Audio Engineering Society

Convention Paper

Presented at the 147th Convention
2019 October 16 – 19, New York

Profiling Audio Compressors with Deep Neural Networks

Scott H. Hawley[1], Benjamin Colburn[2], and Stylianos I. Mimilakis[3]

[1]*Department of Chemistry & Physics, Belmont University, Nashville, TN USA*
[2]*ARiA Acoustics, Washington, DC USA*
[3]*Fraunhofer Institute for Digital Media Technology, Ilmenau, Germany*

Correspondence should be addressed to Scott H. Hawley (scott.hawley@belmont.edu)

ABSTRACT

We present a data-driven approach for predicting the behavior of (i.e. profiling) a given parameterized, non-linear time-dependent audio signal processing effect. Our objective is to learn a mapping function that maps the unprocessed audio to the processed, using time-domain samples. We employ a deep auto-encoder model that is conditioned on both time-domain samples and the control parameters of the target audio effect. As a test-case, we focus on the offline profiling of two dynamic range compressors, one software-based and the other analog. Our results show that the primary characteristics of the compressors can be captured, however there is still sufficient audible noise to merit further investigation before such methods are applied to real-world audio processing workflows.

1 Introduction

The ability to digitally model musical instruments and audio effects allows for multiple desirable properties [1], among which are i) portability – virtual instruments and software effects require no space or weight; ii) flexibility – many such effects can be stored and accessed together and quickly modified; iii) signal to noise – often can be higher with digital effects; iv) centralized, automated control; v) repeatability – digital effects can be exactly the same, as opposed to physical systems which may require calibration; and vi) extension – the development of digital effects involves fewer constraints than their real-world counterparts.

The process of constructing such models has traditionally been performed using one of two main approaches. One approach is the physical simulation of the processes involved [1], whether these be acoustical processes such as reverberation[2] or "virtual analog modeling" of circuit elements [3, 4]. The other main approach has been to emulate the requisite audio features via signal processing techniques which seek to capture the salient aspects of the sounds and transformations under consideration. Both of these approaches are typically performed with the goal of faithfully reproducing one particular effect, such as audio compressors [5, 6, 7, 8].

Rather than modeling one particular effect, a different class of systems are those which can 'profile' and 'learn' to mimic the tonal effects of other units. One popular commercial example is the Kemper Profiler Amplifier [1], which can learn to emulate the sounds of amplifiers and speaker cabinets in the user's possession, to enable them to store and easily transport a virtual array of analog gear. Another product in this category is

[1]https://www.kemper-amps.com/profiler/overview

the "ToneMatch" feature of Fractal Audio's Axe-Fx [2], which supplies a large number of automatically-tunable pre-made effects units including reverberation, delay, equalization, and formant processing.

The present paper involves efforts toward the goal of profiling 'general' audio effects. For systems which are linear and time-invariant (LTI), one can develop finite impulse response (FIR) filters, *e.g.,* for convolution reverb effects. But for systems which involve nonlinearity and/or time-dependence, more sophisticated approaches are required. Deep learning has demonstrated great utility at such diverse audio signal processing tasks as classification [9], onset detection [10], source separation [11], event detection [12], dereverberation [13], denoising [14], remixing [15], and synthesis[16, 17, 18], as well as dynamic range compression to automate the mastering process [19]. In the area of audio component modeling, deep learning has been used to model tube amplifiers [20] and most recently guitar distortion pedals [21]. Besides creating specific effects, efforts have been underway to explore how varied are the types of effects which can be learned from a single model [22], to which this paper comprises a contribution. A challenging goal in deep learning audio processing is to devise models that operate directly on the raw waveform signals, in the time domain, known as "end-to-end" models [23]. Given that the raw waveform data exists in the time domain, there are questions as to whether an end-to-end formulation is most suitable [24], however it has been shown to be useful nevertheless. Our approach is end-to-end, however, we make use of a spectral representation within the autoencoders and for regularization.

Our efforts in this array have been focused on modeling dynamic range compressors for three reasons: 1. They are a common feature of audio engineering signal chains. 2. They constitute a challenging problem to solve: As noted earlier, existing methods are sufficient for many linear and/or time-independent effects. Our own investigations demonstrated that effects such as echo or distortion could be modeled via Long Short-Term Memory (LSTM) cells, but compressors proved to be 'unlearnable' to our networks. This present work therefore describes one solution to this problem. 3. They represent a set of capabilities that may be required for modeling more general effects

[2] https://www.fractalaudio.com/iii

Thus the method presented in this study is intended for learning *general audio effects*, for which the *special case* of the compressor represents a useful and challenging milestone. In this study we are not estimating compressor parameters [8], although deep neural networks have recently shown proficiency at this task as well [25]. Rather, being given parameters associated with input-output pairs of audio data, we synthesize audio by means of a network which aims to emulate 'arbitrary' mappings. The hope is that by performing well on the challenging problem of dynamic range compression, such a network could also prove useful for learning other audio effects as well. Given that the goal of our system is to successively approximate the audio signal chain through a process of training, we refer to the computer code as SignalTrain.[3]

This paper proceeds as follows: In Section 2, we describe the problem specification, the design of the neural network architecture, its training procedure, and the dataset. In Section 3 we relate results for two compressor models, one digital and one analog. Finally we offer some conclusions in section 4 and outline some avenues for future work.

