Audio Engineering Society

Convention Paper 10596

Presented at the 152nd Convention
2022 May, In-Person and Online

Efficient neural networks for real-time modeling of analog dynamic range compression

Christian J. Steinmetz and Joshua D. Reiss

Centre for Digital Music, Queen Mary University of London, UK

Correspondence should be addressed to Christian J. Steinmetz (`c.j.steinmetz@qmul.ac.uk`)

ABSTRACT

Deep learning approaches have demonstrated success in modeling analog audio effects. Nevertheless, challenges remain in modeling more complex effects that involve time-varying nonlinear elements, such as dynamic range compressors. Existing neural network approaches for modeling compression either ignore the device parameters, do not attain sufficient accuracy, or otherwise require large noncausal models prohibiting real-time operation. In this work, we propose a modification to temporal convolutional networks (TCNs) enabling greater efficiency without sacrificing performance. By utilizing very sparse convolutional kernels through rapidly growing dilations, our model attains a significant receptive field using fewer layers, reducing computation. Through a detailed evaluation we demonstrate our efficient and causal approach achieves state-of-the-art performance in modeling the analog LA-2A, is capable of real-time operation on CPU, and only requires 10 minutes of training data.

1 Introduction

While a significant amount of processing in audio and music production is performed digitally, there is a rich history of analog equipment that remains in high demand for its unique sonic signature. As a result, there has been an interest in virtual analog modeling [1–5], the task of constructing digital models to emulate these analog devices. While there are a range of traditional approaches in analog modeling, there has been a growing interest in neural network approaches [6–9]. These approaches enable constructing emulations using only input-output measurements from the device, which has the potential to significantly lower the engineering effort in creating effect emulations.

Thus far, applications of neural networks for audio effect modeling have focused mostly on modeling vacuum-tube amplifiers [10–13] and distortion circuits [6, 7, 9, 14, 15]. In contrast, time-varying nonlinear effects, like dynamic range compressors [16], potentially pose a greater challenge in the modeling task due to their time-dependant nonlinearities, and have so far seen less attention. A model of the 1176N compressor was proposed [8], but it did not address the device control parameters and was evaluated only with electric guitar and bass signals. Modeling the LA-2A was addressed in [17, 18], and while their model captured the overall characteristics of the device, it exhibits artifacts, is noncausal, and not capable of real-time operation, limiting its utility in audio engineering contexts. Recently, temporal convolutional networks (TCNs) have shown success in modeling dynamic range compression [19, 20], however, these models are also noncausal and computationally expensive.

To address these limitations we propose a more efficient and causal formulation of the TCN with the aim of facilitating real-time operation on CPU. We realize that while computation across the temporal dimension can be parallelized in the TCN, computations through the depth of the network are sequential. Therefore shallower networks provide one route towards greater efficiency, yet often at the cost of smaller context window sizes, also know as the receptive field, which may limit the accuracy of the model.

Our proposed efficient TCN employs rapidly growing dilation factors, effectively enforcing very sparse convolutional kernels, which facilitates shallow networks that achieve the same receptive field as deeper networks. We carry out a range of experiments to validate our proposed architecture in the task of modeling the analog LA-2A compressor. We demonstrate that our proposed TCN architecture with fewer layers and sparse kernels performs competitively with larger noncausal formulations, producing strong results in a listening test, while also running in real-time on CPU. Additionally, we examine the role of dataset size and find that only 10 minutes of training data is required. We provide audio examples, code, and pre-trained models online[1].

2 Background

We consider an audio effect $f(x,\phi)$ that takes as input an audio signal $x \in \mathcal{X}$ and a set of P parameters $\phi \in \mathbb{R}^P$ that control the operation of the system, producing a corresponding processed version of the signal $y \in \mathcal{Y}$. In the case of an analog effect, x and y are continuous time signals. Since we aim to create a digital emulation, we utilize discrete time measurements, treating these signals as vectors $x, y \in \mathbb{R}^S$ with S samples.

Our aim is to construct a neural network $g_\theta(x,\phi)$ that produces a signal $\hat{y}$ perceptually indistinguishable from the output y of the real effect. The modeling process involves training $g_\theta(x,\phi)$ with a dataset of E examples $\mathcal{D} = \{(x_i, y_i, \phi_i)\}_{i=1}^{E}$ containing input-output recordings (x_i, y_i) at different device configurations ϕ_i . A loss function $\mathcal{L}(\hat{y}, y)$ is used to measure the difference between the output of the network and the target system, which provides a means to update the weights θ through a given number of optimization steps. A successful model will accurately capture the behavior of the system across the space of control parameters Φ as well as the space of all possible input signals $\mathcal{X}$.

