Efficient Neural Networks for Real-time Analog Audio Effect Modeling

Preprint · February 2021

2 authors, including:

Christian J Steinmetz
Queen Mary, University of London
19 PUBLICATIONS **84** CITATIONS

Efficient Neural Networks for Real-time Analog Audio Effect Modeling

Christian J. Steinmetz and Joshua D. Reiss
Centre for Digital Music, Queen Mary University of London, London, UK
c.j.steinmetz@qmul.ac.uk

***Abstract*—Deep learning approaches have demonstrated success in the task of modeling analog audio effects such as distortion and overdrive. Nevertheless, challenges remain in modeling more complex effects, such as dynamic range compressors, along with their variable parameters. Previous methods are computationally complex, and noncausal, prohibiting real-time operation, which is critical for use in audio production contexts. They additionally utilize large training datasets, which are time-intensive to generate. In this work, we demonstrate that shallower temporal convolutional networks (TCNs) that exploit very large dilation factors for significant receptive field can achieve state-of-the-art performance, while remaining efficient. Not only are these models found to be perceptually similar to the original effect, they achieve a 4x speedup, enabling real-time operation on CPU, and can be trained using only 1% of the data from previous methods.**

I. Introduction

Audio effects provide the ability to adjust perceptual attributes of audio signals such as loudness, timbre, pitch, spatialization, or rhythm, and form a core component of the tools used by audio engineers [1]. While a significant amount of processing in audio 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 significant interest in virtual analog modeling [2]–[10], the task of constructing digital models to emulate analog devices.

Digital models of analog equipment have a number of advantages. These include the ability to run multiple instances of the effect concurrently, a reduction in physical space compared to their analog counterparts, and most notably, a reduction in the cost of production compared to analog hardware [11]. Therefore, not only does virtual analog modeling help preserve the heritage of vintage audio hardware, it also enables a larger audience to have access to such tools. Beyond modeling analog effects, neural audio effects also have applications in automatic multitrack mixing [12], where more efficient implementations could enable advancements.

Due to the complex nonlinear behavior of many of these audio signal processing devices, traditional system identification techniques for linear systems, such as the impulse response, are inadequate, and instead advanced modeling approaches are required. Over the past decade, a number of approaches have been investigated, such as wave digital filters (WDF) [2] and Volterra series [13] for modeling vacuum-tube amplifiers, Wiener-Hammerstein models for modeling distortion circuits [4], and state-space models for dynamic range compressors [14]. While these methods have achieved some success, they may require knowledge of the underlying analog circuit, be computationally complex, or otherwise unable to capture some types of nonlinearities. For these reasons, there is growing interest in the application of deep learning to overcome these limitations in a data-driven manner.

Thus far, applications of neural networks have focused mostly on modeling vacuum-tube amplifiers [15]–[18] and distortion circuits [7], [8], [10], [19], with some demonstrating the ability to run in real-time on CPU. In contrast, dynamic range compressors [20] pose a greater challenging in the modeling task due to their time-dependant nonlinearities, and have so far seen less attention. While Martínez Ramírez et al. [9] briefly address the 1176N compressor, they do not consider modeling the parameters of the compressor and only utilize electric guitar and bass signals. Hawley et al. [21], [22] model the LA-2A compressor and its parameters using diverse content, but while they capture the overall characteristics of the device, their model exhibits artifacts, is noncausal, and with over 4 M parameters, is not capable of real-time operation. Recently, temporal convolutional networks (TCNs) have been shown to be successful in audio effect modeling [23]. While they demonstrate promising results in modeling the LA-2A compressor, these models are also noncausal, computationally expensive, and utilize a significant amount of training data.

We aim to address these limitations with a more efficient formulation of the TCN that employs causal convolutions. 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 greater efficiency, but often lack sufficient receptive field. We find that by using rapidly growing dilation factors, larger than previously used, we can construct shallow networks that achieve comparable receptive field with greater efficiency, producing minimal impact on performance. Our contributions are summarized as follows:

- We employ causal convolutions with rapidly growing dilation factors to construct shallower networks for modeling an analog dynamic range compressor.
- These models run in real-time on both GPU and CPU, and we characterize the trade-off between feedforward and recurrent models for real-time audio processing.
- We demonstrate that only 1% of the data from previous approaches is sufficient for training these models.
- In a listening test, our real-time models produce results perceptually similar to the original effect.

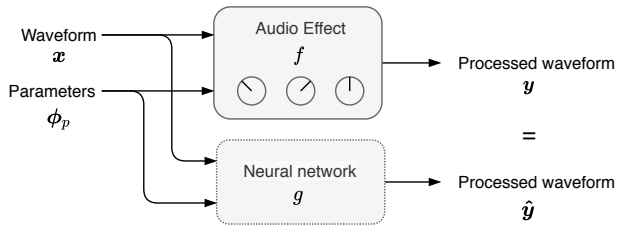

Fig. 1. Block diagram of the audio effect modeling task where a neural network g is used to emulate a parametric audio effect f.