2 System Design

2.1 Problem Specification

The objective is to accurately model the input-output characteristics of a wide range of musical signal processing effects and their parameterized controls, in a *model-agnostic* manner. That is to say, not to merely infer certain control parameters which are then used in conjunction with pre-made internal effect modules (*e.g.,* as is done by Axe-FX). We apply our method to the case of compressors in this paper, but we operate no internal compressor model – the system *learns what a compressor is* in the course of training using a large variety of training signals and control settings.

We conceive of the task as a supervised learning regression problem, performed in an end-to-end manner. While other approaches have made use of techniques such as μ-law companding and one-hot encoding to formulate the task as a classification problem [27], we have not yet done so. Rather than predicting one audio sample (*i.e.,* time-series value) at a time, we map a

[3] Source code and datasets accompanying this are paper publicly released via Supporting Materials [26].

range of inputs to a range of outputs, *i.e.,* we window the audio into "windows." This allows for both speed in computation as well as the potential for modeling non-causal behavior such as reverse-audio effects or time-alignment. [4]

We trained against two software compressors, with similar controls but different time scales. The effect we designate "Comp-4C" which operates in a sequential manner (later samples explicitly depend on earlier samples) and has four controls for Threshold, Ratio, Attack and Release. The other formulation, "Comp-4C-Large," allows for wider ranges of the control parameters. For an analog effect we used a Universal Audio LA-2A, output audio for a wide range of input audio as we varied the Peak Reduction knob and the Comp/Lim switch. (The input and output gain knobs were left fixed in the creation of the dataset.) These effects are summarized in Table 1.

Typically audio effects are applied to an entire "stream" of data from beginning to end, yet it is not uncommon for digital audio effects processors to be presented with only a smaller "window" (also referred to as a "chunk," "frame", "input buffer," *etc.*) of the most recent audio, of a duration usually determined by computational requirements such as memory and/or latency. For time-dependent effects such as compressors, the size of the window can have repercussions as information preceding the window boundary will necessarily propagate into the window currently under consideration. This introduces a concern over "causality," occurring over a timescale given by the exponential decay due to the compressor's attack and release controls.

This suggests two different ways to approach training, and two different ways to specify the dataset of pairs of input audio and target output audio. The first we refer to as "streamed target" (ST) data, which is the usual method of applying the audio effect to the entire stream at once. The second we refer to "windowed target" (WT) data, in which the effect is applied sequentially to individual windows of input. WT data will necessarily contain transient errors (compared to ST data) occurring on a frequency of the inverse of the window duration. If however one adds a "lookback buffer," i.e. making the length of the output shorter than that of the input, then this "lookback" can be chosen to be large enough that transient errors in the WT data decay (exponentially) below the "noise floor" before the output is generated. The goal of this study is to produce ST data as accurately as possible, as it corresponds to the normal application of audio effects, but WT data is in some sense "easier" to learn. Indeed, in our early attempts with the LA-2A compressor and ST data, the model was not able to learn *at all*, because the lookback buffer was not long enough.

The difference between ST and WT data constitutes a lower bound on the error produced by our neural network model: we do not expect the model to perform better than the "true" effect applied to WT data. The dependence of this error bound on the size of the lookback buffer can be estimated in a straightforward way, and can provide guidance on the size of buffer that should be used when training the model. Such estimates are shown in Figure 1. In order to allow for low enough error while not putting too great a strain on computational resources, we will choose model sizes with lookback windows sufficient to allow a lower bound on the error in the range of 10^{-5} to 10^{-4}.

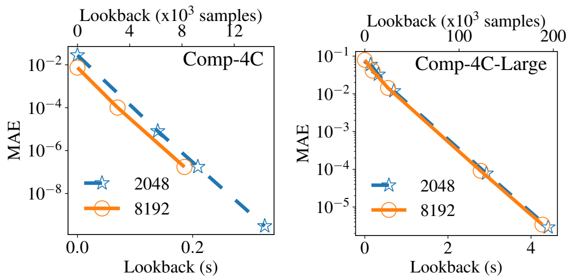

Fig. 1: Mean Absolute Error (MAE) between streamed target (ST) data and windowed target (WT) data, for the software effect as a function of lookback buffer size at 44.1 kHz, for two different input window sizes (2048 and 8192). This represents a theoretical limit for the accuracy of the neural network model.

