DEEP LEARNING CONDITIONED MODELING OF OPTICAL COMPRESSION

Riccardo Simionato and Stefano Fasciani

Department of Musicology
University of Oslo
Oslo, Norway
riccardo.simionato@imv.uio.no | stefano.fasciani@imv.uio.no

ABSTRACT

Deep learning models applied to raw audio are rapidly gaining relevance in modeling audio analog devices. This paper investigates the use of different deep architectures for modeling audio optical compression. The models use as input and produce as output raw audio samples at audio rate, and it works with no- or small-input buffers allowing a theoretical real-time and low-latency implementation. In this study, two compressor parameters, the *ratio*, and *threshold* have been included in the modeling process aiming to condition the inference of the trained network. Deep learning architectures are compared to model an all-tube optical mono compressor including feed-forward, recurrent, and encoder-decoder models. The results of this study show that feed-forward and long short-term memory architectures present limitations in modeling the triggering phase of the compressor, performing well only on the sustained phase. On the other hand, encoder-decoder models outperform other architectures in replicating the overall compression process, but they overpredict the energy of high-frequency components.

1. INTRODUCTION

Virtual Analog (VA) modeling aims to emulate in the digital domain electrical or electro-mechanical musical devices. This field has a long history, and so far several digital models for different types of analog devices have been proposed [1]. The nonlinearities in circuits or mechanics of these devices determine their unique and appealing sonic characteristics. Such nonlinearities are challenging to model because they need to be emulated with digital signal processing algorithms.

Digital models are categorized as "white-box" or "black-box". The first category is based on the simulation of the system's components by discretizing differential equations to generate a numerical solution. A "black-box" model replicates the system's response by observing the input-output behavior to estimate the model's inner parameters. Examples of the "white-box" approach for VA are found in [2, 3], while "black-box" models are used in [4, 5].

Recently, studies based on deep learning techniques have shown promising results in model audio systems, supported by optimized libraries and modern GPU-equipped workstations. Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) are the predominant techniques in this field. WaveNet is among the most used architectures [6], which is a fully probabilistic and auto-regressive deep neural network using a stack of convolutional layers to model the conditional probability distribution. Several modifications of WaveNet have been proposed [7, 8]. These architectures have also been employed to model nonlinearities in circuits of analog audio effects. In particular, a feed-forward variant of WaveNet has been used with a vacuum-tube amplifier [9] and distortion pedals [10]. RNNs have been used for the same task as well [11], and in some cases, the implementation of the inference part satisfies the real-time constraints [12, 13]. Another interesting approach is described in [14], where a hybrid model with an adaptive front-end, followed by a latent-space, and a synthesis back-end has been investigated to achieve a general-purpose deep learning model for audio effects. This has been applied to different types of analog effects, with both short-term and long-term memory, including plate and spring reverberators [15]. Lastly, simpler architectures, such as feed-forward deep neural networks have been explored as well. In [16], a feed-forward deep neural network embedded within a discrete-time state-space system has been proposed to model guitar distortion circuits and a low-pass filter. In this work, the authors used measurements of inner signals within the circuits to learn the trajectory of the system, which represents the set of points in the state space that are the future states resulting from a given initial state.

This paper follows up VA investigation using deep learning techniques and addresses the case of an optical compressor. Compression is a particular type of audio processing that reduces the dynamic of the signal, taming the volume of loud sounds and amplifying quiet sounds according to a specific temporal profile. An earlier attempt to model an analog optical leveling amplifier has been presented in [17], where the authors take a frequency domain approach and use large artificial neural networks working on relatively long segments of the input signal. A more recent example exhibits real-time modeling of the same device [18] by exploiting a modified version of a Temporal Convolutional Network (TCN) [19]. In this paper we also explore the conditioning of the model against two control parameters of the device, allowing the trained network to apply different types of compression. Previous works are mostly focused on modeling for a static representation of the audio effect (i.e. fixed parameters), while a single variable control parameter has been considered in [9, 13, 17] and two variable control parameters only in [18]. The conditioning is realized by feeding the network with extra inputs. In this study, we follow a similar approach proposed in [9, 13, 17] but we scale up the problem to two variable parameters. In addition, as noted in [13], deep networks can suffer from aliasing-like effects. For this reason, we also investigated how our models are affected by the same problem.

The rest of the paper is organized as follows. Section 2 details the specific compressor device. An overview of the different architectures used in this study is presented in Section 3. Dataset, ex-

periments and, measurements are described in Section 4. Section 5 provides an objective evaluation of the results and the comparison between the different architectures. Finally, Section 6 concludes this paper by discussing open challenges and future directions.

2. DEVICE UNDER STUDY

A dynamic range compressor is an automatic volume control: it attenuates the amplitude of an audio signal by a given amount when this exceeds a given threshold. Its functioning is regulated by the *threshold* parameter, the point above which the compressor starts to attenuate the input signal, and the *ratio* parameter, the amount of compression applied. Compressors present two additional parameters, namely the *attack* and *release*, which determine the time it takes to apply and remove the attenuation.

The compressor we chose for this study is the `CL 1B`[1], the popular unit manufactured by `TUBE-TECH`. Originally introduced in 1991, the `CL 1B` is tube-based and optical, where the audio signal feeds a lighting element that in turn illuminates a light-sensitive resistor in the compression circuit. The resistance affects the compression circuit determining how much and how quickly to attenuate the incoming audio signal. The `CL 1B` presents an output tube-based push-pull amplifier with variable gain, which is used to add harmonic distortion after the compression stage and not to limit the dynamics. The unit also features an input amplifier whose gain can be tuned only internally for calibration purposes. In this work, we condition the CL 1B model only against variations of the *threshold* and *ratio* parameters. The remaining are fixed, as detailed in Section 4. With these settings, the trained model captures the leveling amplifier component of the `CL 1B` compressor.

