–988, Seoul, South Korea, 2025.

[2] Allen, A. N., Harvey, M., Harrell, L., Jansen, A., Merkens, K. P., Wall, C. C., Cattiau, J., and Oleson, E. M., "A Convolutional Neural Network for Automated Detection of Humpback Whale Song in a Diverse, Long-Term Passive Acoustic Dataset," *Frontiers in Marine Science*, 8, p. 607321, 2021, ISSN 2296-7745, doi:10.3389/fmars.2021.607321.

Audio Engineering Society

Late Breaking Demo Paper

Presented at the AES International Conference on
Artificial Intelligence and Machine Learning for Audio
2025 September 8–10, London, UK

This Late Breaking Demo Paper was selected after a minimal screening process and was not peer reviewed. This Paper has been reproduced from the author's advance manuscript without editing, corrections, or consideration by the Review Board. The AES takes no responsibility for the contents. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Audio Engineering Society.

MULTIVOX: Spatial & Multimodal Data of Singing Groups

Gerardo Meza Gómez[1,2], Mariana Sepúlveda[1,2], Adrian Roman[3], Rodrigo Sigal Sefchovich[1], and Iran Roman[4]

[1]*Universidad Nacional Autónoma de México*
[2]*Universidad Autónoma de Tamaulipas*
[3]*New York University*
[4]*Queen Mary University of London*

Correspondence should be addressed to Gerardo Meza (gmeza@uat.edu.mx)

ABSTRACT

We introduce MULTIVOX, a spatialized audio-visual dataset of 154 a cappella choral performances recorded in acoustically distinct environments. MULTIVOX captures 360 video, Ambisonics, stereo, & near-field microphones with structured annotations of performer locations and characteristics. MULTIVOX enables research in music informatics, sound localization, ensemble synchronization, and performer-specific modeling in real-world contexts.

1 Introduction

We present MULTIVOX[1], a dataset advancing machine perception of musical ensembles. Choral performance involves complex performer-space interactions [1], challenging voice separation and spatial localization [2]. Prior multimodal work targets speech/acoustic scenes [3], treating music peripherally [4]. MULTIVOX contains 154 a cappella performances with synchronized 360° video, Ambisonics, stereo, and close-field audio. Spatial annotations and varied configurations enable real-world research.

2 Methods

Singer recruitment. MULTIVOX features 35 undergrad music students (ages 19–27, mean 21) from the Universidad Autónoma de Tamaulipas, all with choral singing experience. Eleven participants identified as female and twenty-four as male. Participants wore black masks leaving only mouth and nostrils visible.

Fig. 1: (1) Mask worn to conceal identity. Example choirs with 6 (panel 2) and 16 (panel 3) singers.

Recording equipment. Far-field equipment included an Insta360 X3 camera, Zoom H3-VR Handy Recorder, and RØDE M5 microphones connected to a Zoom

[1]dataset website: https://multi-vox.github.io

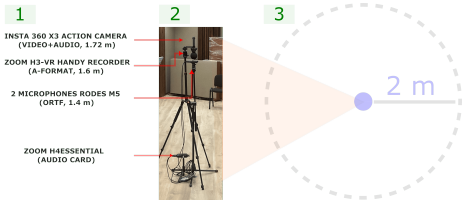


Fig. 2: (1) and (2) the far-field recording devices and their relative height positioning; (3) equipment distance relative to the circle of singers.

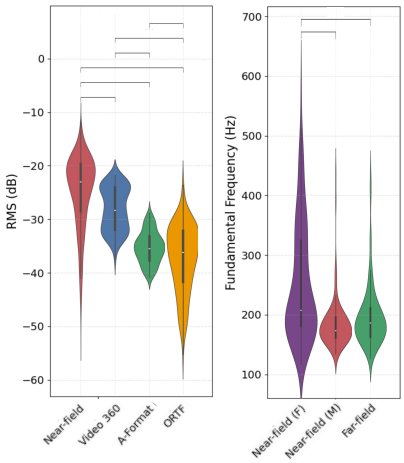


Fig. 3: Left: RMS distributions by recording modality; near-field recordings show significantly higher RMS. Right: F0 comparison; female near-field recordings have higher fundamental frequencies than male near-field and all far-field.