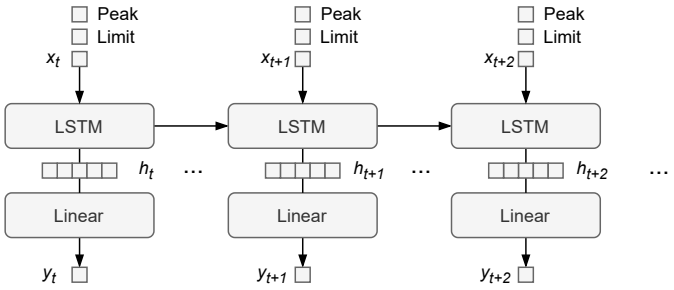

Fig. 2. General LSTM with conditioning based on Wright et al. [7] The input at each time step is a vector of 3 elements: the current input sample, along with the conditioning parameters for the limit and peak reduction controls.

II. Audio effect modeling

In this task, we consider an audio effect f, a function that takes as input an audio signal $\mathbf{x} \in \mathbb{R}^N$ of N samples, and a set of P parameter values $\boldsymbol{\phi} \in \mathbb{R}^P$, e.g. the value of the knobs of the device shown in Fig. 1. This function then produces a corresponding processed version of the signal $\mathbf{y} \in \mathbb{R}^N$. Our aim is then to construct a model g that takes as input $\mathbf{x}$ and $\boldsymbol{\phi}$, and produces a signal $\hat{\mathbf{y}}$ that is perceptually indistinguishable from $\mathbf{y}$. We formalize this goal across the space of device parameters and input signals as shown in Eq. 1. We intend to construct g as a neural network and employ modern deep learning methods to learn to emulate the original audio effect f, using pairs of input-output $(\mathbf{x}, \mathbf{y})$ recordings from the device.

$$g(\boldsymbol{\phi}, \mathbf{x}) = f(\boldsymbol{\phi}, \mathbf{x}), \ \forall \ \boldsymbol{\phi}, \mathbf{x}. \tag{1}$$

While there have been a number of works that apply neural networks for the audio effect modeling task, many of these works consider only a single configuration of the device, i.e. they optimize g without conditioning at only a single value of $\boldsymbol{\phi}$ [15], [17], [24]. Other approaches consider multiple parameterizations, but during training use only a subset of input signal types $\mathbf{x}$, e.g. only electric guitar and bass signals [7], [8], [16]. While such an approach may be applicable in the case of modeling guitar amplifiers and distortion effects, for general audio processing devices, like the dynamic range compressor, we cannot make such strong assumptions. Instead, we consider modeling the device across all configurations $\boldsymbol{\phi}$ and all realistic audio signals $\mathbf{x}$. This complicates the modeling process and likely requires we collect more data, build larger models with greater expressivity, and train for a longer period. All of these details present unique challenges, especially when we aim to deploy these models in a real-time audio application on CPU.

III. Related work

A collection of architectures have been proposed for the audio effect modeling task. These can be divided into three categories, recurrent neural networks (RNNs), and their variants (LSTM, GRU, vanilla RNN), temporal convolutional networks (TCNs), also known as the feedfoward WaveNet, and architectures that combine both elements. Simple RNNs have been shown to be effective in modeling nonlinear effects like those produced from vacuum-tube amplifiers and guitar distortion effects, often within perceptual tolerances [7], [10], [15]–[17]. These formulations process the signal on a sample-by-sample basis. While many approaches consider only a single configuration of the device parameters, Wright et al. [7] demonstrated device parameters can be modeled by appending them directly to the input sequence as conditioning as shown in Fig. 2. While these models are easy to implement and are capable of running in real-time in optimized C++ implementations, they have well known disadvantages, namely the inability to parallelize computations across the temporal dimension, overall slower convergence, and empirical limits to their memory [25]. So far these issues have not yet restricted the application of RNNs for modeling distortion-based effects. However, they may cause issues when scaling these methods for more complex nonlinear modeling tasks, like that of the dynamic range compressor, which we will examine in Sec. VI.

In contrast to RNN-based approaches, Damskägg et al. [8] proposed the application of TCNs to the same task of modeling guitar distortion effects. The TCN is a generalization of convolutional networks applied to sequence modeling (dilated 1-dimensional convolution + nonlinearity). Interestingly, yet maybe somewhat unsurprisingly, these models resemble Wiener-Hammerstein models [26], a traditional statistical approach to modeling nonlinear systems. TCNs were popularized in the autoregressive WaveNet [27] for speech synthesis, but have now been adapted for many tasks, including both causal and noncausal [28] feedforward formulations.