3. DEEP LEARNING NETWORK ARCHITECTURES

We investigate several architectures to model the `CL 1B` compressor. In particular, we use fully connected and Long Short-Term Memory (LSTM) layers with a sequential architecture as well as with an encoder-decoder configuration.

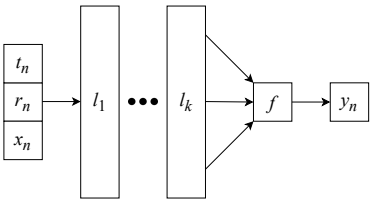


Figure 1: *Generic architectures for the feed-forward and recurrent neural networks. l represents the fully connected or LSTM hidden layer for the feed-forward and recurrent model, respectively. In both cases, the last layer indicated as f, is a fully connected unit with a single neuron. x_n, t_n and r_n represent the input, threshold, and ratio value at time-step n.*

3.1. Feed-forward models

Feed-forward models are relatively simple neural networks, as illustrated in Figure 1, consisting of only k fully connected layers with u neurons. Neurons in each layer can include an activation function. When the activation function is absent, the layer performs an affine transformation. In our experiments, these networks are fed with a single input audio sample and predict a single audio output sample at each time step. Hence, the output layer presents a single neuron only. The model can be represented with the following equation:

$$\hat{y}_n = f^*(x_n; \theta), \tag{1}$$

which defines a mapping where y_n is the predicted output sample, x_n is the input sample, θ is the parameters the network learns, and f^* the approximated function. In the rest of this paper, we refer to the model as 'FF'.

3.2. Recurrent models

For the recurrent models, we select gated RNN-based models, in particular LSTM [20]. In this case, the networks' architecture consists of k LSTM layers, including u units, and a final fully connected layer with a single neuron. LSTMs partially solve some of the shortcomings of vanilla RNNs when modeling long sequences, such as the vanishing gradient problem [21]. LSTMs featuring a gated recurrent unit with skip-connections allow gradients to flow across many time-steps. The LSTM model is shown in Figure 1. This time in Eq. 1 has to be added the recurrent unit's state at the previous time step s_{n-1}, becoming

$$\hat{y}_n = f^*(x_n; s_{n-1}; \theta). \tag{2}$$

The unit's state consists of two vectors, the cell state, c, and the hidden state, h. At each time-step, the current time-step input, x_n, the initial cell state, c_{n-1}, and the initial hidden state, h_{n-1} represent the inputs. The LSTM then produces the updated hidden state, h_n, and the updated cell state, c_n, as the outputs.

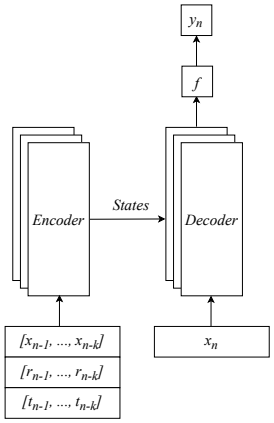


Figure 2: *ED model architecture. Encoder and Decoder can be stacked layers, and f represents the final fully-connected layer with a single neuron and followed by a nonlinear activation function. t and r stand as threshold and ratio values used as conditioning for the network. The internal states of the encoder, computed with k past samples of the input sequence, act as conditioning for the decoder. The decoder is trained to predict the output sample given the input sample at the current time-step and conditioned by the internal state computed by the encoder.*

[1] `http://www.tube-tech.com/cl-1b-opto-compressor/`

3.3. Encoder-decoder models

The third architecture is designed in an encoder-decoder fashion, following the architectures for sequence modeling tasks [22]. In the sequence-to-sequence model, the encoder processes n past sample of the input sequence and returns its final internal state and output. The output is discarded while the internal state is passed to the decoder as a sort of conditioning. Specifically, the final state vector and output from the encoder set the first internal state of the decoder. The decoder then learns to predict the target at each time-step, given m past values of the true output ($[n-1, ..., n-m]$). The decoder is trained to predict the target signal at the next time-step, given the previous outputs. The architecture of the decoder, as shown in Fig. 2, presents single or multiple stacked LSTM layers, ending with a fully connected layer including a single neuron. This architecture exploits the so-called "teacher forcing", a method for training recurrent neural network models that use the ground truth from prior time-steps as input. In this case, a separate auto-regressive setup for inference has to be implemented.

Compared to the sequence-to-sequence model, our proposed architecture presents some key changes. The decoder was conditioned on the input sequence but without including the sample at the current time-step. This sample was instead the input for the decoder, which was trained to predict the output given only the samples from the input sequence. In this way, the encoder passes the internal state computed from past time-step samples and the decoder learns to predict the target at each time-step, given the input sample at the current step. In this case, a separate inference model is not needed since the networks are not trained with the ground truth. In the rest of this paper, we refer to the model as 'ED'.

3.4. Conditioning

In the case of FF and LSTM models, *threshold* and *ratio* values were added to the input vector to have an input vector with three values: $u_n = [x_n, t_n, r_n]$, as illustrated in Fig. 1. On the other hand, in the case of encoder-decoder architecture, *threshold* and *ratio* were included in the encoder input. Since in the encoder, past samples were also used in the prediction process, the input becomes a kx3 dimensional matrix with k representing the number of past samples considered. The decoder layer was fed with only the input values at current time step. In all cases, before to fed the networks, the conditioning values were normalized between [0,1].