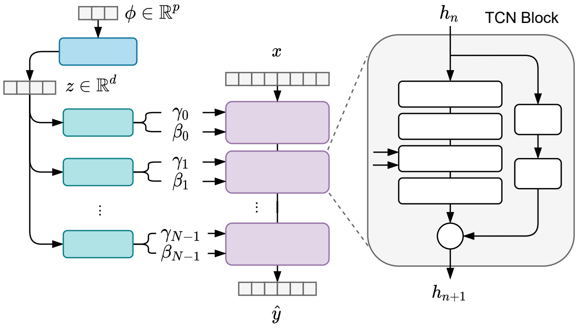

Fig. 1: TCN [20] with a series of convolutional blocks along with conditioning module (MLP) that adapts the gain γ_n and bias β_n at each layer as a function of the control parameters ϕ.

2.1 Related work

While there has been significant work in neural network approaches for distortion-based audio effects [6–15, 21–23], there has been less work in modeling analog dynamic range compression. The 1176N compressor was addressed in [8], where the authors utilized a range of different architectures including convolutional, recurrent, and a combination of the two. While their objective evaluation and listening test indicated strong performance in the task of modeling this compressor, their approach was limited in that they only considered one configuration of the device parameters. In addition, they trained and evaluated their models using only electric guitar and bass signals at a sample rate of 16 kHz, potentially limiting application to other sources.

A dataset containing measurements from the analog LA-2A was presented in [17], as well as a model using an autoencoder operating on spectral representations. While their approach modeled the device parameters, used a wide range of content (voice, music, noises, etc.), and operated at 44.1 kHz, it was found to exhibit noticeable artifacts. Further experimentation found reducing the diversity of sources in training and evaluation improved performance, but architectural modifications were not successful in addressing the artifacts [18].

Similar to the feedforward WaveNet [24] employed in modeling the 1176N [8] and distortion effects [13], a modified TCN was proposed for modeling a range of effects including compression, along with the control parameters [19, 20]. Their approach replaced the gated convolution with feature-wise linear modulation (FiLM) to adapt based on the control parameters. This

[1] https://csteinmetz1.github.io/tcn-audio-effects

approach achieved state-of-the-art performance in modeling the LA-2A, however, these models are relatively large and noncausal, prohibiting real-time operation.

This architecture of the state-of-the-art TCN is shown in Fig. 1. It consists of residual blocks, composed of 1-dimensional convolutions with increasing dilation factors, followed by batch normalization, conditional feature-wise linear modulation (FiLM) [25], and a PReLU [26] nonlinearity. To obtain a large receptive field multiple blocks are stacked with a dilation factor that grows as a power of 2 as the depth of the network increases. A network with N layers uses convolutions with a dilation at layer $n \in 0,1,\ldots,N-1$ given by $d_n = 2^n$. This enables a larger receptive field in a more efficient manner using progressively more sparse convolutional kernels.

The FiLM operation enables adaptation of the network behavior based on the control parameters. This involves an affine transformation of intermediate activations h_n with a set of scaling γ_n, and bias β_n parameters for each channel that are unique to each layer. This operation at the n^{th} layer is given by $F(h_{n,c},\gamma_{n,c},\beta_{n,c}) = \gamma_{n,c}h_{n,c} + \beta_{n,c}$, where c is the channel index. In order to generate the scaling and bias parameters for each layer, a multilayer perceptron (MLP) projects the device control parameters to an embedding $z \in \mathbb{R}^d$, shown in Fig. 1 Left. A linear layer at each block uniquely adapts z, the global conditioning, to produce $2C_n$ values, where C_n is the number of convolutional channels at the n^{th} layer.

3 Proposed method

In the design of a TCN for real-time operation we combine a causal formulation of the TCN along with an overall shallower network by using convolutions with rapidly growing dilation factors.

We first consider the requirement for noncausality, which imparts a lower-bound on the latency our system can achieve. While noncausality may aid in the modeling task, a causal TCN *should* be capable of modeling our causal analog system. We propose to do so by adopting causal convolutions, which are a common feature of TCNs [27], and have been utilized in previous work on modeling distortion effects [13].