H4essential audio card, all mounted on a tripod. Each participant used their personal smartphone or tablet as a near-field microphone for individual voice capture.

Recording protocol. Ten sessions were conducted: five in an auditorium and five in a recording studio. Each session included four iterations of assigned repertoire. Singer positions were pre-marked, and a logbook documented each participant's name, position, gender, and height. The four iterations followed this protocol: 1) Full choir (but some singers silent). 2) Only male singers; females remain silent. 3) Only female singers; males remain silent. 4) Singers face to an assigned direction; some remain silent. **Session details.** Studio sessions used six-singer choirs. Auditorium sessions had either 6 or 16 singers (four sessions with 16, one with 6). Silent participants stayed in place. **Data processing.** Videos were standardized to 1960× 980 pixels. Audio resampled to be 44.1 kHz, 24-bit depth. Temporal alignment across modalities used cross-correlation followed by manual verification and automatic segmentation. **Annotations.** Each performance comes with structured annotations including song title, key, recording space, participant spatial positions, facing directions, demographics, and singing activity status.

3 Analysis of Multimodal Features

Analysis revealed key dataset characteristics. Singer positions are uniformly distributed around the 360 recording setup, with peaks at 0 and 180. Height distribution shows gender-based patterns: females predominantly 150-169cm, males above 170cm. Near-field recordings capture significantly higher RMS amplitudes than far-field microphones due to proximity. Female near-field devices record higher F0 values than other modalities. Synchronization analysis demonstrates the dataset's unique value in capturing both individual vocal events and ensemble coordination. Natural micro-timing differences between singers are observed in near-field recordings. Their onsets overall align closely with far-field energy peaks, enabling detailed modeling of group vocal performance dynamics.

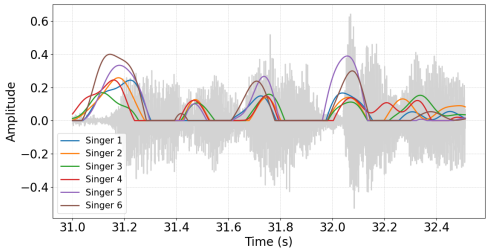


Fig. 4: Overlay of individual singer onsets (colored lines) from the rectified derivative of near-field envelopes. Gray waveform shows far-field reference audio. Lines reveal micro-timing differences in vocal attacks.

References

[1] Fadil, L. et al., "Employing Chroma-gram techniques for audio source separation in human-computer interaction," in *AIP Conference*, 2025.

[2] Shimada, K. et al., "Multi-ACCDOA: Localizing and detecting overlapping sounds from the same class with ADPIT," in *ICASSP*, 2022.

[3] Shimada, K. et al., "Stereo Sound Event Localization and Detection with Onscreen/offscreen Classification," *DCASE*, 2025.

[4] Roman, A. et al., "Generating Diverse Audio Visual 360 Soundscapes for SELD," *arXiv*, 2025.

Audio Engineering Society

Late Breaking Demo Paper

Presented at the AES International Conference on
Artificial Intelligence and Machine Learning for Audio
2025 September 8–10, London, UK

This Late Breaking Demo Paper was selected after a minimal screening process and was not peer reviewed. This Paper has been reproduced from the author's advance manuscript without editing, corrections, or consideration by the Review Board. The AES takes no responsibility for the contents. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Audio Engineering Society.