4. EXPERIMENTAL STUDIES

We carried out experiments with all architectures detailed in the previous section, which have been implemented in Python using Tensorflow. For the training, we used an Adam optimizer with 0.001 as the initial learning rate. The Mean Squared Error (MSE) has been selected as the loss function. To select the best configuration for each architecture, we used Optuna [23], a framework for the automated search of optimal hyper-parameters. Regarding ED models, we experimented with changing the size of the input raw audio input vector, appending a variable number of past input samples to the current input sample. In particular, we explored $[2, 4, 8, 16]$ as input vector size, which are relatively small values that can provide a low-latency response in a theoretical real-time implementation. We have also explored different sizes for each architecture, considering 8, 16, 32, 64, or 128 numbers of hidden units and 1 or 2 layers. For the activation function, we experimented with the $sigmoid$ and $tanh$, since their S-shaped characteristic curves are close to the nonlinear saturation profiles usually found in analog circuits. For each combination of the hyper-parameters listed above, we trained the various architectures for 20 epochs. Results on the test set were evaluated quantitatively by observing the MSE and by graphically analysing the predicted waveforms against the target ones.

The three final models (best FF, best LSTM, and best ED) have been trained for 100 epochs with a batch size of 128, using an early stopping condition in case performance does not improve after 20 epochs. The models were evaluated against three different settings of the parameters, representing gentle (*threshold* = -10 dBU, *ratio* = 4.66:1), medium (*threshold* = -30 dBU, *ratio* = 3.33:1), and heavy (*threshold* = -40 dBU, *ratio* 7.33:1) compression. Medium compression conditioning values were not seen by the networks during training. The architectures of the final models and the associated results are detailed in section 5.

4.1. Dataset

The Dataset has been collected from the `CL 1B` by feeding its input with a selection of audio signals and simultaneously recording its output. A MOTU M4 audio interface[2] was used for this purpose. The input is a mono audio signal which has an overall duration of 332 s and it includes a sequence of frequency sweeps (ranging from 20-Hz to 20 kHz), white noises with increasing amplitude (linear and logarithmic ramp), guitar, bass, and drums recordings (loop and single notes). Sweeps and white noises have different lengths, specifically 0.2, 0.4, 0.8, and 1.6 s. Guitar, bass, and drums are taken from Fraunhofer IDMT datasets[3] and they were included with a variety of amplitudes. Since the compressor affects and depends on the dynamics of the sounds, this selection includes signals with a wide variety of amplitude levels and temporal envelopes. The left input channel of the audio interface was connected to the left output channel of the interface itself. The right input channel of the audio interface was connected to the output of the compressor, and the input of the compressor was connected to the right output channel of the audio interface. This allows recording effectively both compressor's input and output signals compensating for the minor sound coloring and latency of the audio interface. The audio data was recorded at a sampling rate of 96 kHz and then downsampled to 48 kHz for training the models. A rate of 48 kHz provides a good trade-off between audio quality and computational requirements for training and inference of the considered models. Before starting the recording session, the levels of the audio interface inputs were matched, keeping the settings of the compressor in such a way that it was not triggered by the input signal, and then calibrated by adjusting the input gains of the audio interface. Finally, the output signals were recorded with different values for the *threshold* and *ratio* parameters. For the *ratio*, the output signal was recorded at 7 equally spaced values spanning from 2:1 to 10:1. Attack and release times were both fixed at 0.5 ms and 50 ms, respectively. These are fairly fast attack and release times. Considering the sampling rate of 48 kHz, we have approximately 24 samples for the attack phase and 2400 samples for the release. This resulted in limiting the delay between reaching the *threshold* and triggering the compression, and in turn, the

[2] `https://motu.com/en-us/products/m-series/m4/`

[3] `https://www.idmt.fraunhofer.de/en/publications/datasets.html`

temporal dependencies that the network must learn. For the *threshold*, we recorded the output with values spaced by 10 dBU starting from -10 dBU to -40 dBU. In total, we have 28 permutations of *ratio* and *threshold* values, for a total recording time of 2.58 hours. Finally, the compressor output gain was fixed at 0 dBU.

4.1.1. Training, Validation & Test sets

The recordings with the *ratio* set to 3.33:1 and *threshold* set to -30 dBU are used only for testing the model and are not included in the training and validation sets. In this way, we had a portion of the dataset with parameter values not seen in the training phase, which allows one to further test the capability of the model to correctly predict the output using unseen conditioning parameters. In the same way, a portion of sounds of 32 s, including guitar, bass, drum kick, hi-hat, and sweep, with different values of *ratio* and *threshold* has also been used exclusively for testing the generalization capability of the networks. The selected sounds present different amplitudes as well. However, all of them are within the reach of the minimum (-10 dBU) threshold activating the compressor. Finally, the models have been trained on 85% of the remaining dataset, while 15% served for validation.

5. RESULTS & DISCUSSION

In this section, we detail the results achieved by training the architectures described in Section 3, commenting on their performance and limitations. Generally, regarding the FF and LSTM models, we have not observed particular improvements when using the $tanh$ activation function in the last hidden and output layer with respect to $sigmoid$ (0.56% of test error reduction), but we kept this option for the later experiments since the training converges faster. On the other hand, with the ED model, the $sigmoid$ provided better performance (34, 71% of test error reduction).