In the case of the noncausal TCN [20], the input receptive field is split evenly between the past and future samples, such that a delay of ≈150 ms is required for adequate “look-ahead”. To achieve causality, the output must be a function only of current and previous inputs, which can be achieved with adequate padding. Causal convolutions pad the input on the left with $r-1$ samples, where r is the size of the receptive field of the model. This ensures that the output at each time-step is a function only of the current and past inputs.

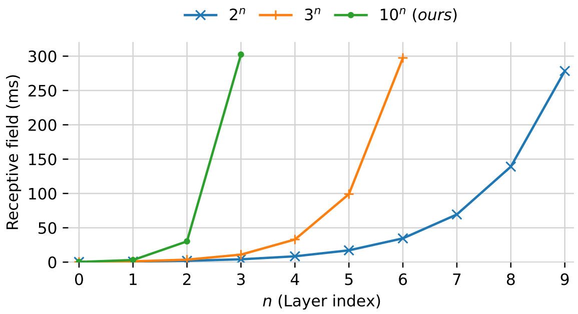

Fig. 2: Effect of the dilation growth on the receptive field of the TCN in milliseconds at $f_s = 44.1$ kHz as a function of the number of layers. Here we use a TCN with kernel size $K = 13$.

Since we opt to only pad the input signal, and not the intermediate activations, the output of each convolution will be smaller than the input. This requires we crop the residual connections in each TCN block. Care must be taken to perform cropping of the residual connections correctly depending on the causality of the model. In the noncausal case, a central crop is taken across the temporal dimension, while in the causal case, this crop selects the last S samples, where S is the number of time-steps at the output of the convolution.

However, causality does not necessarily produce a model capable of real-time operation. The model must additionally be able to process a buffer of S samples in less than S/f_s seconds, where f_s is the sample rate. To reduce the computational complexity of the TCN, we acknowledge that while computation across the temporal dimension can be parallelized within the frame, computation through the depth of the network cannot. Therefore, one straightforward route to decreasing the run-time involves simply constructing shallower networks. Unfortunately, this often comes at the cost of a smaller receptive field assuming the kernel size and dilation pattern are maintained, which often leads to a decrease in the model accuracy.

To rectify this, we propose a simple modification. We can achieve a comparable receptive field with fewer layers by using dilation factors that grow more rapidly in comparison to the base 2 convention [28], $d_n = 2^n$ with $n \in 0,1,...,N-1$. In the case of the TCN with N layers, the receptive field at the n^{th} layer is given by the recursion $r_n = r_{n-1} + (K-1) \cdot d$, where K is the kernel size, and d is the dilation factor. The receptive field at the first layer is given by the kernel size $r_0 = K$. Following this, we plot the receptive field of different TCNs as a function of the network depth N and the dilation growth with kernel size $K = 13$ in Fig. 2.

The noncausal TCN from [20] requires $N = 10$ layers in order to achieve a receptive field of approximately 300 ms. While less common, some recent speech synthesis models employ more aggressive dilation patterns, such as $d_n = 3^n$ [29, 30]. However, this still requires $N = 7$ layers to achieve a comparable receptive field. Therefore, we propose to use an even larger dilation growth, $d_n = 10^n$, which enables only $N = 4$ layers to achieve the same receptive field as previous methods. Since the dilation factors are progressively increased, the first few layers still use relatively dense filters, as in the previous approaches, yet with the later layers achieving much larger receptive field. To our knowledge, there has been no investigation of models that utilize dilation factor growth at this rate.

4 Experimental design

4.1 Dataset

To validate our proposed TCN in the analog audio effect modeling task we consider the SignalTrain dataset[2] [17]. This dataset provides approximately 20 hours of input-output recordings at $f_s = 44.1$ kHz from the analog LA-2A dynamic range compressor. It covers a diverse range of audio content including individual instruments, loops, and complete musical pieces, in addition to tones and noise bursts. This compressor features two control parameters: a binary switch that places the device in either *compress* or *limit* mode, as well as a continuous peak reduction parameter that controls the amount of compression as a function of the input level. The dataset provides audio processed by the compressor at 40 different parameter configurations, enabiling the ability to model the device at multiple different configurations. We use the same training, validation, and test split in the original dataset.

4.2 Models

We re-implement the TCN from [20, 31], which we denote TCN-324-N. This model has 10 layers with a dilation pattern given by $d_n = 2^n$, where each layer includes 32 channels. This model is noncausal and achieves a receptive field of 324 ms at $f_s = 44.1$ kHz. We also adapt the LSTM architecture proposed in [6], which we denote LSTM-32, since it features a single recurrent layer with 32 hidden units. We consider variants of the TCN to investigate the impact of noncausality, and the ability to achieve greater efficiency with shallower networks and larger dilation factors. These also employ 32 channels, but utilize a more rapidly growing dilation pattern given by $d_n = 10^n$, enabling the use of fewer layers with a similar receptive field.