The general architecture for the TCN in the context of audio effect modeling is shown in Fig. 3, based upon the implementation from [12]. It consists of residual blocks, composed of 1-dimensional convolutions with increasing dilation factors, followed by batch normalization, a conditional affine transformation (FiLM) [29], and a PReLU [30] nonlinearity. A common approach to obtaining a large receptive field is to stack multiple blocks with a dilation factor that grows as a power of 2 as the depth of the network increases, such that the dilation of the l-th layer is given by $d_l = 2^{l-1}$.

Since no padding is applied other than to the input sequence, the output of each convolution will be smaller than the input. This requires we crop the residual connections. The residual connections are implemented with 1x1 grouped convolutions, which enables only scaling of the features. Care must be taken to perform cropping of the residual connections correctly depending on the causality of the model. In the noncausal case,

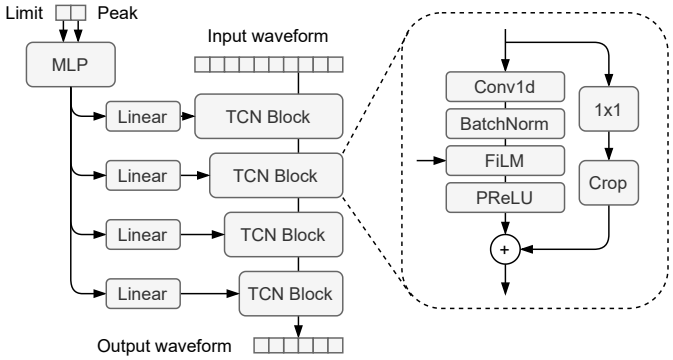

Fig. 3. General TCN architecture featuring a global conditioning module (3 layer MLP) that generates embeddings for each FiLM operation in the TCN processing pipeline as a function of the limit and peak reduction controls. The contents of the TCN block are shown in the dashed block on the right.

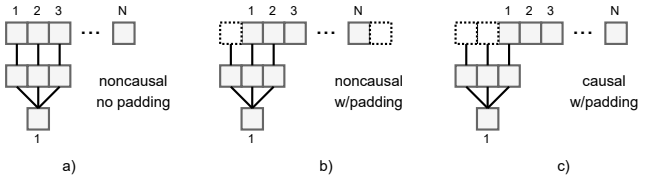

Fig. 4. Illustration of padding approaches for an input signal of N samples with a convolutional kernel of size 3. a) Noncausal TCN formulation where no padding is applied [12], b) Standard "same" padding, which is also noncausal, c) Truly causal padding, as we employ.

we employ a central crop across the temporal dimension, while in the causal case, this crop selects the last N samples, where N is the number of time-steps at the output of the convolution.

The traditional gated convolution [31] for conditioning is replaced with FiLM, which performs a conditional affine transformation of each channel activation $\mathbf{x}_c$, with scaling γ_c, and bias β_c parameters. The output of this operation at each layer is given by

$$F(\mathbf{x}_c, \gamma_c, \beta_c) = \gamma_c \mathbf{x}_c + \beta_c, \tag{2}$$

where c is the channel index. Batch normalization, (as shown in Eq. 3 without its own affine transformation) is applied before this operation, not only to stabilize training, but to ensure the conditioning information is applied optimally, since the activations will be normalized first.

$$\mathbf{y} = \frac{\mathbf{x} - \mathrm{E}[\mathbf{x}]}{\sqrt{\mathrm{Var}[\mathbf{x}] + \epsilon}} \tag{3}$$

In order to generate these scaling and bias parameters, a 3 layer multilayer perceptron (MLP) projects the device control parameters (limit and peak reduction controls in the case of the LA-2A compressor) into a 32-dimensional embedding, shown in Fig. 3 Left. This embedding is then passed to a linear layer at each block, which uniquely adapts the conditioning. Note that the γ_c and β_c parameters will adapt at inference based on the device control parameters, extending the expressivity of the model. This form of conditioning is not only simpler, but has demonstrated superior performance as a conditioning mechanism in convolutional networks in many domains, including audio [32]–[34].

IV. Efficient TCNs

In the design of a TCN for real-time operation, we first consider the requirement for noncausality, which imparts a lower-bound on the latency our system can achieve. In the case of the noncausal TCN-324-N model [35], the input receptive field is split evenly between the past and present samples, such that a delay of ≈150 ms is required for adequate "look-ahead" in a real-time context. Even so, this additionally assumes the model can operate in real-time on the provided hardware.

While noncausality may aid in the modeling task, we should be able to accurately model this device with a causal model since the analog LA-2A is itself causal. We propose to do so by adopting causal convolutions, which are a common feature of TCNs [25]. Causal convolutions are achieved by padding the input on the left with $r-1$ samples, where r is the receptive field of the model. In the case of the TCN, the receptive field at layer l is given by $r_l = r_{l-1} + (k-1) \cdot d$, where k is the kernel size, and d is the dilation factor. This padding is illustrated in Fig. 4c for a convolution with a kernel size of 3. In order to achieve causality, the output must be a function only of current and previous time-step input values. Notice that the "same" padding employed in deep learning frameworks will not be causal (Fig. 4b), as will no padding (Fig. 4a).

