FULLY CONDITIONED AND LOW-LATENCY BLACK-BOX MODELING OF ANALOG 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

Neural networks have been found suitable for virtual analog modeling applications. Several analog audio effects have been successfully modeled with deep learning techniques, using low-latency and conditioned architectures suitable for real-world applications. Challenges remain with effects presenting more complex responses, such as nonlinear and time-varying input-output relationships. This paper proposes a deep-learning model for the analog compression effect. The architecture we introduce is fully conditioned by the device control parameters and it works on small audio segments, allowing low-latency real-time implementations. The architecture is used to model the CL 1B analog optical compressor, showing an overall high accuracy and ability to capture the different attack and release compression profiles. The proposed architecture' ability to model audio compression behaviors is also verified using datasets from other compressors. Limitations remain with heavy compression scenarios determined by the conditioning parameters.

1. INTRODUCTION

The unique timbre and sound coloring provided by analog circuits and their nonlinearities are still appealing to musicians and sound engineers. The digital emulation of vintage analog musical instruments and audio effects, known as Virtual Analog (VA), has been an active field of research and development for several years [1]. To date, a variety of digital emulations of audio analog devices have been introduced [2]. Recently artificial neural networks have been successfully employed also for VA audio effects [3, 4]. In particular, nonlinear distortion circuits [5, 6, 7] have been accurately modeled using architectures based on Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN). Models suitable for real-world applications (i.e., low-latency and real-time computation) have also been proposed [8, 9, 10]. Challenges remain for nonlinear time-varying audio effects, where the state-of-the-art employs models requiring long audio segments and large networks [11], which are detrimental for latency and real-time computation. This work further investigates time-varying effects, in particular dynamic range compression. Audio compression is a nonlinear effect that reduces the dynamic range of the input signal by a given amount when this exceeds a given threshold [12]. Compression is usually applied to reduce the dynamic range of the signal. The gain reduction is time-varying according to variable attack and release times. In an early attempt to model an analog compressor/leveling amplifier [13], an artificial neural network, based on fully-connected layers, predicts the STFT of the compressed signal. The model, in this case, is large and works with long segments of the input signal. State-of-the-art modeling of audio compression is based on Temporal Convolutional Networks (TCN), which have been adapted to model analog compression [14], allowing real-time computation. However, input-output latency is still high because the model requires a long input segment ($\sim$1.5 seconds) to compute a segment of output accordingly to the receptive field of the network. Specifically, the lower bound latency is equal to the size of the receptive field, which in this case is between 101 and 1008 ms. Both works aimed at modeling the *TELETRONIX LA-2A leveling amplifier* with only two conditioning parameters: a binary switch to choose between the limit and compression modes and the peak reduction. Lastly, a gray-box model for the same device is presented in [15], where RNN and Multi-Layer Perceptron (MLP) networks are used in combination with traditional signal processing techniques. In particular, MLP is used to predict the parameters of the static compression curve based on the conditioning information, while RNN predicts the time parameters for the filters controlling the attack and release times. This model emulates only the compression mode (i.e., the switch parameter is fixed) and includes the peak reduction parameter as conditioning.

In our previous work, we focused on dynamic range compression as well but modeling another analog compressor device [16]. We investigated the use of Encoder-Decoder (ED) Long Short Term Memory (LSTM) based architectures to learn a static temporal profile of the TUBE-TECH CL 1B opto compressor (i.e., fixed attack and release time) conditioned to two control parameters: compression ratio and threshold. The architectures we used are relatively small with respect to network size and length of input segment, allowing low-latency real-time implementations of the model. Here we present improvements to the previously proposed architecture, and we use it for fully-conditioned modeling of the CL 1B, including variable attack and release time. Generally, LSTMs, when taken alone, struggle to learn long temporal dependencies, while CNNs require long receptive fields to model them. We show how combining LSTM and CNN in an ED architecture improves the modeling accuracy, enabling learning various compression amounts and temporal profiles given by different combinations of threshold, ratio, attack, and release values. The ED architecture can provide similar accuracy using networks with a relatively small number of parameters and a small input size compared to TCN-based works. The output gain parameter of CL 1B, and of other compressors as well, is excluded from conditioning because it controls an independent amplification stage applied after the compression, which is not influencing the compression process. In analog compressor devices, the output gain can add characteristic harmonic distortion after the compression stage, but modeling distortion circuits is beyond the scope of this work. Such circuits can be modeled separately from existing techniques [8]. In addition, [13] showed that a single architecture can model compressors with different characteristics; thus we evaluate the proposed ED models with a set of heterogeneous compressors. The rest of the paper is organized as follows. Sec. 2 presents the various compressors used in this study and the associated datasets. Section 3 details the proposed architecture. An overview of the experiments carried out to validate the architecture is in Sec. 4, while the results are presented in Sec. 5. Sec. 6 concludes the paper.

2. DATASETS

In this section, we describe the various compressor datasets we use to train and test our models. Using multiple datasets collected from devices with different characteristics allows for verifying whether the proposed deep learning architecture can model audio compressors in general rather than only a specific unit.