Towards Intelligent Music Education: Score-Informed Transcription and Performance Assessment

Jackson Loth[1], Aikaterini Maria Primenta[1], Jingjing Tang[1], Xavier Riley[1], Simon Dixon[1], and Emmanouil Benetos[1]

[1]*Centre for Digital Music, Queen Mary University of London*

Correspondence should be addressed to Jackson Loth (j.j.loth@qmul.ac.uk)

ABSTRACT

We present work in progress on a score-informed automatic music transcription and performance assessment system designed to support automated music education. The system comprises an automatic guitar transcription model trained using a combination of public and internal data, along with pre-trained transcription models for piano and violin. After transcription, the system evaluates performances by aligning the transcribed results with music scores to identify various errors in pitch, rhythm, tempo, and other aspects of the performance.

1 Introduction

In recent years, advances in artificial intelligence have significantly transformed the field of music education, introducing innovative tools to improve the learning experience for both teachers and students. AI-assisted music education systems blend traditional teaching approaches with intelligent technologies capable of analyzing, evaluating, and responding to musical performances in real time.

These systems are composed of several key components, including automatic transcription, performance assessment, and personalized feedback. Together, these elements make music practice more interactive and adaptive. As a result, teachers can provide more personalized instruction based on students' progress and specific needs, while students benefit from immediate feedback during their practice sessions.

Our goal is to develop a system that provides students with automated feedback on their instrumental performances. Focusing on piano, violin, and guitar, the system compares an audio recording of a student with the original score by aligning these two, and then identifies and reports mistakes made during the performance.

Our approach consists of three key components:**(1)** automatically transcribing guitar, piano, and violin recordings into MIDI format; **(2)** automatically aligning the ground truth score with the MIDI transcription of the student's performance; and **(3)** detecting performance errors based on the alignments produced in step 2.

2 Methodology

2.1 Transcription

Our transcription work initially focuses on guitar. We leverage a private internal dataset of 2.8 hours of fully annotated recordings produced by Algorivm[1] for our training data. Our final dataset is a combination of this internal data and GuitarSet[1] for a total of 5.8 hours of training data. To ensure accurate note timings in our training data, we create a fine-alignment of MIDI notes to note onsets in our dataset [2]. Our transcription model is first trained on the MAESTRO dataset [3] and then fine-tuned on the mixed dataset described above. This model is evaluated on the test split of GuitarSet and our internal dataset, achieving precision, recall and f1 scores of 0.8895, 0.9067 and 0.8973 respectively at the frame level and 0.8903, 0.8926 and 0.8900 at the note level.

2.2 Performance Assessment

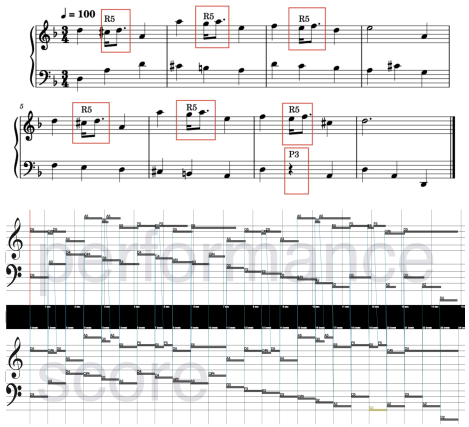


Fig. 1: Alignment of a transcribed piano performance in MIDI format with the original score using Parangonada [4], highlighting pitch and rhythmic errors in the performance.

We apply Parangonada [4] to align the transcribed MIDI files with the original music scores, as illustrated in Figure 1. This approach provides note-wise alignments between the score and performance, which serve as the basis for subsequent analysis. Our error detection in musical performances focuses on five main categories: pitch, rhythm, tempo, structural, and intonation errors, each further divided into specific subcategories. Pitch errors, such as missing and extra notes, are directly identified from the alignment results. Rhythmic and tempo-related errors are detected through both local and global analysis on note values and tempo estimations. We will extend the system to detect errors such as structural deviations and intonation error, and on classifying detected errors in a more formative manner.