Experiments on FF models show that they perform best with 2 layers with 32 units each. The validation and test errors for the different numbers of hidden units are detailed in Tab. 1. The activation function ('tanh') was added in the last hidden layer (l_k in Fig. 1). This configuration determined an improvement of 80.62% in the test error compared to the case of 'tanh' in the output layer and 82.82% compared to the use of only linear activation functions. FF models, as visible in Fig. 4 (b), appear to learn well the sustained dynamic of the output sounds, hence the amount of compression to be applied. However, they fall short in predicting the attack phase of the compression, in some cases operating as the compression is applied instantaneously. This behavior was expected because FF networks do not embed information about the past input signal and it is more challenging for them to capture the time dependencies, which are crucial for modeling compression. Therefore FF models present evident limitations when predicting output for impulsive input signals. This behavior is visible in the heavy-compression example in Fig. 3, where the target waveforms represent the real output of the compressor. The predicted signal is a compressed drum kick sound. The initial 'click' is not well predicted, while the rest is fairly accurate. This behaviour generates lower energy at high frequencies, as visible in the associated power spectrum, where the accuracy drops for components higher than 1 kHz. Other examples are visible in Fig. 4. Sounds as hi-hat and guitar are better predicted by analyzing the frequency representation of the signal, while in the time domain we can still see limitations on the attack phase prediction. For the bass signal, the models predict instead slightly larger values in the initial part, resulting in additional energy at high-frequency components.

Table 1: *Validation and Test losses for the two layers FF and LSTM models, and ED model (16 samples of window length) against the number of hidden units. Test conditioning loss refers to tests with compression settings never seen by the networks during training. Test loss instead refers to the test with all the other parameter combinations seen during the training. In both cases, the test signals include sounds not used to train the models.*

FF model			
# units	Val loss	Test conditioning loss	Test loss
8	$2.0858e^{-5}$	$1.8236e^{-5}$	$2.8656e^{-4}$
16	$2.0047e^{-5}$	$1.2695e^{-5}$	$2.9173e^{-4}$
32	$\mathbf{1.8755}e^{-5}$	$\mathbf{1.2208}e^{-5}$	$\mathbf{2.5381}e^{-4}$
64	$2.1747e^{-5}$	$1.8929e^{-5}$	$2.8372e^{-4}$
128	$2.0411e^{-5}$	$1.2868e^{-5}$	$2.8080e^{-4}$
LSTM model			
# units	Val loss	Test conditioning loss	Test loss
8	$1.5943e^{-5}$	$1.0857e^{-5}$	$2.6795e^{-4}$
16	$1.5811e^{-5}$	$1.0975e^{-5}$	$2.4218e^{-4}$
32	$\mathbf{1.5672}e^{-5}$	$\mathbf{1.0689}e^{-5}$	$\mathbf{2.4300}e^{-4}$
64	$1.5873e^{-5}$	$1.0916e^{-5}$	$2.4684e^{-4}$
128	$1.6568e^{-5}$	$1.2076e^{-5}$	$2.4371e^{-4}$
ED model			
# units	Val loss	Test conditioning loss	Test loss
8	$6.5702e^{-6}$	$4.7197e^{-6}$	$1.2214e^{-4}$
16	$5.2088e^{-6}$	$3.8354e^{-6}$	$1.0342e^{-4}$
32	$4.1773e^{-6}$	$3.2147e^{-6}$	$0.6228e^{-4}$
64	$\mathbf{3.9449}e^{-6}$	$\mathbf{3.0945}e^{-6}$	$\mathbf{0.6207}e^{-4}$
128	$4.0206e^{-6}$	$3.1645e^{-6}$	$0.6222e^{-4}$

Similar to the FF models, also LSTMs perform better with 2 layers and 32 units. Tab. 1 summarizes the errors for different numbers of hidden units. In this case, the 'tanh' activation function is used in the output layer. This determined a 7.44% reduction in the test error compared to the use of the linear activation function and 3.16% compared to the use of 'tanh' in the last hidden layer only. In Fig. 5(c) is visible how they present almost the same limitations as the FF model. LSTM models match accurately the amount of the compression during the sustained part but do not show particular improvements in predicting the initial transient phase of the sound. Therefore, performance on impulsive signals is overall better than FF models, although overall performance is very close to the FF case, with slightly lower losses, as visible in Figs. 3 and 5. This can be explained by the fact that LSTMs are able to take into account past information and hence learn the temporal dependencies during the attack phase with a higher accuracy.

Encoder-decoder models present a clear performance improvement compared to previous architectures. The encoder takes into account past samples of the input signal, helping the networks to learn the attack and release compression profiles. Losses for different numbers of past input samples given as input to the network are detailed in Tab. 2. Using an input size of 16 samples provided the best performance. This result is reasonable and expected, since increasing the size turns in greater information regarding the past of the signal. The best number of units per layer is 64, as can be seen in Tab. 1. We kept one layer in both encoder and decoder for two reasons: to limit the size and to compare against other