2.2 Model Specification

The architecture of the SignalTrain model consists of the front-end, the autoencoder-like module, and the back-end module. The proposed architecture shares some similarities with the U-Net [28] and TFNet [29] architectures. Like U-Net, it has an encoder-decoder "hourglass" configuration with skip connections spanning across the middle, and like TFNet, it operates in both the time and spectral domains explicitly. The

[4] The system could be modified to predict one sample at a time, however our experience with this model has found this practice to be neither necessary nor helpful.

Effect Name	Type	Controls: Ranges
Comp-4C	Software	Threshold: -30–0 dB, Ratio: 1–5, Attack: 1–40 ms, Release: 1–40 ms
Comp-4C-Large	Software	Threshold:-50–0 dB, Ratio: 1.5–10, Attack: 0.001-1 s, Release: 0.001-1 s
LA-2A	Analog	Comp/Lim Switch: 0/1, Peak Reduction: 0–100

Table 1: Compressor effects trained. Comp-4C and Comp-4C-Large allow different control ranges but use the same Python code, which is available in supplementary materials [26].

front-end module is comprised by a set of two 1-D convolution operators that are responsible for producing a signal sub-space similar to a time-frequency decomposition, yielding magnitude and phase features. The autoencoder module consists of two deep neural networks for processing individually the magnitude and phase information of the front-end module. Each deep neural network in this autoencoder consists of 7 fully connected, feed-forward neural networks (FC). It should be noted that the "bottleneck" latent space of each deep neural network is additionally conditioned on the control variables of the audio effect module that are represented as one-hot encoded vectors.

Figure 2 illustrates the neural network architecture for the SignalTrain model, which essentially learns a mapping function from the un-processed to the processed audio, by the audio effect to be profiled, and is conditioned on the vector of the effect's controls (*e.g.,* the "knobs"). In order to obtain the predicted output waveform, the back-end module uses another set of two 1-D transposed convolutional operators. Similarly to the analysis front-end, the back-end is initialized using the bases of a discrete Fourier transform. It should be stated that all the weights are subject to optimization and are expected to vary during the training of the model. The frame and hop sizes used for the convolutions are 1024 and 384 samples, respectively.

Unlike some other proposed architectures which use convolutional layers [28], we use fully-connected (FC) layers that have shared-weights with respect to the subspace dimensionality (*i.e.,* the frequencies). That is done for two reasons. The first reason is that the number of the parameters inside the model is dramatically reduced, and secondly we preserve the location of the magnitude and phase information of the original signal. Essentially, the operations carried by each deep neural network in the autoencoder module can be seen as non-linear affine transformations of the transpose time-frequency patches (spectrograms) of the input signal. Furthermore, we apply residual (additive) skip connections [31] inspired by U-Net [28] and "skip filter"

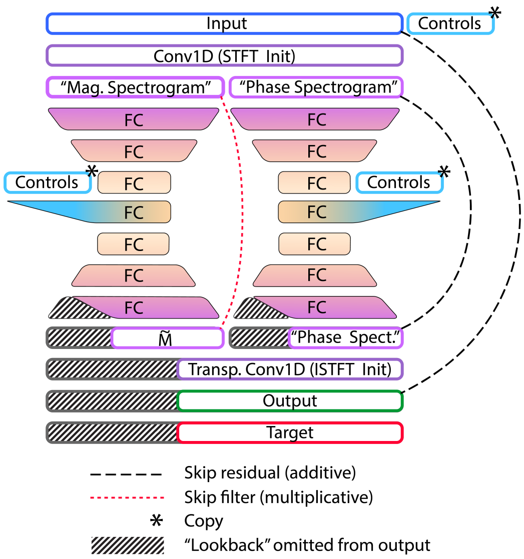

Fig. 2: Diagram of the SignalTrain network architecture. The designations 'STFT,' 'ISTFT,' 'Magnitude' and 'Phase' are used for simplicity, since the complex-valued discrete Fourier transform is used to initialize the weights of the 1-D convolution layers. All other layers are fully-connected (FC) with ELU [30] activations. Typically our output and target waveforms are smaller than the input; this difference in size (indicated by the cross-hatched region designated "lookback") means the autoencoder is 'asymmetric.' On the autoencoder output, the "magnitude spectrogram designated $\tilde{M}$ is used for regularizing the loss, Eq. (1).

(multiplicative) connection for the magnitude only [19]. These skip connections dramatically improve the speed of training, and can be viewed in three complementary ways: allowing information to propagate further through the network, smoothing the loss surface [32], and/or allowing the network to compute formidable perturbations subject to the goal of profiling an audio effect.

In the middle of the autoencoder, we concatenate values of the effect controls (*e.g.,* threshold, ratio, *etc.*) and "merge" these via an additional FC layer. The first layer of the autoencoder maps the number of time frames in the spectrograms to 64, with subsequent layers shrinking (or, on the output side, growing) this by factors of 2. The resulting model has approximately 4 million trainable parameters.

2.3 Training Procedure

We use a log-cosh loss function [33], which has similar properties to the MAE (*i.e.,* L1 norm divided by the number of elements) in the sense that it forces the predicted signal to closely follow the target signal at all times, however the roundness of the log-cosh function near zero allows for significantly better training at low loss values than does L1, which has a discontinuous derivative at zero.