Nevertheless, causality does not necessarily produce a model capable of real-time operation. The model must additionally be able to process a buffer of N samples in less than $N \cdot f_s$ 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, computation through the depth of the network cannot. This imposes a run-time constraint that is a function of the model depth, as well as other factors like the number of convolutional channels, the hardware platform (CPU/GPU), and the convolution implementation. Therefore, one straightforward route to decreasing the run-time involves simply constructing shallower networks. Unfortunately, this comes at the cost of a smaller receptive field, which is critical for the modeling task.

In order to rectify this, we propose a simple adjustment. 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, $d_l = 2^{l-1}$. While less common, some recent speech synthesis models employ more aggressive dilation patterns, such as $d_l = 3^{l-1}$ [36], [37]. Though this will achieve a larger receptive field, it will not produce a significant difference when using only a few layers, as we desire for this efficient application. Therefore, we propose to use even larger dilation growth, $d_l = 10^{l-1}$, which enables only four layers to achieve the same receptive field as previous methods. To our knowledge, there has been no investigation of models that utilize dilation factor growth at this rate.

V. Loss functions

Audio effect modeling tasks generally consider a loss in the time-domain between the target output y and the prediction from the model $\hat{y}$. Commonly, this is the mean absolute error (MAE) computed on the time-domain signal (Eq. 4). Related variants such as the error-to-signal ratio [7], [18] or the log-cosh [21] have also been employed.

$$\ell_{\text{MAE}}(\hat{y}, y) = \frac{1}{N} \sum_{i=1}^{N} |\hat{y}_i - y_i| \tag{4}$$

These losses are convenient, yet enforce strict adherence to the time-domain target and its absolute phase. Nevertheless, the goal is to produce an audio signal that is perceptually similar to the original effect. Therefore, exactly following the time-domain output is not required, and such losses may place a stronger constraint than required during training [38].

This motivates the use of a spectral magnitude loss, which is largely agnostic to small phase shifts. The Short-time Fourier transform (STFT) loss proposed by Arik et al [39] is now commonplace in audio tasks, and addresses this need. It is composed of two terms, the spectral convergence ℓ_{SC} (Eq. 5), and spectral log-magnitude ℓ_{SM} (Eq. 6), where $||\cdot||_{\text{F}}$ is the Frobenius norm, $||\cdot||_1$ is the L_1 norm, and N is the number of STFT frames. The overall STFT loss ℓ_{STFT} is defined as the sum of these two terms, as shown in Eq. 7.

$$\ell_{\text{SC}}(\hat{y}, y) = \frac{\| \, |\text{STFT}(y)| - |\text{STFT}(\hat{y})| \, \|_{\text{F}}}{\| \, |\text{STFT}(y)| \, \|_{\text{F}}} \tag{5}$$

$$\ell_{\text{SM}}(\hat{y}, y) = \frac{1}{N} \left\| \log\left(\, |\text{STFT}(y)| \, \right) - \log\left(\, |\text{STFT}(\hat{y})| \, \right) \right\|_1 \tag{6}$$

$$\ell_{\text{STFT}}(\hat{y}, y) = \ell_{\text{SC}}(\hat{y}, y) + \ell_{\text{SM}}(\hat{y}, y). \tag{7}$$

Previous investigation into loss functions for the audio effect modeling task found that optimizing a time-domain loss produced better results when evaluated with time-domain metrics, and slightly worse results with frequency-domain metrics, with the opposite being true when optimizing a frequency-domain loss [35]. Therefore, to balance this, we employ a loss that combines both the MAE and the STFT loss terms as shown in Eq. 8.

$$\ell_{\text{MAE+STFT}}(\hat{y}, y) = \ell_{\text{MAE}}(\hat{y}, y) + \ell_{\text{SC}}(\hat{y}, y) + \ell_{\text{SM}}(\hat{y}, y). \tag{8}$$

This approach is not unique, and a similar approach was taken in SignalTrain, which combined the time-domain logcosh loss with a logarithmic weighted frequency error [21]. Additionally, the use of a pre-emphasis filter has also been proposed to encourage audio effect models to better capture high frequency content [40]. In our case, we found the application of the STFT loss sufficient to rectify this for modeling the LA-2A. This is likely due to the fact that this loss function places a larger weight on the high frequency content due to the linear spacing of the STFT frequency bins.

VI. Experiments

A. Dataset

We consider the SignalTrain dataset[1] introduced by Hawley et al. [21]. This dataset provides approximately 20 hours of input-output recordings at $f_s = 44.1$ kHz from an 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 particular compressor features two control parameters that are varied in the dataset: 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, known as the "threshold" on other compressors. They provide audio processed by the compressor at 40 different parameter configurations. We utilize the train/val/test split from the original dataset, which ensures that each subset contains unseen audio.