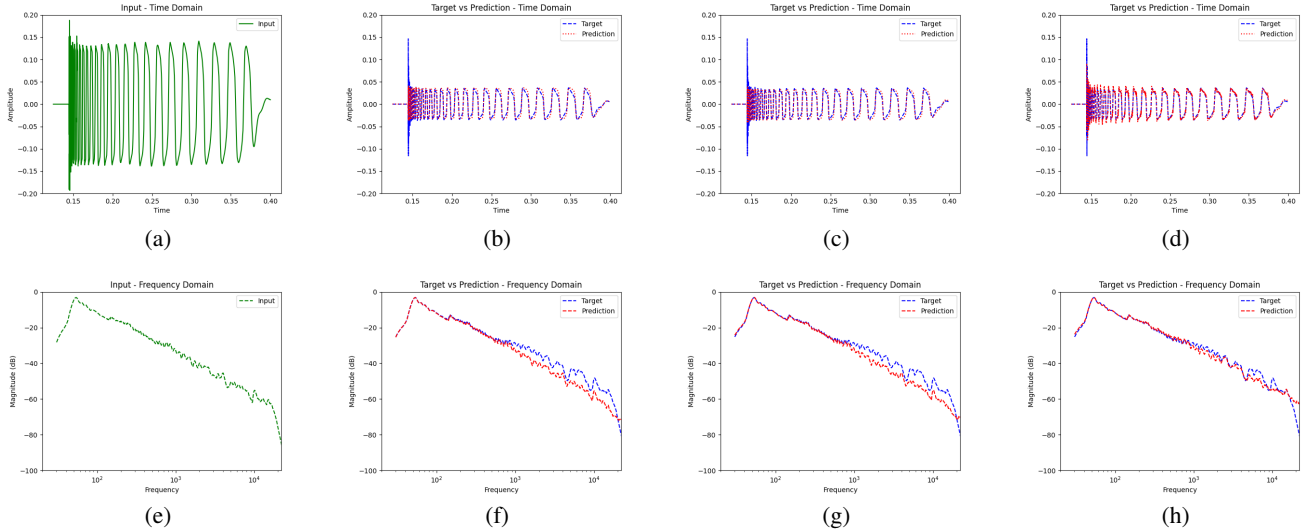


Figure 3: *Prediction and target output for a kick drum input signal (a) and relative power spectra of the entire signal (e) for each architecture: FF model (b) (f), LSTM (c) (g), ED (d) (h).*

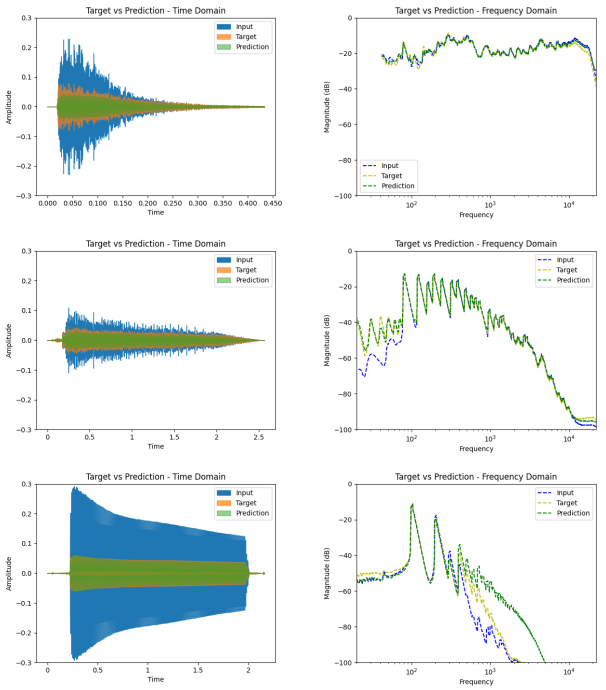


Figure 4: *Input, prediction, and target outputs for hi-hat (top), guitar [E1] (middle), and bass [A1] (bottom) and the relative power spectra of the entire signal using the best FF model.*

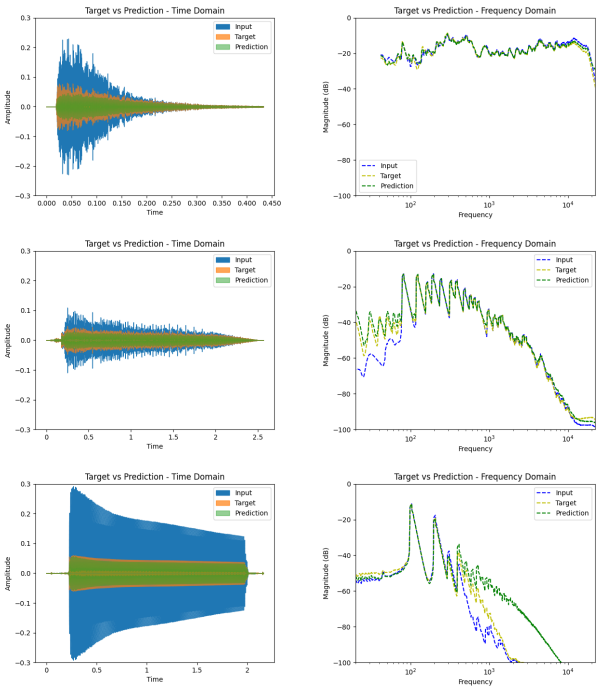


Figure 5: *Input, prediction, and target outputs for hi-hat (top), guitar [E1] (middle), and bass [A1] (bottom) and relative power spectra of the entire signal using the best LSTM model.*