Furthermore we include an L1 regularization term with a small coefficient λ (*e.g.,* 2e-5), consisting of the magnitude spectrogram $\tilde{M}$ from the output side of the autoencoder, weighted exponentially by frequency-bin number f_k to help reduce high-frequency noise in the predicted output. Thus the equation for the loss function is given by

$$\text{Loss} = \log\left[\cosh\left(\tilde{y} - y\right)\right] + \lambda \exp[(f_k)^{\alpha}] \cdot |\tilde{M}|_{L1}, \quad (1)$$

where $\tilde{y}$ and y are the predicted and target outputs, respectively, and choosing $\alpha = 1$ implies exponential weighting by frequency bin f_k, and choosing $\alpha = 0$ disables any such weighting.

Simply training on a large amount of musical audio files is not necessarily the most efficient way to train the network – depending on the type of effect being profiled, some signals may be more 'instructive' than others. A compressor requires numerous transients of significant size, whereas an echo (or 'delay') effect may train most efficiently on uncorrelated input signals (*e.g.,* white or pink noise). Therefore, we augment a dataset of music recordings with randomly-generated sounds intended to provide both dynamic range variation and broadband frequency coverage.

By virtue of the automation afforded by software effects such as Comp-4C, we can train *indefinitely* using randomly-synthesized signals which change during each iteration. But for the LA-2A, we created a large (20 GB) input dataset of public domain musical sounds and randomly-generated test sounds, concatenated these and divided the result into (unique) files of 15-minute duration, using a fixed increment of "5" on the LA-2A's Peak Reduction knob between recordings, for both settings of the Comp/Lim switch. For prerecorded (*i.e.,* non-synthesized) audio, windows from the input (and for ST data, target) data are copied from random locations in the audio files, along with the control settings used. Data augmentation is applied only in the form of randomly flipping the phase of inputs and targets.

To achieve the results in this paper, we trained for two days (see "Implementation," below) on what corresponded to approximately 2000 hours of audio sampled at 44.1 kHz (or 130 GB if it were stored on disk). As a performance metric, we keep a separate, fixed "validation set" of approximately 12 minutes of audio; all results displayed in this paper are for validation data, *i.e.,* on data which the network has not "seen" before.

The arrangement of this data is "maximally shuffled," *i.e.,* we find that training is significantly more smooth and stable when *the control settings are randomly changed for each data window within each mini-batch.* Trying to train using the same knob settings for large sequences of inputs – as one might expect to do by taking a lengthy pair of (input-output) audio clips obtained at one effect setting and breaking them up into a sequential mini-batch of training windows – results in unstable training in the sense that the (training and/or validation) loss varies much more erratically and, overall, decreases much more slowly than for the 'maximally shuffled' case in which one varies the knob settings with every window. This shuffling from window to window is another reason why our model is not autoregressive: because we wish to learn to model the controls with the effect.

When starting from scratch, weights are initialized randomly except for the weights connecting to the input and output layers convolutional layers, which are initialized using basis values corresponding to a Discrete Fourier Transform (DFT), and its inverse transform, respectively. These are subsequently allowed to evolve as training proceeds. For the different task of image classification on the ImageNet dataset [34], the combination of Adam [35] with weight decay [36] has been shown [37] to be among the fastest training methods available when combined with learning rate scheduling. We also adopt this combination for our problem.

An important feature, found to decrease both final validation loss values and the number of epochs required to reach them, is the use of learning rate scheduling, *i.e.,* adjusting the value of the learning rate dynamically during the course of gradient-based optimization, rather than keeping the learning rate static. We follow the “1-cycle” policy [38], which incorporates cosine annealing [39], in the manner popularized by the Fasti.ai team [40]. Compared to using a static learning rate, the 1-cycle policy allowed us to reach roughly 1/10th the error in 1/5 the time.

2.4 Implementation

The SignalTrain code was written in Python using the PyTorch [41] library along with Numba for speeding up certain subroutines. Development and training was primarily conducted on a desktop computer with two NVIDIA Titan X GPUs. Later in the project we upgraded to two RTX 2080Ti GPUs, which, with the benefit of NVIDIA’s “Apex” mixed-precision (MP) training library [5], yielded speedup of 1.8x over the earlier runs.

3 Results

3.1 Software Compressor: “Comp-4C”

We ported MATLAB code to Python for a single-band, hard-knee compressor with four controls: threshold, ratio, attack and release times [42]. (This compressor implements no side-chaining, make-up gain or other features.) As it is a software compressor, the training data could be generated “on the fly," choosing control (‘knob’) settings randomly according to some probability distribution (*e.g.,* uniform, or a beta distribution to emphasize the endpoints [6]). This synthesis allows for a virtually limitless size of the training dataset. Our early experiments used such a dataset, but given that intended goal of this system is to profile systems within a finite amount of time, and particularly *analog* effects which would typically require the creation of a finite set of recordings, we chose to emulate the intended use case for analog gear, namely a finite dataset in which the control knob settings are equally spaced, with 10 settings per control.