In order to observe the impact of the receptive field on model performance, we train variants of the efficient TCNs with receptive fields of 101 ms (TCN-100), 302 ms (TCN-300), and 1008 ms (TCN-1000). To observe the need for noncausality, we train each model in both causal and noncausal formulations. Models ending in "-N" are noncausal, while those ending in "-C" are causal. We also investigate the amount of training data required. We train the TCN-300-C model with subsets of the dataset that contain only 10% and 1% of the training data by splitting the training set by the parameter configurations, and randomly sampling an equal amount of audio from each of these configurations.

4.3 Training

All models were trained with a batch size of 32 and inputs of 65536 samples (≈1.5 s at 44.1 kHz) for a total of 60 epochs on a single GPU. The only augmentation applied during training was a phase inversion of the input and target signals applied with probability $p = 0.5$ [17]. We employed Adam [32] with an initial learning rate of $3 \cdot 10^{-4}$, decreasing the learning rate by a factor of 10 after the validation loss had not improved for 10 epochs. In evaluation, we used the model weights from each configuration that achieved the lowest validation loss during training. Additionally, we used automatic mixed precision to decrease training time and memory consumption, which we found had negligible effect on the model performance or training stability. We have made the code to reproduce these experiments available online[3].

[2] (Version 1.1) `https://zenodo.org/record/3824876`

[3] `https://github.com/csteinmetz1/micro-tcn`

Model	*K*	*N*	*d*	*C*	*P*	**R.f.**	**RT (CPU/GPU)**	**MAE** ↓	**STFT** ↓	**LUFS** ↓
TCN-324-N [20]	15	10	2	32	162 k	324 ms	0.5x / 17.1x	1.70e-2	0.587	0.520
TCN-100-N	5	4	10	32	26 k	101 ms	4.2x / 37.1x	1.58e-2	0.768	1.155
TCN-300-N	13	4	10	32	51 k	302 ms	1.8x / 37.3x	**7.66e-3**	0.600	0.602
TCN-1000-N	5	5	10	32	33 k	1008 ms	0.5x / 26.4x	1.20e-1	0.736	0.934
TCN-100-C	5	4	10	32	26 k	101 ms	5.0x / 37.2x	1.92e-2	0.770	1.225
TCN-300-C	13	4	10	32	51 k	302 ms	2.2x / 37.3x	1.44e-2	0.603	0.761
TCN-1000-C	5	5	10	32	33 k	1008 ms	0.6x / 26.4x	1.17e-1	0.692	0.899
LSTM-32	-	-	-	-	5 k	-	0.9x / 2.8x	1.10e-1	**0.551**	**0.361**

Table 1: Performance on the LA-2A test set. Models ending with -N are noncausal, and those ending -C are causal. *K* is the kernel size, *N* is the number of layers, *d* is the dilation growth factor, *C* is the number of convolutional channels, and *P* is the total number of trainable parameters. R.f is the receptive field in milliseconds. The real-time factor (RT) is reported on CPU and GPU with a frame size of 2048 samples.

Model	*C*	*P*	**RT**	**MAE**	**STFT**	**LUFS**
324-N	32	162 k	0.5x / 17.1x	1.70e-2	**0.587**	**0.520**
324-N	16	47 k	1.3x / 17.1x	4.38e-2	0.796	1.305
324-N*	8	16 k	2.2x / 17.1x	5.29e-2	1.143	1.315
300-C	32	51 k	2.2x / 33.4x	**1.44e-2**	0.603	0.761

Table 2: TCN-324 models using fewer convolutional channels. *Model diverged during training.

4.4 Loss function

For training we used a combination of the error in the time and frequency domains. We compute the mean absolute error (MAE) for the time domain component $\mathcal{L}_{\text{time}}$ and the multi-resolution short-time Fourier Transform error [31, 33] for the frequency domain $\mathcal{L}_{\text{freq}}$ component as used in previous work [20]. The overall loss function is given as a sum of these two terms $\mathcal{L}_{\text{overall}} = \mathcal{L}_{\text{time}} + \alpha \cdot \mathcal{L}_{\text{freq}}$. We used $\alpha = 1$ in all experiments. In effect this weights the frequency domain loss more greatly, due to the differing scales of the terms.