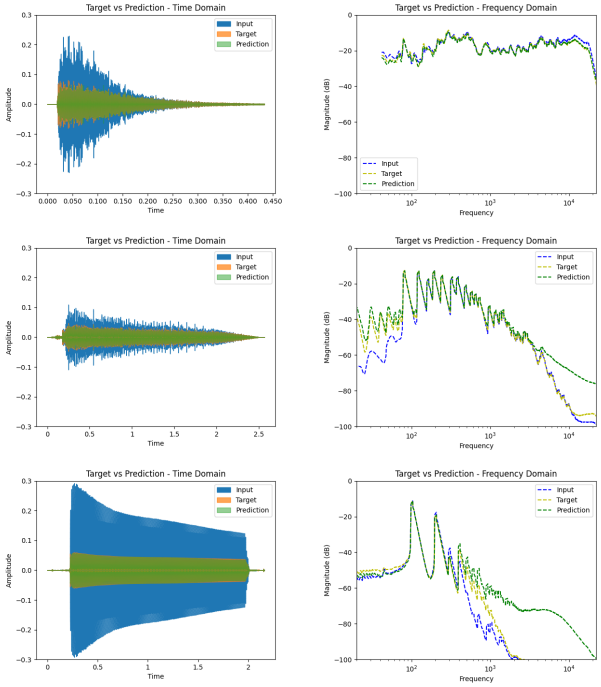


Figure 6: *Input, prediction, and target outputs for hi-hat (top), guitar [E1] (middle), and bass [A1] (bottom) and relative power spectra of the entire signal using the best ED model.*

architecture with the same overall number of layers. In general, considering the frequency response, the encoder-decoder architectures show limitations in modeling accurately the high frequency response, especially when the input is a low-frequency signal, such as the drum kick and bass, as can be seen in Figs. 3 and 6. For these signals, the energy of the high-frequency component is relatively low, and inaccurate predictions in such a portion of the spectrum are not sufficiently penalized by the loss function. However, the encoder-decoder approach shows the best accuracy and learning capability in predicting the attack and release phases, as can be seen in Fig. 3.

Table 3 details the Error-to-Signal Ratio (ESR) obtained using the three best models over four different types of input sounds: drum kick, hi-hat, guitar, and bass. The associated best models' configurations are reported in Tab. 5. To further assess the conditioning capability of these models, we computed setting the *threshold* to -30 dBU and the *ratio* to 3.33:1. These values were not included in the dataset used for training the network.

Given the vast difference between the number of parameters, we also compared the performance across the models keeping the capacities roughly similar (13k number of parameters). These results confirm the previous quantitative evaluation and for space reasons, we report the computed test loss only. The test losses are $2.8100e^{-4}$ for the FF model with $12,703$ parameters (two layers with 110 units each), $2.4300e^{-4}$ for the LSTM model with $12,961$ parameters (two layers with 32 units each), and $0.6437e^{-4}$ for ED model with $12,761$ parameters (one encoder and one decoder layer with 38 units each).

Listening to the predicted signals, the models can be ranked differently than the quantitative evaluations based on the ESR. In

Table 2: *Validation and Test losses for the ED models against the number of the input window length. The results refer to models with encoder and decoder consisting of one layer and* 8 *neurons,* 0.001 *learning rate, and 'sigmoid' activation function in the output layer. Test conditioning loss refers to tests with compression settings never seen by the networks during training. Test loss instead refers to the test with all the other parameter combinations seen during the training. In both cases, the test signals include sounds not used to train the models.*

Input size	Validation loss	Test cond. loss	Test loss
2	$1.3356e^{-5}$	$9.2715e^{-6}$	$1.2173e^{-3}$
4	$8.8111e^{-6}$	$6.3076e^{-6}$	$2.1775e^{-4}$
8	$8.8367e^{-6}$	$7.4624e^{-6}$	$1.2916e^{-4}$
16	$\mathbf{6.5702}e^{-6}$	$\mathbf{4.7197}e^{-6}$	$\mathbf{1.2214}e^{-4}$

Table 3: *ESR values for the best models predict the compressor output for four different input sounds: drum kick, hi-hat, guitar, and bass. These samples and the threshold (-30 dBU) and ratio (3.33:1) values used for conditioning have not been included in the sets during the training.*

$r = 3.33{:}1, t = -30$	Kick	Hi-Hat	Guitar	Bass
FF	0.0691	**0.0730**	0.0452	0.0552
LSTM	0.0567	0.0732	0.0361	0.0629
ED	**0.0273**	1.3445	**0.0203**	**0.0375**

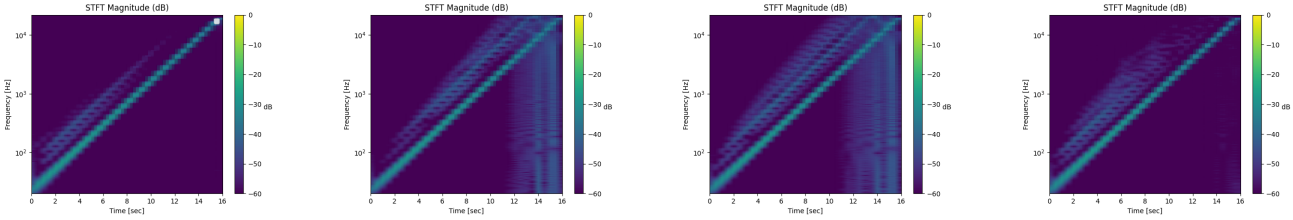
particular, the FF model appears to perform better than LSTM, which resulted in slightly more noisy. The ED model predictions are generally worse than the previous two models due to the noise at high frequencies. In general, hi-hat predictions are all perceptually closest and similar to the original one, likely due to their noisy sonic characteristics.