[5] `https://github.com/NVIDIA/apex`

[6] Initially we chose control settings according to a symmetric beta distribution, with the idea that by mildly emphasizing the endpoints of the control ranges, the model would learn more efficiently, however experience showed no performance enhancement compared to choosing from a uniform distribution.

Figure 3 shows the performance of the model compared to the target audio, for the case of a step-response, a common diagnostic signal for compressor performance [4, 8, 43]. The predicted values follow the target closely enough that we show their differences in Figure 4. Key differences occur at the discontinuities themselves (especially at low attack times), and we see that the predictions tend to “overshoot” slightly on at the release discontinuity (we speculate that this is due to slight errors in the phase in the spectral decomposition in the model), but that in between and after the discontinuities the predictions and target match closely.

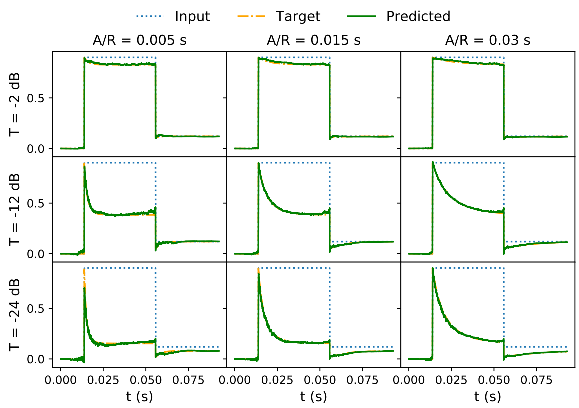

Fig. 3: Sample model step-response performance for the Comp-4C effect using WT data on a domain of 4096 samples at 44.1 kHz, for various values of threshold (T) and attack-release (A/R, set equivalently). In all graphs, the ratio=3. See Figure 4 for a plot of the difference between predicted and target outputs, and Supplemental Materials [26] for audio samples and an interactive demo with various input waveforms and adjustable parameters.

As noted in Section 2.1, the size of the lookback window can have an effect on the error bounds. Figure 5 shows that the loss on the Validation set to be consistent with estimates obtained for the cases depicted in Figure 1. And yet listening to these examples (see Supplemental Materials [26]) one notices noise in the predicted results, suggesting that the lookback window size (or “causality noise”) is not the only source of error in the model.

Although step responses are a useful diagnostic, the neural network model approximates the input-output mappings it is trained on, and is ultimately intended

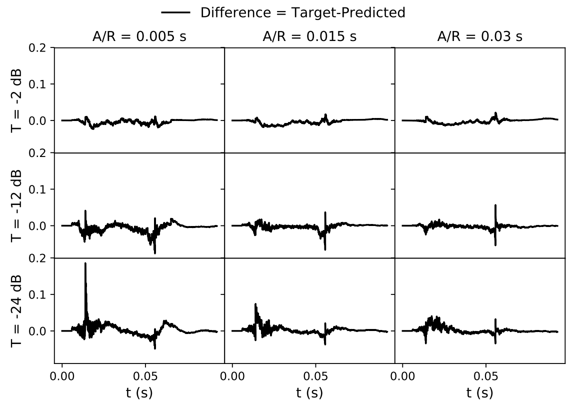

Fig. 4: The difference between predicted and target outputs for the step responses shown in Figure 3. We see the largest errors occur precisely at the step discontinuities, likely due to inadequate approximation in the "spectral" representation within the model.

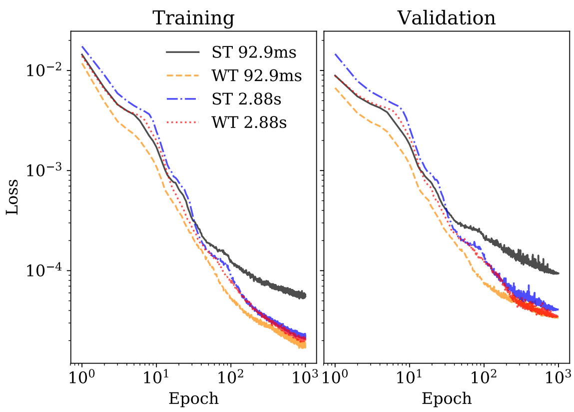

Fig. 5: Typical loss on Training & Validation sets for Comp-4C-Large effect, while training for ST and WT data, for two different lookback window lengths. (Because our data is randomly-sampled, "Epoch" does not refer to a complete pass through the dataset, but rather the arbitrary selection of 1000 mini-batches.)

for use with musical sounds which may typically lack such sharp discontinuities. Thus a comparison of compressor output for musical sounds is in order as well. Figure 6 shows a comparison of frequency spectrum for a full-band recording (*i.e.,* drums, bass, guitar, vocals) in the testing dataset. It also shows that scaling the L1 regularization exponentially by frequency can yield a reduction in high-frequency noise, sacrificing