4.5 Metrics

We considered three metrics for the objective evaluation of the models. The first two are components of the training objective, the MAE of the time-domain signal, and the multi-resolution STFT error (denoted STFT). As a perceptually informed metric, we define the loudness error as the absolute error between the loudness of the prediction and target signals computed using the ITU-R BS.1770 perceptual loudness recommendation [34, 35]. With this metric we can measure to what degree the perceived loudness was captured by the model, which is likely correlated with the application of the correct gain reduction as a result of compression.

5 Results

Results comparing our causal and efficient TCNs to previous approaches are shown in Table 1. The model hyperparameters, *K* kernel size, *N* number of layers, and *d* dilation growth factor are reported, along with the number of model parameters *P* and the receptive field in milliseconds. We report the real-time factor (RT) for a frame size of 2048 samples, which is described in more detail in Sec. 5.3. These results suggest that causal formulations of the TCN are able to achieve comparable performance to their noncausal variants, with the most significant difference being that noncausal models appear to achieve slightly superior time domain performance and lower dB LUFS error. However, the TCN-1000-C model is an exception, performing slighter better than the TCN-1000-N across all metrics.

With regards to the TCNs, it appears that models with around 300 ms of receptive field achieve superior performance. Although, this may be due to the smaller number of parameters in the models with different receptive field. Nevertheless, the TCN-1000-C model, which features few parameters and the largest receptive field, is not capable of real-time operation. Our efficient TCNs, which employ very large dilation growth factors and are shallower than the TCN-324-N model, yet have comparable receptive field and performance while using a third of the parameters and providing up to four times faster run-time on CPU.

Notably, the LSTM-32 model achieves the best performance across both the STFT and LUFS metrics, but an order of magnitude worse with respect to the time-domain performance (MAE). However, it is difficult

Model	Data	Config	Total	MAE	STFT	LUFS
324-N	100%	30 m	19.5 h	1.70e-2	0.587	**0.520**
300-C	100%	30 m	19.5 h	1.44e-2	0.603	0.761
300-C	10%	3.0 m	1.9 h	**1.38e-2**	**0.587**	0.630
300-C	1%	0.3 m	11.3 m	1.40e-2	0.599	0.740

Table 3: TCN-300-C with varying amount of data.

to make a conclusion based soley on these objective metrics, which motivates our listening study as outlined in Sec 5.4 The strong performance of the LSTM-32 demonstrates the major advantage of recurrent models, namely that they are able to achieve an adaptive receptive field in a parameter efficient manner. In this case, the LSTM-32 uses 32x fewer parameters than the TCN-324 model. Nevertheless, while this class of models is parameter efficient, processing across the temporal dimension cannot be parallelized. In this case, the LSTM-32 model is not capable of real-time operation on CPU in the PyTorch implementation even when compiled via torchScript[4]. Additionally, the LSTM-32 model required over 8 times longer to train (108 hr) compared to the TCN-300-C model (13 hr).

5.1 Parameter scaling

To further demonstrate the efficacy of larger dilation factors, we demonstrated that merely scaling down the parameters of the TCN-324-N model does not provide comparable accuracy and efficiency. We trained narrower variants of the TCN-324-N model with fewer convolutional channels, as shown in Table 2. We found that while scaling down the width of these models does increase the real-time factor, it comes at the cost of performance, with the TCN-300-C significantly outperforming these variants. This strengthens our claim that using very sparse convolutional kernels is an effective method for achieving sufficient receptive field without sacrificing performance in the modeling task.

5.2 Data efficiency

While the SignalTrain dataset provides 20 hours of recordings from the LA-2A, we investigated the requirement for such a large dataset. We split the original training dataset into random subsets, with a balanced number of examples for each parameter configuration. The 10% subset contains a total of 1.9 hours of audio with 3 minutes of audio per configuration of the compressor parameters. Furthermore, the 1% subset results in a total of just 11 minutes of audio in total, with only 18 seconds per configuration. Results for the TCN-300-C trained with these subsets are compared against TCNs trained with the complete dataset in Table 3.