2.1. CL 1B Compressor

The CL 1B[1] is an analog optical compressor manufactured by TUBE-TECH. In optical compression, a lighting-emitting element is fed with the audio signal, and this illuminates a light-sensitive resistor, also known as a photocell. The amplitude of the input signal determines the brightness of the element that, in turn, changes the resistance in the gain attenuation circuit. This device presents five variable control parameters: *ratio*, *threshold*, *attack*, *release times*, and *output gain*. We have built the CL 1B dataset[2] by recording data directly from the device as described in [16]. For this study, we have extended the dataset to include variable ratio, threshold, attack, and release time, but for each combination of the control parameters, we have limited the audio recording to 210 seconds. The output gain is fixed to 0 dB. Input signals include 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), and vocals. The output signal was recorded for 5 different values of each of the four parameters (equally spaced within the selected range), resulting in 625 combinations, which corresponds to a dataset of $\sim$ 36 hours recorded at 48 kHz. Threshold values ranges from 0 to -40 dBu, ratio values from 2:1 to 10:1, attack time from 0.5 to 300 ms, and release time from 0.05 to 10 seconds. The exact values of attack and release time are not marked on the device; therefore, we assume a linear range between the maximum and the minimum and the maximum indicated in the manual. The recordings associated with each combination of the control parameters are split into 21 parts of 10 seconds. From these, 50% of them are randomly removed, ensuring that no parameter combination is either over- or under-represented in the training set (i.e., random selection with uniform distribution against parameter combinations). Therefore, the complete training set includes $\sim$ 18 hours of recording associated with 625 different parameter combinations, out of which 20% is used for validation.

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

[2]https://doi.org/10.5281/zenodo.6497085

2.2. Software Compressors

To further verify the extent to which the proposed architecture models audio compression, we have built three additional datasets from software compressors. We have selected compressors implemented as VST plugins because data collection can be completely automated, adapting an existing tool to work with audio effects [17]. The selected plugins are: the Softube FET Compressor[3], the PSP MicroComp[4], and the u-he Presswerk[5]. The first is a pure digital audio compressor, while the other two are VA devices. The input signal is the same described in Sec.2.1. For each software compressor, we have selected 4 variable parameters, summarized in Tab. 1, with functionality and range close to those selected for the CL 1B dataset. Also, in these cases, the 210 seconds output signals were recorded at 48 kHz for 625 different combinations of the control parameters (i.e., 5 equally spaced values within the selected ranges). The training, validation, and test set are built identically to that of CL 1B. Since FET Compressor has a fixed threshold, we have included in the set of variable parameters its input gain, which allows us to vary the relative attack/release point with respect to the input signal.

Table 1: *Selected variable parameters and respective ranges for the software compressors.*

VST	Parameter	Values
MicroComp	ratio	[2:1, 12:1]
	threshold	[0, -30] dB
	attack time	[0.5, 300] ms
	release time	[0.05, 3.30] s
Fet Compressor	ratio	[2:1, 10:1]
	input	[-6, +6] dB
	attack time	[20, 800] μs
	release time	[0.05, 1.1] s
Presswerk	ratio	[2:1, 10:1]
	threshold	[-10, -40] dB
	attack time	[0.1, 150] ms
	release time	[0.015, 2.5] s

2.3. LA-2A Leveling Amplifier

Teletronix LA-2A Leveling Amplifier[6] is an analog limiter/compressor, whose optical gain reduction works similarly to the CL 1B. The LA-2A is used in all previous works on black box modeling of analog compression [14, 15]. We consider this device in our experiments to allow a performance comparison with the state-of-the-art, although the LA-2A has a fundamental difference from other compressors used in our study. The LA-2A does not present variable attack and release time that users can fix using dials. Instead, the LA-2A presents an average attack time of 10 ms and a multi-stage release. The duration of the first stage is 0.06 seconds, while the second stage of release is controlled by the photocell's memory, which depends on the brightness and time the light-emitting has been on. The duration of the second stage ranges from 0.5 to 5 seconds for the complete release. This implies

[3]https://www.softube.com/fet-compressor-mk-ii

[4]https://www.pspaudioware.com/products/psp-mastercomp#psp-microcomp

[5]https://u-he.com/products/presswerk/

[6]https://www.uaudio.com/hardware/la-2a.html

that if the compression is heavy and/or the signal has been above the threshold for a long time, the LA-2A's release will be slower. Therefore the attack and release times of the LA-2A are unknown a priori but depend on the past input signal. The device presents an *output gain*, a switch controlling if the device is operating in a *limit* or *compression* mode, and a *peak reduction* that controls the amount of compression to apply. The LA-2A dataset[7] [13] contains approximately 20 hours of recordings at 44.1 kHz of the device fed with various acoustic and synthetic instruments, percussive clips, excerpts of musical pieces, tones, and noise bursts. The dataset includes the variations of two control parameters (in total 20 combinations): the binary switch that sets the device in either compress or limit mode and the peak reduction parameter that controls the amount of compression as a function of the input level. We use the same training, validation, and test split included in the dataset, representing 80%, 15%, and 5%, respectively. Recordings taken for each combination of the control parameters are not identical in both duration and contents.