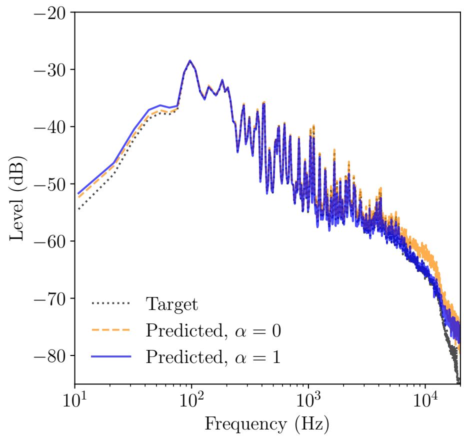

Fig. 6: Power spectra for musical audio in the Test dataset [26] compressed with Comp-4C control parameters [-30, 2.5, .002, .03]. Here we see the effects of weighting the L1 regularization in the loss function Eq. (1) exponentially by frequency ($\alpha = 1$) or not ($\alpha = 0$): weighting by frequency shifts a nontrivial amount of high frequency noise toward a proportionally small increase at low frequencies. Although noise is still clearly audible in both predicted outputs (refer to Supplemental Materials [26] to hear audio samples), the result is that the listener perceives less overall noise in the output when the frequency-weighted L1 regularization is used.

a proportionally smaller amount of accuracy at low frequencies.

3.2 Analog Compressor: LA-2A

A primary interest in the application our method is not for cases in which a software plugin already exists, but rather for the profiling of analog units. As an example, we choose the Universal Audio's Teletronix LA-2A, an electro-optical compressor-limiter [44], the controls for which consist of three knobs and one switch. Given that two of the knobs are only for input-output gain adjustment, for this study, we focus only on varying the "Peak Reduction" (PR) knob, and the "Compress/Limit" switch. The switch is treated like a knob, with limits of 0 for "Compress" to 1 for "Limit."

Figure 7 shows the loss on the validation set for the LA-2A for different lookback sizes. The dashed (black) line shows a model with an input size of 8192*2=16384

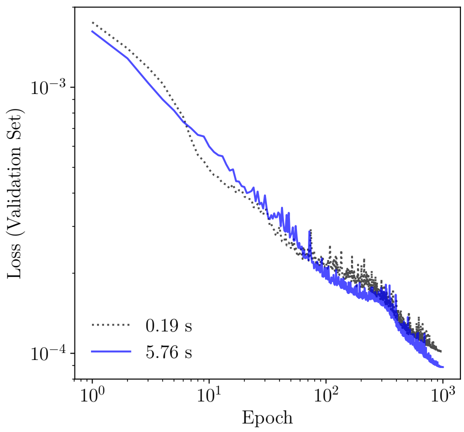

Fig. 7: Training history on the LA-2A dataset for different lookback sizes. The "kink" near epoch 300 is a common feature of the 1-cycle policy [38, 40] when using an "aggressive" learning rate (in this case, 7e-4). Both runs achieve comparable losses despite the longer lookback buffer needing nearly 5 times as much execution time.

and took 15 hours to run, the solid (blue) line is a model with input size of 8192*27=221184 and took 72 hours. Both runs used output sizes of 8192 samples. In all our runs, a loss value of approximately 1e-4 is achieved, regardless of the size of the model – even for a lookback extending beyond the "5 seconds for complete release" typically associated with the LA-2A [45]. This indicates that the finite size of the lookback window (or "causality noise") is not the primary source of error; this is consistent with the Comp-4C results (*e.g.,* see Figure 5). The primary source of error remains an ongoing subject of investigation. Graphs of example audio waveforms from the Testing dataset are shown in Figure 8, where it is noteworthy that the model will at times over-compress the onset of an attack as compared to the true LA2A target response.

4 Conclusion

In pursuit of the goal of capturing and modeling generic audio effects by means of artificial neural networks, we have focused this study on dynamic range compressors as a representative problem set because their nonlinear, time-dependent nature makes them a challenging class of problems, and because they are a class of effects

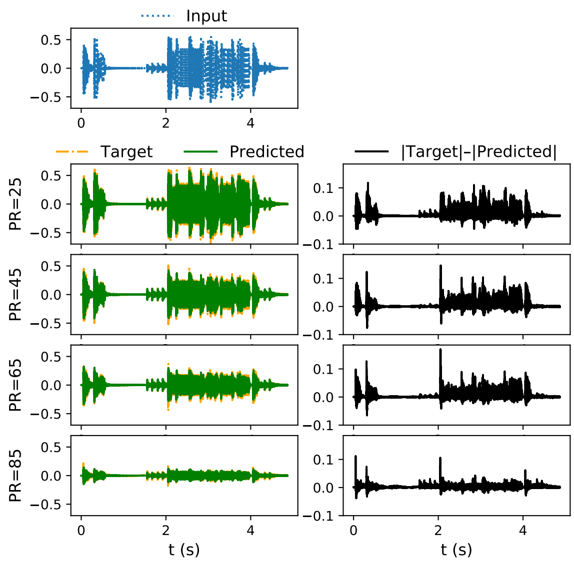

Fig. 8: Sample output for LA-2A using drum recordings from the Testing dataset, for various values of the Peak Reduction control. (The Comp/Lim switch setting had negligible effect on these outputs.) We see that the model's predictions typically slightly underestimate the target value for attack transients. Audio samples are available in Supplemental Materials[26].