B. Models

We re-implement the TCN from [35], which we denote TCN-324-N. This model has 10 layers with a dilation pattern given by $d_l = 2^{l-1}$, 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 by Wright et al. [7], 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 $d_l = 10^{l-1}$, enabling the use of fewer layers.

In order to observe the impact of the receptive field, 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.

C. Training

All models are 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 is a phase inversion of the input and target signals applied with probability $p = 0.5$. We employ Adam with an initial learning rate of $3e^{-4}$, decreasing the learning rate by a factor of 10 after the validation loss has not improved for 10 epochs. Additionally, we use automatic mixed precision to decrease training time and memory consumption. We make the code for these experiments available[2].

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

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

Model	k	ℓ	d	c	Params	R.f.	RT (CPU/GPU)	MAE	STFT	LUFS (dB)
TCN-324-N [12]	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 [7]	-	-	-	-	5 k	-	0.9x / 2.8x	1.10e-1	**0.551**	**0.361**

TABLE I

MODEL PERFORMANCE ON THE SIGNALTRAIN TEST SET. MODELS ENDING WITH -N ARE NONCAUSAL, AND THOSE ENDING IN -C ARE CAUSAL. k IS THE KERNEL SIZE, l IS THE NUMBER OF LAYERS, d IS THE DILATION GROWTH FACTOR, AND c IS THE NUMBER OF CONVOLUTIONAL CHANNELS. REAL-TIME FACTOR (RT) IS REPORTED ON BOTH CPU AND GPU WITH A FRAME SIZE OF 2048 SAMPLES.

Model	c	Params.	RT (CPU/GPU)	MAE	STFT	dB LUFS
TCN-324-N	32	162 k	0.5x / 17.1x	1.70e-2	**0.587**	**0.520**
TCN-324-N	16	47 k	1.3x / 17.1x	4.38e-2	0.796	1.305
TCN-324-N*	8	16 k	2.2x / 17.1x	5.29e-2	1.143	1.315
TCN-300-C	32	51 k	2.2x / 33.4x	**1.44e-2**	0.603	0.761

TABLE II

VARIANTS OF THE TCN-324 MODEL USING FEWER CONVOLUTIONAL CHANNELS. *THIS MODEL DIVERGED DURING TRAINING.

Model	Data	Per config.	Total	MAE	STFT	dB LUFS
TCN-324-N	100%	30 min	19.5 hr	1.70e-2	0.587	**0.520**
TCN-300-C	100%	30 min	19.5 hr	1.44e-2	0.603	0.761
TCN-300-C	10%	3.0 min	1.9 hr	**1.38e-2**	**0.587**	0.630
TCN-300-C	1%	0.3 min	11.3 min	1.40e-2	0.599	0.740

TABLE III

TCN-300-C MODEL WITH VARYING AMOUNT OF TRAINING DATA.

VII. EVALUATION

We consider three different metrics for the objective evaluation of the models. The first two are components of the training objective, the MAE of the time-domain signal (Eq. 4), and the STFT loss (Eq. 7). 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 recommendation [41], given by

$$\ell_{\text{LUFS}}(\hat{y}, y) = |G(\hat{y}) - G(y)|, \tag{9}$$

where $G(\cdot)$ is the ITU-R BS.1770 loudness algorithm. With this metric, we can measure to what degree the perceived loudness was captured by the model, which is correlated with the application of the correct gain reduction.

Results comparing our causal and efficient TCNs to previous approaches are shown in Table I. The model hyperparameters, k kernel size, l number of layers, and d dilation growth factor are reported, along with the number of model parameters and the receptive field in milliseconds achieved at $f_s = 44.1$ kHz. We also provide the real-time factor (RT) computed for a frame size of 2048 samples, which is described in more detail in Sec. VII-B. 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.

With regards to the receptive field, 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 model, which features few parameters and the largest receptive field, is still not capable of real-time operation. The most significant finding is that our efficient TCNs, which employ very large dilation growth factors and are shallower than the TCN-324-N model, have comparable receptive field and performance while using a third of the parameters and providing up to four times faster run-time on CPU.

In order to investigate the efficacy of larger dilation factors, and to demonstrate that merely scaling down the width of the TCN-324-N model does not provide comparable accuracy and efficiency, we train narrower variants of the TCN-324-N model with fewer convolutional channels, as shown in Table II. We find 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.

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). This 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, leading to significantly slower training and inference times. In this case, the LSTM-32 model is not capable of real-time operation on CPU in the PyTorch implementation, and additionally required over 8 times longer to train (108 hr) compared to the TCN-300-C model (13 hr).