To further evaluate the performance, we also investigated the behaviour of our models in predicting the output for two different combinations of *ratio* and *threshold* included in the dataset representing gentle and heavy compression. The input signals were the same used for the previous evaluation. As expected, these experiments show that it is more challenging to accurately predict output for larger values of the parameters, especially for the *ratio*. These determine abrupt changes in the dynamics and turn into a more challenging scenario for the models. This can also be observed in Tab. 1 and Tab. 2, where for all models the test conditioning losses are generally lower than the test and validation losses. Conditioning losses refer to a single medium compression scenario with settings not included in the training set. This case is easier to predict than heavier compression scenarios, which are included in the other losses.

While similar perceptual performance emerges when listening to the predicted sounds, Tab. 4 shows that LSTM and FF models perform closer to ED models with smaller (i.e. easier) values of the ratio. Instead, with more compression and in turn bigger transients, ED models appear significantly more capacity to make accurate predictions, especially due to their ability to model accurately the attack and release phases.

A comprehensive set of audio examples, additional figures, dataset, and source code are available online.[4]

[4] https://github.com/RiccardoVib/CONDITIONED-MODELING-OF-OPTICAL-COMPRESSOR



(a) (b) (c) (d)

Figure 7: *Spectrograms of frequency sweep input (a), and of output the predictions for the FF (b), LSTM (c), ED (d) models. The Spectrograms refer to* 3.33:1 *of ratio and* −30 *dBU of threshold.*

Table 4: *ESR values for the best models predict the compressor output for four different input sounds: drum kick, hi-hat, guitar, and bass. While conditioning values used in these tests are not new for the models, the samples used here were not part of the training or validation set.*

r = 4.66:1, $t = -10$	Kick	Hi-Hat	Guitar	Bass
FF	0.0093	0.0318	0.0068	0.0156
LSTM	**0.0091**	**0.0287**	0.0058	0.0131
ED	0.0092	1.3408	**0.0054**	**0.0063**
r = 7.33:1, $t = -40$	Kick	Hi-Hat	Guitar	Bass
FF	0.2582	0.1860	0.0765	0.2746
LSTM	0.1624	**0.1643**	0.0941	0.1529
ED	**0.1090**	1.2532	**0.0407**	**0.1255**

Table 5: *Models Set-Ups after hyper-parameter tuning.*

FF	**LSTM**
# Layers: 2	# Layers: 2
# Hidden Units: 32	# Hidden Units: 32
Activation function: 'tanh'	Activation function: 'tanh'
# of Parameters: 1, 217	# of Parameters: 12, 961

ED
Encoder Layers: 1
Decoder Layers: 1
Hidden Units: 64
Activation function: 'sigmoid'
of Parameters: 34, 369

5.1. Aliasing-like effect

To investigate the aliasing-like issue, we analyzed the predicted output for a 16 s input frequency sweep in the range [20, 20000] Hz. Listening to the predicted output sound and inspecting the associated spectrograms it is evident that the FF and LSTM models are more severely affected by this problem, which can be heard starting from 3 kHz. For the other types of input signals included in the dataset, aliasing is not noticeable when listening to the predicted compressor output signals. Encoder-decoder models, on the other hand, appear to be more robust against the aliasing-like issue, and therefore they outperform other architectures also from this perspective. Indeed, aliasing is noticeable only from 5 kHz onward when using the frequency sweep as the input. Aliasing is visible in Fig. 7, which shows the spectrograms for the prediction of the four architectures with the frequency sweep at their input. As expected, we noticed that the aliasing is reduced for models with better ESR. Fig. 7 shows the case of *ratio* and *threshold* set to 3.33:1 and −30 dBU, respectively. The same overall behaviour was observed in the other two cases, gentle and heavy compression. The only difference we noted is the perceptibility of this problem. With gentle compression, since the amplitude of the sweep is greater than in the other two cases, the aliasing-like issue can be heard slightly earlier. The opposite situation happens with heavy compression.

6. CONCLUSION & FUTURE WORK

In this study, we investigated different deep learning architectures for conditioned modeling of an audio analog optical compressor. The models work with no- or small-input buffers in such a way to be suitable for low-latency real-time implementations. In this investigation, two compression parameters, such as the *ratio* and *threshold*, have been included as conditioning values for the networks. The applied architecture was feed-forward and long short-term memory layers, in series or in an encoder-decoder fashion. To evaluate the performance, we used four types of sound signals (kick, hi-hat, guitar, and bass) not included in the training dataset and three different compression settings (heavy, medium, and gentle). The low-frequency signals, such as basses and drum kicks, are the most challenging to predict accurately. Experiments show that feed-forward and LSTM architectures are capable to learn the sustained part of the compression process but show limitations with the transients. The proposed encoder-decoder model, inspired by sequence-to-sequence architectures, appears to outperform feed-forward and LSTM. In particular, it can learn longer temporal dependencies and predict challenging scenarios such as heavy compression with higher accuracy. In addition, the largest encoder-decoder model we used takes approximately 48 hours for the training, on a virtual workstation equipped with 8-Core of an Intel Xeon Gold 5215 CPU @ 2.50 GHz, 32 GB of memory, and a GPU NVIDIA Tesla $V100$ PCIe 16 GB (shared with another instance), and 2.256 seconds for the inference of one second of audio (0.047 ms per sample) on a 2.3 GHz 8-Core Intel Core i9 processor. Encoder-decoder models also appear to suffer less from the aliasing-like effect. All models are capable of interpolating between 'conditioning' settings, and compressing accurately audio signals using parameters not included in the training dataset. In general, all models show more difficulties to predict high frequencies, which results particularly noticeable in the case of the kick and bass.