[1] https://algorivm.com/

3 Conclusion

We have presented progress on score informed transcription for the purposes of improving musical education. Future work will focus on improving transcription accuracy using synthesized data and developing a more sophisticated assessment pipeline. We hope that this demo helps inspire further work in intelligent musical education systems.

4 Acknowledgments

This work is supported by Innovate UK (Project no. 10105279), the UKRI Centre for Doctoral Training in Artificial Intelligence and Music (Grant no. EP/S022694/1), and China Scholarship Council (grant no. 202008440382).

References

[1] Xi, Q., Bittner, R. M., Pauwels, J., Ye, X., and Bello, J. P., "GuitarSet: A Dataset for Guitar Transcription," in *19th ISMIR Conference*, pp. 453–460, 2018.

[2] Riley, J., Guo, Z., Edwards, D., and Dixon, S., "GAPS: A Large and Diverse Classical Guitar Dataset and Benchmark Transcription Model," in *25th ISMIR Conference*, 2024.

[3] Hawthorne, C., Stasyuk, A., Roberts, A., Simon, I., Huang, C.-Z. A., Dieleman, S., Elsen, E., Engel, J., and Eck, D., "Enabling Factorized Piano Music Modeling and Generation with the MAESTRO Dataset," in *ICLR*, 2019.

[4] Peter, S. D., Cancino-Chacón, C. E., Foscarin, F., McLeod, A. P., Henkel, F., Karystinaios, E., and Widmer, G., "Automatic Note-Level Score-to-Performance Alignments in the ASAP Dataset," *Transactions of the International Society for Music Information Retrieval*, 6(1), pp. 27–42, 2023, doi:10.5334/tismir.149.

Audio Engineering Society

Late Breaking Demo Paper

Presented at the AES International Conference on
Artificial Intelligence and Machine Learning for Audio
2025 September 8–10, London, UK

This Late Breaking Demo Paper was selected after a minimal screening process and was not peer reviewed. This Paper has been reproduced from the author's advance manuscript without editing, corrections, or consideration by the Review Board. The AES takes no responsibility for the contents. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Audio Engineering Society.

Real-time Voice Conversion with Online Speaker Adaptation for Creative Audio Processing

Anders R. Bargum[1,2], Stefania Serafin[2], and Cumhur Erkut[2]

[1]*Khora & Heka VR, Copenhagen*
[2]*Multisensory Experience Lab, Aalborg University, Copenhagen*

Correspondence should be addressed to Anders R. Bargum (arba@create.aau.dk)

ABSTRACT

Advances in speech generation and representation learning have enabled high-quality voice conversion systems, yet real-time performance and user-defined voice cloning remain underexplored. This paper introduces a real-time voice conversion model that utilizes an online speaker embedding platform to generate speaker statistics for ad hoc speaker adaptation and cloning. We provide a brief overview of the system, its real-time implementation, and demonstrate its use in creative audio and music production scenarios.

1 Introduction

Voice conversion (VC) modifies the speaker identity of a speech signal while preserving its linguistic content. Recent progress in self-supervised learning (SSL) and large language models (LLMs) has improved VC by using unified acoustic-linguistic representations instead of traditional encoders based on automatic speech recognition (ASR) and automatic speaker verification (ASV). However, SSL models often have high computational demands, limiting real-time use. Building on the framework from [1], we propose an any-to-any VC plugin optimized for real-time performance and high sampling rates (44.1 kHz), targeting applications such as digital audio workstations (DAWs). Furthermore, our system incorporates an online speaker adaptation platform that enables users to import and apply vocal characteristics of unseen target speakers. The system uses an autoencoder architecture, with explicit disentanglement of linguistics, prosody, and speaker identity, allowing for fine-grained control. A demonstration of the system can be found in the accompanying video[1].