A. Data efficiency

While the SignalTrain dataset provides 20 hours of recordings from the LA-2A, we investigate the requirement for such a large dataset in this task. We split the original training dataset into random subsets, with a balanced number of examples for

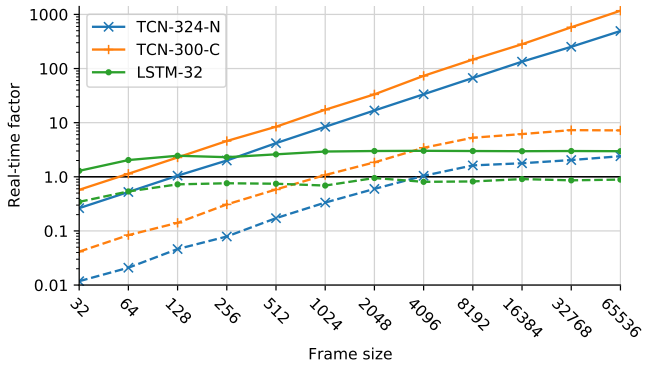

Fig. 5. RT of models on GPU (solid) and CPU (dashed) at different frame sizes. RT greater than 1 is required for real-time operation.

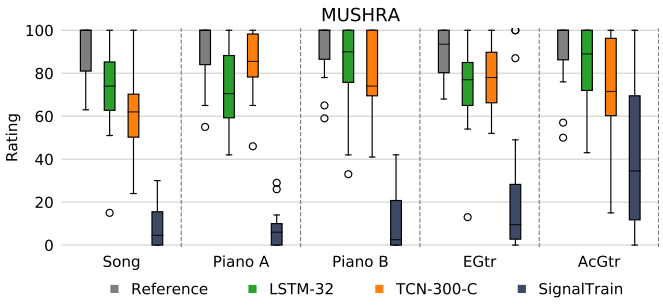

Fig. 6. Ratings of the five passages from the MUSHRA-style listening studying with 18 participants after post-screening. The TCN-300-C model used in the evaluation was trained with only 1% of the training dataset.

each parameter configuration. In the 10% subset there is a total of 1.9 hours of audio with 3 minutes of audio per configuration of the compressor. 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 model trained with these subsets are compared against the version trained with the complete dataset in Table III.

We find that reducing the size of the training dataset does not harm 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 subset that was selected, for example, selecting more samples with tones and noise bursts, which could be more informative. Nevertheless, these results are promising, indicating that modeling related analog audio effects could be achieved with significantly smaller datasets, which greatly lowers the burden in creating such datasets. This agrees with the findings from previous work on modeling much simpler effects, such as distortion [7] and guitar amplifiers [16].

B. Compute efficiency

We further investigate 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

$$\text{RT} := \frac{N \cdot f_s}{t}, \tag{10}$$

where N is number of samples processed at a sampling rate of f_s, and t is the time taken to process those N samples. For GPU, measurements are performed on a RTX 3090, and for CPU, measurements are performed on a 2018 MacBook Pro. Results are shown in Fig. 5, with solid lines representing run-time on GPU, and dashed lines representing run-time on CPU. In this block-based formulation, the TCN models require a buffer of past samples such that we pass an input of $N + r - 1$ samples, where N is the number of desired output samples and r is the receptive field of the model 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 find 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. These results indicate that on GPU, the TCN-300-C model achieves real-time operation with frame sizes down to 64 samples, and can take advantage of a 1000x speedup when operating on longer sequences, unlike the LSTM.

This understanding of recurrent and convolutional models can help guide the model design process for modeling effects. In cases where very low latency is required using small frame sizes ($\leq$ 64 samples), 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 significant speedup, useful in offline use cases, e.g. rendering a mixdown. Note also that these results are something of a worse-case scenario, since optimized C++ implementations may achieve a speedup compared to the PyTorch implementations used in our analysis [7], [8], [10].

C. Listening test

While objective metrics enable comparison of relative performance, they provide little insight into the degree to which a model emulates the device from a perceptual standpoint. To further evaluate their performance, we carried out a multistimulus listening test, similar to MUSHRA [42]. Five passages from the test set are used, each around 12 seconds in duration. We processed these stimuli using SignalTrain, the LSTM-32 model, and our proposed causal TCN-300-C model trained with 1% of the dataset. These listening examples are made available online[3]. We did not include a low quality anchor as there is no clear choice in the case of dynamic range compression [43], [44]. We used the webMUSHRA interface [45], which enabled the listening study to be performed remotely with participants using their own playback system.

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

We enlisted 19 participants, all of whom reported experience with audio engineering and were familiar with the LA-2A compressor. We performed a post-screening analysis to assess the quality of 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. 6. Both the LSTM-32 and TCN-300-C perform similarly across the passages, yet appear slightly below the reference. In contrast, it is clear that participants noticed the strong noise artifacts produced by the SignalTrain model. Interestingly, it appears 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 wide range of the ratings for the reference.

To formalize these observations, we perform the Kruskal-Wallis H-test, which indicates 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 reveals there is 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 ($p_{\text{adj}} = 6.84 \cdot 10^{-9}$) models. This indicates, that while challenging, listeners likely could perceive a small difference among the models in comparison to the reference. Nevertheless, while our objective metrics indicated the LSTM-32 model was superior, 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 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.