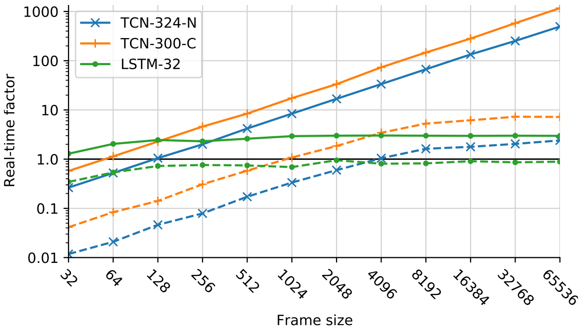

Fig. 3: RT on GPU (solid) and CPU (dashed) at different frame sizes. RT greater than 1 is required for real-time operation.

We found reducing the size of the training dataset did not significantly impact performance. Surprisingly, there is an improvement in performance using the smaller training subsets. We hypothesize this could be due to some special characteristics of the random subsets that were selected. For example, perhaps more samples with tones and noise bursts were selected, which could be more informative, or vice versa. This indicates that modeling related analog dynamic range compression effects could be achieved with significantly smaller datasets. This greatly lowers the burden in creating such datasets and agrees with findings from previous works in modeling other effects [6, 11].

5.3 Compute efficiency

We investigated the run-time of these models in a block-based implementation that aims to mimic a standard audio effect. The real-time factor (RT) is defined as

$$\mathrm{RT} := \frac{S}{T \cdot f_s}, \tag{1}$$

where S is the number of samples processed at a sampling rate of f_s, and T is the time in seconds to process those S samples. We measure the real-time factor at power of 2 frame sizes, $F \in 32, 64, \ldots, 65536$, on both GPU and CPU. For GPU, measurements are performed on a RTX 3090, and for CPU, measurements are performed on a 2018 MacBook Pro with an Intel Core i7-8850H @ 2.6 GHz. Results are shown in Fig. 3

[4] https://pytorch.org/docs/stable/jit.html

on GPU (solid lines) and CPU (dashed lines). In this block-based formulation, the TCN models require a buffer of past samples such that we pass an input of $S+r-1$ samples, where S is the number of output samples and r is the receptive field in samples.

For the LSTM, the real-time factor on both GPU and CPU is constant with respect to the frame size, which is due to the inability to parallelize computations across the temporal dimension. In our PyTorch implementation, we found the LSTM was close, but not able to achieve real-time operation. On the other hand, we found the real-time factor for the TCN model is proportional to the frame size, with larger frame sizes producing greater real-time factors as a result of greater parallelization, both on CPU and GPU. This enables real-time operation on CPU at frame sizes down to 1024 samples, which we found also to be the case in our implementation of the model in a JUCE plugin.

This understanding of recurrent and convolutional models can help guide the architecture design process for modeling effects. In cases where very low latency is required, assuming a recurrent model of sufficient size can run in real-time on the target platform, these models provide a good option. On the other hand, convolutional models demonstrate a clear advantage in that larger frame sizes will provide a significant speedup, useful in offline use cases, such as rendering a mixdown, or when using neural audio effects in other contexts, such as automatic mixing [20]. These results represent a worse-case scenario, since optimized C++ implementations may achieve a speedup compared to the PyTorch models used in our analysis [6, 7, 9, 36].

5.4 Listening study

To further evaluate model performance, we carried out a multistimulus listening test, similar to MUSHRA [37]. Five passages from the test set were used, each around 12 seconds in duration. We processed these stimuli using the SignalTrain model, the LSTM-32 model, and our proposed causal TCN-300-C model trained with 1% of the dataset ($\approx 10\,\text{min}$). We did not include a low quality anchor as there was no clear choice in the case of dynamic range compression [38, 39]. We used webMUSHRA [40], which enabled the study to be performed online, and allowed participants to instantaneously switch between different stimuli in order to facilitate comparison of small differences.

We enlisted 19 participants, all of whom reported experience with audio engineering and were familiar with the LA-2A. We performed a post-screening analysis to assess the participants, and removed ratings from one participant who assigned the reference a score of less than 50 in 4 of the 5 passages.Results from the remaining 18 participants are presented in Fig. 4.

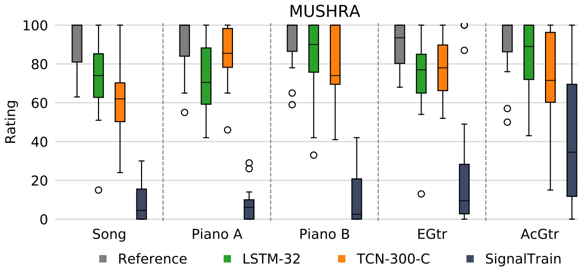

Fig. 4: Ratings of the five passages from the MUSHRA style listening studying with 18 participants after the post-screening process.