of high interest in the field of musical audio production. Rather than rely on domain-specific knowledge of audio compressors in constructing our end-to-end system, our model learns the effects the parameterized controls in the process of training on a large dataset consisting of input-output audio pairs and the control settings used. We treat signals as wide-sense stationary for the sake of batch-based SGD optimization and we are aware that we miss some information due to that. That can be tackled by balancing the trade-off between time-frequency resolution. The results capture the qualities of the compressors sampled, although the speed of execution and the residual noise in the neural network output suggest that practical implementations of this method may await improvements in computer implementation and refinements to the model. We are interested in trying a model based on WaveNet [27, 46] or WaveRNN[47] for comparisons to our model regarding speed and accuracy. As the intent of this effort are the modeling of effects in general, more work remains to probe the limits of our method toward a variety of other signal processing effects, both analog and digital, as well for the construction of new effects by solving "inverse problems" such as de-compression [48].

References

[1] Smith, J. O., *Physical Audio Signal Processing: For Virtual Musical Instruments and Audio Effects*, W3K Publishing, 2011.

[2] Välimäki, V., Parker, J., Savioja, L., Smith, J., and S Abel, J., "More Than 50 Years of Artificial Reverberation," 2016.

[3] Yeh, D. T., Abel, J. S., Vladimirescu, A., and Smith, J. O., "Numerical Methods for Simulation of Guitar Distortion Circuits," *Computer Music Journal*, 32(2), pp. 23–42, 2008.

[4] Eichas, F., Gerat, E., and Zölzer, U., "Virtual Analog Modeling of Dynamic Range Compression Systems," in *Audio Engineering Society Convention 142*, 2017.

[5] Stikvoort, E. F., "Digital Dynamic Range Compressor for Audio," *J. Audio Eng. Soc*, 34(1/2), pp. 3–9, 1986.

[6] Kröning, O., Dempwolf, K., and Zölzer, U., "Analysis and Simulation of an Analog Guitar Compressor," in *14th Interational Conference on Digital Audio Effects (DAFx-11)*, 2011.

[7] Gerat, E., Eichas, F., and Zölzer, U., "Virtual Analog Modeling of a UREI 1176LN Dynamic Range Control System," in *Audio Engineering Society Convention 143*, 2017.

[8] Simmer, U., Schmidt, D., and Bitzer, J., "Parameter Estimation of Dynamic Range Compressors: Models, Procedures and Test Signals," in *Audio Engineering Society Convention 120*, 2006.

[9] Choi, K., Fazekas, G., Sandler, M., and Cho, K., "Convolutional recurrent neural networks for music classification," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2392–2396, IEEE, 2017.

[10] Schlüter, J. and Böck, S., "Improved musical onset detection with convolutional neural networks," in *2014 ieee international conference on acoustics, speech and signal processing (icassp)*, pp. 6979–6983, IEEE, 2014.

[11] Huang, P.-S., Kim, M., Hasegawa-Johnson, M., and Smaragdis, P., "Joint optimization of masks and deep recurrent neural networks for monaural source separation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(12), pp. 2136–2147, 2015.

[12] Stowell, D., Giannoulis, D., Benetos, E., Lagrange, M., and Plumbley, M. D., "Detection and classification of acoustic scenes and events," *IEEE Transactions on Multimedia*, 17(10), pp. 1733–1746, 2015.

[13] Arifianto, D. and Farid, M. N., "Dereverberation binaural source separation using deep learning," *The Journal of the Acoustical Society of America*, 144(3), pp. 1684–1684, 2018.

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

[15] Pons, J., Janer, J., Rode, T., and Nogueira, W., "Remixing music using source separation algorithms to improve the musical experience of cochlear implant users," *The Journal of the Acoustical Society of America*, 140(6), pp. 4338–4349, 2016.

[16] Engel, J., Resnick, C., Roberts, A., Dieleman, S., Eck, D., Simonyan, K., and Norouzi, M., "Neural Audio Synthesis of Musical Notes with Wavenet Autoencoders," *arXiv preprint arXiv:1704.01279*, 2017.

[17] Wang, Y., Skerry-Ryan, R. J., Stanton, D., Wu, Y., Weiss, R. J., Jaitly, N., Yang, Z., Xiao, Y., Chen, Z., Bengio, S., Le, Q. V., Agiomyrgiannakis, Y., Clark, R., and Saurous, R. A., "Tacotron: Towards End-to-End Speech Synthesis," in *INTERSPEECH*, 2017.