VIII. Discussion

TCNs are well-suited for audio effect modeling since no downsampling in time occurs throughout the network. This contrasts with autoencoding approaches, like SignalTrain, that must accurately reconstruct high frequency information in the decoder, which can be challenging. While sample-based LSTMs also perform no downsampling, and provide comparable accuracy, they are an order of magnitude slower. Additionally, they do not provide significant speedup on GPU, like the parallelizable TCNs. The most significant caveat of the TCN is that it must employ a sufficiently large receptive field. This may not be feasible for some effects with low frequency oscillators, such as phaser or chorus effects [46], [47].

Our investigations only serve to demonstrate that it is possible for this formulation of the TCN models to achieve sufficient accuracy in the modeling task while also achieving real-time operation. Further investigation is needed to determine which kinds of audio signals are most informative, and the required density of the parameter sampling to achieve sufficient performance. Future work could consider more advanced techniques to designing efficient neural networks such as quantization, distillation, and pruning, along with optimized implementations for target hardware, i.e. general purpose CPUs.

IX. Conclusion

We demonstrated TCNs employing causal convolutions with rapidly growing dilation factors can enable shallow networks to achieve sufficient receptive field in a compute-efficient manner. This causal and efficient TCN formulation was effective in modeling an analog dynamic range compressor, ultimately enabling real-time operation on CPU. Furthermore, we found that only 1% of the examples from the previously proposed dataset was necessary for adequate performance. We conducted a listening study to evaluate our efficient model and found it achieves a high level of perceptual similarity to the original effect, outperforming previous methods. We additionally provided a C++ plugin of the pre-trained model and demonstrated its ability for real-time operation on CPU. Future work involves 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.

Acknowledgement

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

References

[1] T. Wilmering, D. Moffat, A. Milo, and M. B. Sandler, "A history of audio effects," *Applied Sciences*, vol. 10, no. 3, 2020.

[2] M. Karjalainen and J. Pakarinen, "Wave digital simulation of a vacuum-tube amplifier," in *IEEE Int. Conf. on Acoustics Speech and Signal Processing (ICASSP)*, 2006.

[3] D. T. Yeh, J. S. Abel, and J. O. Smith, "Automated physical modeling of nonlinear audio circuits for real-time audio effects—Part I: Theoretical development," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 4, pp. 728–737, 2009.

[4] F. Eichas, S. Möller, and U. Zölzer, "Block-oriented modeling of distortion audio effects using iterative minimization," in *Int. Conf. on Digital Audio Effects (DAFx)*, 2015.

[5] F. Eichas, E. Gerat, and U. Zölzer, "Virtual analog modeling of dynamic range compression systems," in *142nd Audio Engineering Society Convention (AES)*, 2017.

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

[7] A. Wright, E.-P. Damskägg, V. Välimäki *et al.*, "Real-time black-box modelling with recurrent neural networks," in *Int. Conf. on Digital Audio Effects (DAFx)*, 2019.

[8] E.-P. Damskägg, L. Juvela, V. Välimäki *et al.*, "Real-time modeling of audio distortion circuits with deep learning," in *Sound and Music Computing Conf. (SMC)*, 2019.

[9] M. A. Martínez Ramírez, E. Benetos, and J. D. Reiss, "Deep learning for black-box modeling of audio effects," *Applied Sciences*, vol. 10, no. 2, p. 638, 2020.

[10] J. Chowdhury, "A comparison of virtual analog modelling techniques for desktop and embedded implementations," *arXiv:2009.02833*, 2020.

[11] V. Valimaki, F. Fontana, J. O. Smith, and U. Zolzer, "Introduction to the special issue on virtual analog audio effects and musical instruments," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 4, pp. 713–714, 2010.

[12] C. J. Steinmetz, J. Pons, S. Pascual, and J. Serrà, "Automatic multitrack mixing with a differentiable mixing console of neural audio effects," *arXiv:2010.10291*, 2020.

[13] S. Orcioni, A. Terenzi, S. Cecchi, F. Piazza, and A. Carini, "Identification of volterra models of tube audio devices using multiple-variance method," *Journal of the Audio Engineering Society (JAES)*, vol. 66, no. 10, pp. 823–838, 2018.

[14] O. Kröning, K. Dempwolf, and U. Zölzer, "Analysis and simulation of an analog guitar compressor," in *Int. Conf. on Digital Audio Effects (DAFx)*, 2011.

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

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

[17] Z. Zhang, E. Olbrych, J. Bruchalski, T. J. McCormick, and D. L. Livingston, "A vacuum-tube guitar amplifier model using long/short-term memory networks," in *Proc. of IEEE SoutheastCon*, 2018, pp. 1–5.

[18] E.-P. Damskägg, L. Juvela, E. Thuillier, and V. Välimäki, "Deep learning for tube amplifier emulation," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 471–475.

[19] M. A. M. Ramírez and J. D. Reiss, "Modeling nonlinear audio effects with end-to-end deep neural networks," in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 171–175.