Both the LSTM-32 and TCN-300-C performed slightly below the reference. With some stimuli the median rating of the LSTM-32 is greater (Song, Piano B, AcGtr), while at other times the TCN-300-C is greater (Piano A, EGtr). In contrast, it is clear that participants noticed the strong noise-like artifacts produced by the SignalTrain model. Some participants struggled to differentiate between the reference and LSTM-32 and TCN-300-C models, as they rated the reference lower than these models in some cases. This is evident from the high variance in the ratings for the reference.

To formalize these observations, we performed the Kruskal-Wallis H-test, which indicated a difference in the median rating of the models ($F = 186.7, p = 3.21 \cdot 10^{-40}$). A post hoc analysis using Conover's test of multiple comparisons revealed a significant difference in the ratings for the reference and the LSTM-32 ($p_{\text{adj}} = 3.65 \cdot 10^{-11}$) and TCN-300-C ($p_{\text{adj}} = 6.84 \cdot 10^{-9}$). This indicated, that while challenging, listeners likely perceived a small difference among the models in comparison to the reference.

Nevertheless, it appears there is no significant difference in the median ratings between the LSTM-32 and TCN-300-C ($p_{\text{adj}} = 0.37$). These results appear to agree with comments from participants, where both the LSTM-32 and TCN-300-C models were found to very closely capture the character of the LA-2A without imparting artifacts, but differ in cases of strong gain reduction, letting some transients pass through more so than the analog LA-2A.

6 Conclusion

We demonstrated that TCNs employing causal convolutions with rapidly growing dilation factors enable shallow networks to achieve sufficient receptive field in a compute-efficient manner. This causal and efficient TCN formulation was effective in modeling the analog LA-2A dynamic range compressor, ultimately enabling real-time operation on CPU. A listening study found that our proposed model achieved a high level of perceptual similarity to the original device, outperforming the previous SignalTrain model, using only 1% of the full training dataset in the process. However, our results indicated that while challenging, listeners were often able to differentiate the emulations from the original device, leaving room for further improvement. Directions for future investigation involve optimizations in platform specific implementations for further efficiency in real-time operation, as well as investigating how TCNs with rapidly growing dilation factors generalize to other audio effects and related audio signal processing tasks.

Acknowledgements

This work is supported by the EPSRC UKRI Centre for Doctoral Training in Artificial Intelligence and Music (EP/S022694/1).

References

[1] Karjalainen, M. and Pakarinen, J., "Wave digital simulation of a vacuum-tube amplifier," in *ICASSP*, 2006.

[2] Yeh, D. T., Abel, J. S., and Smith, J. O., "Automated physical modeling of nonlinear audio circuits for real-time audio effects—Part I: Theoretical development," *IEEE Transactions on Audio, Speech, and Language Processing*, 18(4), pp. 728–737, 2009.

[3] Eichas, F., Möller, S., and Zölzer, U., "Block-oriented modeling of distortion audio effects using iterative minimization," in *DAFx*, 2015.

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

[5] Gerat, E., Eichas, F., and Zölzer, U., "Virtual analog modeling of a UREI 1176LN dynamic range control system," in *143rd AES Convention*, 2017.

[6] Wright, A., Damskägg, E.-P., Välimäki, V., et al., "Real-time black-box modelling with recurrent neural networks," in *DAFx*, 2019.

[7] Damskägg, E.-P., Juvela, L., Välimäki, V., et al., "Real-Time Modeling of Audio Distortion Circuits with Deep Learning," in *Sound and Music Computing Conf. (SMC)*, 2019.

[8] Martínez Ramírez, M. A., Benetos, E., and Reiss, J. D., "Deep Learning for Black-Box Modeling of Audio Effects," *Applied Sciences*, 10(2), p. 638, 2020.

[9] Chowdhury, J., "A Comparison of Virtual Analog Modelling Techniques for Desktop and Embedded Implementations," *arXiv:2009.02833*, 2020.

[10] Covert, J. and Livingston, D. L., "A vacuum-tube guitar amplifier model using a recurrent neural network," in *IEEE SoutheastCon*, 2013.

[11] Schmitz, T. and Embrechts, J.-J., "Nonlinear real-time emulation of a tube amplifier with a long short time memory neural-network," in *144th AES Convention*, 2018.

[12] 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 *IEEE SoutheastCon*, pp. 1–5, 2018.

[13] Damskägg, E.-P., Juvela, L., Thuillier, E., and Välimäki, V., "Deep learning for tube amplifier emulation," in *ICASSP*, pp. 471–475, IEEE, 2019.