2 Methods

We present an encoder-decoder architecture (Fig. 1) modeled after neural audio codecs such as [2], using causal, dilated, residual 1D convolutions. Content embeddings are derived from soft speech units distilled from HuBERT features discretized into 100 k-means

[1]https://vimeo.com/1111577405?fl=plfe=sh

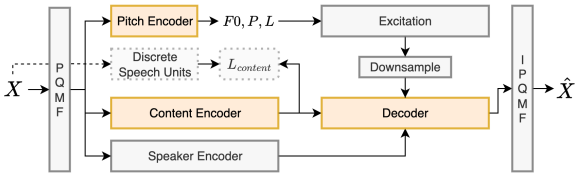


Fig. 1: Abstract global VC pipeline.

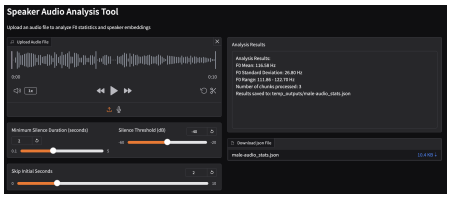


Fig. 2: Online speaker-profile generator.

clusters [3]. All 44.1 kHz input audio is first processed via Pseudo Quadrature Mirror Filters (PQMF) to reduce dimensionality and complexity. To preserve prosody, we use $F0$, periodicity, and loudness estimates to build a sinusoid-plus-noise excitation that steers the decoder.

2.1 Speaker Adaptation

To enable user-defined voice cloning, we implement an ad hoc speaker adaptation mechanism by hosting the speaker encoder online to retrieve a speaker profile, including a global embedding and articulatory statistics. The encoder processes raw waveforms, requiring only a few seconds of audio to generate a 256-dimensional speaker embedding. Unlike complex RNNs commonly used in ASV, our encoder employs a lightweight downsampling CNN architecture similar to our content encoder. The input waveform is processed by 16 PQMFs, features are then extracted through CNN blocks and aggregated over time using Multi-layer Feature Aggregation (MFA) and attentive statistics pooling. The speaker encoder is pretrained on the VoxCeleb dataset with additive margin softmax loss, as described in [4].

3 Results

The speaker embedding model achieves an equal-error-rate (EER) of **3.98**% on unseen speaker verification tasks (3.275M parameters). We train our system, except the static speaker encoder, on LibriTTS train-clean subsets, attaining a word-error-rate (WER) of **6.54**% and a character-error-rate (CER) of **2.04**% on the test-clean subset. Speaker matching is evaluated on 6 unseen VCTK speakers, with an average speaker similarity of **75.36**% measured via Rezemblyzer embeddings.

4 Implementation

We set all convolutions as causal and export the pipeline using Torschript. The Torchscript model is hosted in JUCE and exported as a VST plugin. A pre-defined speaker dictionary is stored using json allowing users to change input identity real-time. For ad hoc speaker import and voice cloning the speaker encoder is deployed on Hugging Face (see Fig. 2). Here users are allowed to upload audio from a target speaker and seamlessly import the resulting profile into the plugin.

5 Summary

This paper introduces a real-time voice conversion system that enables high-quality, user-defined voice cloning using an online speaker embedding platform. The lightweight model supports low-latency operation at 44.1 kHz and is deployable as a VST plugin for creative audio applications. It achieves strong accuracy in speaker adaptation and speech intelligibility on unseen voices. Given risks of misuse such as impersonation and privacy violations, model checkpoints will not be open sourced. Additional safeguards like misuse detection and audio watermarking are planned to ensure responsible and ethical deployment.

References

[1] Yang, Y. et al., "STREAMVC: Real-Time Low-Latency Voice Conversion," in *ICASSP 2024*, pp. 11016–11020, 2024.

[2] Zeghidour, N. et al., "Soundstream: An end-to-end neural audio codec," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30, pp. 495–507, 2021.

[3] van Niekerk et al., "A Comparison of Discrete and Soft Speech Units for Improved Voice Conversion," in *ICASSP 2022*, pp. 6562–6566, 2022.

[4] Jung, J.-w. et al., "Pushing the limits of raw waveform speaker recognition," in *Proc. Interspeech*, 2022.