In future work, we will investigate in more detail the encoder-

decoder approach. Experiments indicated positive correlations between network size and accuracy. Therefore, we cannot exclude performance improvements with larger encoder-decoder models, although the computational burden will further reduce the rates at which experiments can be carried out and the extent to which these models can be used for real-time inference (at least with current computing technologies and architectures). Another aspect to improve is the selection of the loss function because, as previously mentioned, the MSE does not penalise enough wrong predictions for frequencies with low energy, resulting in audible noisy predictions. Moreover, we will extend the conditioning to the full set `CL 1B`, including also attack and release time. This will change the temporal behaviour of the system because the compression would be applied and removed in a variable amount of time. In this way, the model must also learn different temporal dependencies according to the conditioning signals. Finally, we will explore how working with higher sampling rates influences and potentially reduce the aliasing-like issue. Several models in literature have shown that suffer from this aspect but a proper understanding of this phenomenon is still an open challenge.

7. REFERENCES

[1] Jyri Pakarinen, Vesa Välimäki, Federico Fontana, Victor Lazzarini, and Jonathan S Abel, "Recent advances in real-time musical effects, synthesis, and virtual analog models," *EURASIP Journal on Advances in Signal Processing*, vol. 2011, pp. 1–15, 2011.

[2] Maximilian Rest, Julian D Parker, and Kurt James Werner, "Wdf modeling of a korg ms-50 based non-linear diode bridge vcf," in *Proceedings of the International Conference on Digital Audio Effects (DAFx), Edinburgh, UK*, 2017, pp. 5–7.

[3] Jingjie Zhang and Julius O Smith III, "Real-time wave digital simulation of cascaded vacuum tube amplifiers using modified blockwise method," in *Proc. 21th Intl. Conf. Digital Audio Effects (DAFx-18),(Aveiro, Portugal)*, 2018.

[4] Felix Eichas and Udo Zölzer, "Black-box modeling of distortion circuits with block-oriented models," in *Proceedings of the International Conference on Digital Audio Effects (DAFx), Brno, Czech Republic*, 2016, pp. 5–9.

[5] Ben Holmes and Maarten Van Walstijn, "Physical model parameter optimisation for calibrated emulation of the dallas rangemaster treble booster guitar pedal," in *Proc. of the 19th Internation Conference on Digital Audio Effects, Brno, Czech Republic*, 2016, pp. 47–54.

[6] Aäron Van Den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W Senior, and Koray Kavukcuoglu, "Wavenet: A generative model for raw audio," *SSW*, vol. 125, pp. 2, 2016.

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

[8] Marco A Martínez Ramírez and Joshua D Reiss, "End-to-end equalization with convolutional neural networks," in *21st International Conference on Digital Audio Effects (DAFx-18)*, 2018.

[9] Eero-Pekka Damskägg, Lauri Juvela, Etienne Thuillier, and Vesa 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.

[10] Eero-Pekka Damskägg, Lauri Juvela, Vesa Välimäki, et al., "Real-time modeling of audio distortion circuits with deep learning," in *Proc. Int. Sound and Music Computing Conf.(SMC-19), Malaga, Spain*, 2019, pp. 332–339.

[11] Zhichen Zhang, Edward Olbrych, Joseph Bruchalski, Thomas J McCormick, and David L Livingston, "A vacuum-tube guitar amplifier model using long/short-term memory networks," in *SoutheastCon 2018*. IEEE, 2018, pp. 1–5.

[12] Thomas Schmitz and Jean-Jacques Embrechts, "Nonlinear real-time emulation of a tube amplifier with a long short time memory neural-network," in *Audio Engineering Society Convention 144*. Audio Engineering Society, 2018.

[13] Alec Wright, Eero-Pekka Damskägg, Vesa Välimäki, et al., "Real-time black-box modelling with recurrent neural networks," in *22nd international conference on digital audio effects (DAFx-19)*, 2019.

[14] MA Martínez Ramírez, Emmanouil Benetos, and Joshua D Reiss, "Deep learning for black-box modeling of audio effects," *Applied Sciences*, vol. 10, no. 2, pp. 638, 2020.

[15] Marco A Martínez Ramírez, Emmanouil Benetos, and Joshua D Reiss, "Modeling plate and spring reverberation using a dsp-informed deep neural network," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 241–245.

[16] Julian D Parker, Fabián Esqueda, and André Bergner, "Modelling of nonlinear state-space systems using a deep neural network," in *Proceedings of the International Conference on Digital Audio Effects (DAFx), Birmingham, UK*, 2019, pp. 2–6.

[17] Scott H Hawley, Benjamin Colburn, and Stylianos I Mimilakis, "Signaltrain: Profiling audio compressors with deep neural networks," *arXiv preprint arXiv:1905.11928*, 2019.

[18] Christian J Steinmetz and Joshua D Reiss, "Efficient neural networks for real-time modeling of analog dynamic range compression," in *Audio Engineering Society Convention 151*. Audio Engineering Society, 2022.

[19] Colin Lea, Rene Vidal, Austin Reiter, and Gregory D Hager, "Temporal convolutional networks: A unified approach to action segmentation," in *European Conference on Computer Vision*. Springer, 2016, pp. 47–54.

[20] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[21] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep learning*, MIT press, 2016.

[22] Ilya Sutskever, Oriol Vinyals, and Quoc V Le, "Sequence to sequence learning with neural networks," *Advances in neural information processing systems*, vol. 27, 2014.

[23] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 2623–2631.