[14] Ramírez, M. A. M. and Reiss, J. D., "Modeling nonlinear audio effects with end-to-end deep neural networks," in *ICASSP*, 2019.

[15] Nercessian, S., Sarroff, A., and Werner, K. J., "Lightweight and interpretable neural modeling of an audio distortion effect using hyperconditioned differentiable biquads," in *ICASSP*, IEEE, 2021.

[16] 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.

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

[18] Mitchell, W. and Hawley, S. H., “Exploring Quality and Generalizability in Parameterized Neural Audio Effects,” in *149th AES Convention*, 2020.

[19] Steinmetz, C. J., *Learning to mix with neural audio effects in the waveform domain*, Master’s thesis, Universitat Pompeu Fabra, 2020, `https://doi.org/10.5281/zenodo.4091203`.

[20] Steinmetz, C. J., Pons, J., Pascual, S., and Serrà, J., “Automatic multitrack mixing with a differentiable mixing console of neural audio effects,” in *ICASSP*, 2021.

[21] Kuznetsov, B., Parker, J. D., and Esqueda, F., “Differentiable IIR filters for machine learning applications,” in *DAFx*, 2020.

[22] Wright, A. and Välimäki, V., “Perceptual loss function for neural modeling of audio systems,” in *ICASSP*, pp. 251–255, 2020.

[23] Ramírez, M. A. M., Benetos, E., and Reiss, J. D., “A general-purpose deep learning approach to model time-varying audio effects,” in *DAFx*, 2019.

[24] Rethage, D., Pons, J., and Serra, X., “A WaveNet for speech denoising,” in *ICASSP*, pp. 5069–5073, 2018.

[25] Perez, E., Strub, F., De Vries, H., Dumoulin, V., and Courville, A., “FiLM: Visual reasoning with a general conditioning layer,” in *AAAI Conf. on Artificial Intelligence*, 2018.

[26] He, K., Zhang, X., Ren, S., and Sun, J., “Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification,” in *ICCV*, 2015.

[27] Bai, S., Kolter, J. Z., and Koltun, V., “An empirical evaluation of generic convolutional and recurrent networks for sequence modeling,” *arXiv:1803.01271*, 2018.

[28] Yu, F. and Koltun, V., “Multi-Scale Context Aggregation by Dilated Convolutions,” in *ICLR*, 2016.

[29] Yang, G., Yang, S., Liu, K., Fang, P., Chen, W., and Xie, L., “Multi-band MelGAN: Faster waveform generation for high-quality text-to-speech,” *arXiv:2005.05106*, 2020.

[30] Tian, Q., Chen, Y., Zhang, Z., Lu, H., Chen, L., Xie, L., and Liu, S., “TFGAN: Time and Frequency Domain Based Generative Adversarial Network for High-fidelity Speech Synthesis,” *arXiv:2011.12206*, 2020.

[31] Steinmetz, C. J. and Reiss, J. D., “auraloss: Audio focused loss functions in PyTorch,” in *DMRN+15*, 2020.

[32] Kingma, D. P. and Ba, J., “Adam: A method for stochastic optimization,” *arXiv:1412.6980*, 2014.

[33] Yamamoto, R., Song, E., and Kim, J.-M., “Probability density distillation with generative adversarial networks for high-quality parallel waveform generation,” in *INTERPSEECH*, 2019.

[34] ITU-R BS.1770-4, “Algorithms to Measure Audio Programme Loudness and True-peak Audio Level,” Recommendation, International Telecommunications Union, 2015.

[35] Steinmetz, C. J. and Reiss, J. D., “pyloudnorm: A simple yet flexible loudness meter in Python,” in *150th AES Convention*, 2021.

[36] Chowdhury, J., “RTNeural: Fast Neural Inferencing for Real-Time Systems,” *arXiv:2106.03037*, 2021.

[37] ITU-R BS.1534-3, “Method for the subjective assessment of intermediate quality level of audio systems,” Recommendation, International Telecommunications Union, 2015.

[38] Maddams, J. A., Finn, S., and Reiss, J. D., “An autonomous method for multi-track dynamic range compression,” in *DAFx*, 2012.

[39] Ma, Z., De Man, B., Pestana, P. D., Black, D. A., and Reiss, J. D., “Intelligent multitrack dynamic range compression,” *Journal of the Audio Engineering Society*, 63(6), pp. 412–426, 2015.

[40] 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.