[18] Donahue, C., McAuley, J., and Puckette, M., "Adversarial Audio Synthesis," *arXiv preprint arXiv:1802.04208*, 2018.

[19] Mimilakis, S. I., Drossos, K., Virtanen, T., and Schuller, G., "Deep neural networks for dynamic range compression in mastering applications," in *Audio Engineering Society Convention 140*, Audio Engineering Society, 2016.

[20] Damskägg, E.-P., Juvela, L., Thuillier, E., and Välimäki, V., "Deep Learning for Tube Amplifier Emulation," *arXiv preprint arXiv:1811.00334*, 2018.

[21] Damskägg, E.-P., Juvela, L., and Välimäki, V., "Black-Box Modeling of Audio Distortion Circuits with Deep Learning," , submitted to SMC.

[22] Martínez, M. and Reiss, J. D., "Modeling of Nonlinear Audio Effects with End-to-End Deep Neural Networks," 2018.

[23] Dieleman, S. and Schrauwen, B., "End-to-end learning for music audio," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6964–6968, IEEE, 2014.

[24] Lluís, F., Pons, J., and Serra, X., "End-to-end music source separation: is it possible in the waveform domain?" *arXiv preprint arXiv:1810.12187*, 2018.

[25] Sheng, D. and Fazekas, G., "A Feature Learning Siamese Model for Intelligent Control of the Dynamic Range Compressor," .

[26] Supplementary Materials, including audio clips and an interactive demo, are available at a website created for this paper: `http://www.signaltrain.ml`.

[27] Van Den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A. W., and Kavukcuoglu, K., "WaveNet: A Generative Model for Raw Audio." in *SSW*, p. 125, 2016.

[28] Ronneberger, O., Fischer, P., and Brox, T., "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241, Springer, 2015.

[29] Yian Lim, T., A. Yeh, R., Xu, Y., N. Do, M., and Hasegawa-Johnson, M., "Time-Frequency Networks for Audio Super-Resolution," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 646–650, 2018.

[30] Clevert, D.-A., Unterthiner, T., and Hochreiter, S., "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)," *ICLR 2016*, 2016.

[31] 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*, pp. 770–778, 2016.

[32] Li, H., Xu, Z., Taylor, G., and Goldstein, T., "Visualizing the Loss Landscape of Neural Nets," *CoRR*, abs/1712.09913, 2017.

[33] Chen, P., Chen, G., and Zhang, S., "Log Hyperbolic Cosine Loss Improves Variational Auto-Encoder," Submitted to ICML.

[34] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L., "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

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

[36] Krogh, A. and Hertz, J. A., "A simple weight decay can improve generalization," in *Advances in neural information processing systems*, pp. 950–957, 1992.

[37] Gugger, S. and Howard, J., "AdamW and Super-Convergence Is Now the Fastest Way to Train Neural Nets," , last accessed 19 November 2018.

[38] Smith, L. N., "A disciplined approach to neural network hyper-parameters: Part 1 - learning rate, batch size, momentum, and weight decay," *CoRR*, abs/1803.09820, 2018.

[39] Loshchilov, I. and Hutter, F., "SGDR: Stochastic Gradient Descent with Restarts," *CoRR*, abs/1608.03983, 2016.

[40] Gugger, S. and Howard, J. P., "Callbacks.One_cycle | Fastai," , [Online; accessed 25-May-2019].

[41] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A., "Automatic differentiation in PyTorch," in *NIPS 2017*, 2017.

[42] Tarr, E., *Hack Audio (Audio Engineering Society Presents): An Introduction to Computer Programming and Digital Signal Processing in MATLAB*, Routledge, 2018.

[43] Giannoulis, D., Massberg, M., and Reiss, J. D., "Digital Dynamic Range Compressor Design—A Tutorial and Analysis," *J. Audio Eng. Soc*, 60(6), pp. 399–408, 2012.

[44] Lawrence, J. F., "An improved method of audio level control for broadcasting and recording," *Journal of the SMPTE*, 73(8), pp. 661–663, 1964.

[45] "Product Catalog - Universal Audio," , [Online; accessed 01-March-2019].

[46] Rethage, D., Pons, J., and Serra, X., "A Wavenet for Speech Denoising," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5069–5073, IEEE, 2018.

[47] Kalchbrenner, N., Elsen, E., Simonyan, K., Noury, S., Casagrande, N., Lockhart, E., Stimberg, F., van den Oord, A., Dieleman, S., and Kavukcuoglu, K., "Efficient Neural Audio Synthesis," *CoRR*, abs/1802.08435, 2018.

[48] Gorlow, S. and Reiss, J. D., "Model-based inversion of dynamic range compression," *IEEE Transactions on Audio, Speech, and Language Processing*, 21(7), pp. 1434–1444, 2013.