[20] D. Giannoulis, M. Massberg, and J. D. Reiss, "Digital dynamic range compressor design—a tutorial and analysis," *Journal of the Audio Engineering Society*, vol. 60, no. 6, pp. 399–408, 2012.

[21] S. Hawley, B. Colburn, and S. I. Mimilakis, "Profiling audio compressors with deep neural networks," in *147th Audio Engineering Society Convention (AES)*, 2019.

[22] W. Mitchell and S. H. Hawley, "Exploring quality and generalizability in parameterized neural audio effects," in *149th Audio Engineering Society Convention (AES)*, 2020.

[23] C. J. Steinmetz, "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.

[24] M. A. M. Ramírez and J. D. Reiss, "End-to-end equalization with convolutional neural networks," in *Int. Conf. on Digital Audio Effects (DAFx)*, 2018.

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

[26] S. A. Billings and S. Fakhouri, "Identification of systems containing linear dynamic and static nonlinear elements," *Automatica*, vol. 18, no. 1, pp. 15–26, 1982.

[27] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "WaveNet: A generative model for raw audio," *arXiv:1609.03499*, 2016.

[28] D. Rethage, J. Pons, and X. Serra, "A WaveNet for speech denoising," in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 5069–5073.

[29] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville, "FiLM: Visual reasoning with a general conditioning layer," in *32nd AAAI Conf. on Artificial Intelligence*, 2018.

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. of the IEEE Int. Conf. on Computer Vision (ICCV)*, 2015, pp. 1026–1034.

[31] A. v. d. Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu, "Conditional image generation with PixelCNN decoders," in *Proc. of the Int. Conf. on Neural Information Processing Systems (NeurIPS)*, 2016, pp. 4797–4805.

[32] G. Meseguer-Brocal and G. Peeters, "Conditioned-U-Net: Introducing a control mechanism in the U-Net for multiple source separations," in *Proc. of the Int. Society for Music Information Retrieval Conf. (ISMIR)*, 2019.

[33] N. Zeghidour and D. Grangier, "Wavesplit: End-to-end speech separation by speaker clustering," *arXiv:2002.08933*, 2020.

[34] D. Petermann, P. Chandna, H. Cuesta, J. Bonada, and E. Gómez Gutiérrez, "Deep learning based source separation applied to choir ensembles," in *Proc. of the Int. Society for Music Information Retrieval Conf. (ISMIR)*, 2020.

[35] C. J. Steinmetz and J. D. Reiss, "auraloss: Audio focused loss functions in PyTorch," in *Digital Music Research Network One-day Workshop (DMRN+15)*, 2020.

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

[37] Q. Tian, Y. Chen, Z. Zhang, H. Lu, L. Chen, L. Xie, and S. Liu, "TFGAN: Time and frequency domain based generative adversarial network for high-fidelity speech synthesis," *arXiv:2011.12206*, 2020.

[38] S.-W. Fu, C.-F. Liao, and Y. Tsao, "Learning with learned loss function: Speech enhancement with quality-net to improve perceptual evaluation of speech quality," *IEEE Signal Processing Letters*, vol. 27, pp. 26–30, 2019.

[39] S. Ö. Arık, H. Jun, and G. Diamos, "Fast spectrogram inversion using multi-head convolutional neural networks," *IEEE Signal Processing Letters*, vol. 26, no. 1, pp. 94–98, 2018.

[40] A. Wright and V. Välimäki, "Perceptual loss function for neural modeling of audio systems," in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 251–255.

[41] "Algorithms to measure audio programme loudness and true-peak audio level," International Telecommunications Union, Recommendation, October 2015.

[42] "Method for the subjective assessment of intermediate quality level of audio systems," International Telecommunications Union, Recommendation, 2015.

[43] J. A. Maddams, S. Finn, and J. D. Reiss, "An autonomous method for multi-track dynamic range compression," in *Proc. of the Int. Conf. on Digital Audio Effects (DAFx)*, 2012.

[44] Z. Ma, B. De Man, P. D. Pestana, D. A. Black, and J. D. Reiss, "Intelligent multitrack dynamic range compression," *Journal of the Audio Engineering Society*, vol. 63, no. 6, pp. 412–426, 2015.

[45] M. Schoeffler, S. Bartoschek, F.-R. Stöter, M. Roess, S. Westphal, B. Edler, and J. Herre, "webMUSHRA—A comprehensive framework for web-based listening tests," *Journal of Open Research Software*, vol. 6, no. 1, 2018.

[46] V. Wright, Alec; Välimäki, "Neural modelling of LFO modulated time varying effects," in *Int. Conf. on Digital Audio Effects (DAFx)*, 2020.

[47] M. A. M. Ramírez, E. Benetos, and J. D. Reiss, "A general-purpose deep learning approach to model time-varying audio effects," in *Int. Conf. on Digital Audio Effects (DAFx)*, 2019.