3. PROPOSED ARCHITECTURE

The architecture we propose was developed by experimenting primarily with the CL 1B dataset, starting from the ED model we investigated in our previous work [16]. The encoder processes n past sample of the input sequence together with the additional conditioning parameters and returns its final internal states and output. The output is discarded while the internal state is passed to the decoder as its initial internal state. The decoder learns to predict the target sample at each time step, given the input sample at the current time step and a number n of past input samples. The improved architecture we propose in this paper is illustrated in Fig. 1. The input signal and conditioning parameters are fed to the network separately. In order to reduce the model complexity of the encoder, we replaced the LSTM layer with a 1D convolutional layer, which provides a representation of the signal's past information using fewer trainable parameters. Control parameters are fed to the encoder through a simpler fully-connected layer. The decoder still includes an LSTM layer, but it works on a larger input-output audio segment rather than on single samples. The architecture uses as input a segment of $2w$ past sample of the input signal x, which we split into two halves: one labeled as "far" past ($[x_{-2w}, ..., x_{-w}]$) and one labeled "recent" past ($[x_{-w}, ..., x_0]$). The "far" past represents the input for the encoder, and the "recent" past is the input for the decoder, both of size w. The encoder output, representing the internal states of the LSTM layer in the decoder, must match the LSTM number of neurons u. The LSTM layer is followed by a fully-connected layer with *sigmoid* activation function. Finally, at the output, we have another fully-connected layer with, in our case, a number of neurons of the same decoder input segment, where each neuron predicts an audio sample. With this architecture, the size of the output layer, o, can be reduced to predict as little as one sample at a time, allowing a fine-grain accuracy at the expense of the computational cost. In our experiments, we used input and output signal segments of identical size to minimize the computational complexity of the model and to work similarly to most real-time audio stream processing applications, in which input and output buffer sizes are identical. Lastly, preliminary experiments showed that kernels of the convolutional layer with identical size to the encoder input size led to better accuracy. In addition, this choice simplifies the architecture as no transformation of the convolutional layer's output is needed to match the dimensionality of the LSTM internal state. Conditioning parameters compose the input vector for the fully-connected layer in the encoder, as illustrated in Fig. 1. Before feeding the networks, the conditioning values are normalized between $[0, 1]$. The output of the fully-connected layer is added to the output of the convolutional one in order to compute the states to give to the decoder. The internal state size has to match the number of units of the LSTM layer; for this reason, the number of units is the same for all the layers.

[7]`https://zenodo.org/record/3824876`

4. EXPERIMENTAL DESIGN

4.1. Models

As stated in previous sections, our objective is to achieve an accurate black-box modeling of audio compression using deep-learning models with low latency and low computational complexity. We limit our investigation to ED models with w set to $(16, 32, 64)$, representing the number of input samples for both the encoder and decoder. The size w is also the intrinsic audio latency of our architecture. Our models work with an overall input segment size of $2w$ samples. Therefore, in a hypothetical real-time audio stream processing application, the overall latency of the system is $5w$ samples ($4w$ of latency for the double input-output buffering). To contain the computational cost of the models' inference, we take two approaches. First, we limit the number of trainable parameters to $\sim$ 100k at most, and therefore we investigate only models with $(32, 64, 128)$ numbers of units u. Second, we minimize the frequency at which the inference has to be executed to predict the stream of output samples. In particular, we consider only output sizes o equal to w, which requires only two predictions to fill the output audio block with size $2w$. Execution time and memory requirements can be further reduced using existing techniques such as pruning [18], quantization [19], tensor decomposition [20], knowledge distillation [21], and skip-RNN [22]. Low-latency for digital audio effects is essential for live-audio applications, while in production settings, modern digital audio workstations can automatically compensate for the plugin's latency to avoid temporal misalignment in multi-track mixing. However, automatic compensation works only within a limited range. For example, AVID Pro Tools can automatically manage up to 4096 samples of latency.

In order to compare our architecture with the state-of-the-art, we implemented a TCN network that can be trained with our CL 1B dataset. We adapted the best architecture from [14] to work at 48 kHz, taking as input $72,000$ samples (1.5 s) of the audio signal and predicting an output segment of $58,668$ samples ($\sim$ 1.2 s) with a receptive field of $13,332$ samples ($\sim$ 278 ms). This, together with the use of the LA-2A dataset in our experiments, allows a comprehensive cross-comparison of our architecture against the state-of-the-art. Comparisons with baseline architectures, such as simple LSTM or dense layers, are detailed in our previous work [16].

4.2. Training & Testing

All models are trained with a batch size of 128 and employing Adam [23] optimizer with gradient norm scaling of 1 [24]. ED models use an initial learning rate of 10^{-4}, while TCN model uses $3\ 10^{-4}$, as reported in [14]. Models are trained for 50 epochs except for the LA-2A case, in which we train models for 60 epochs

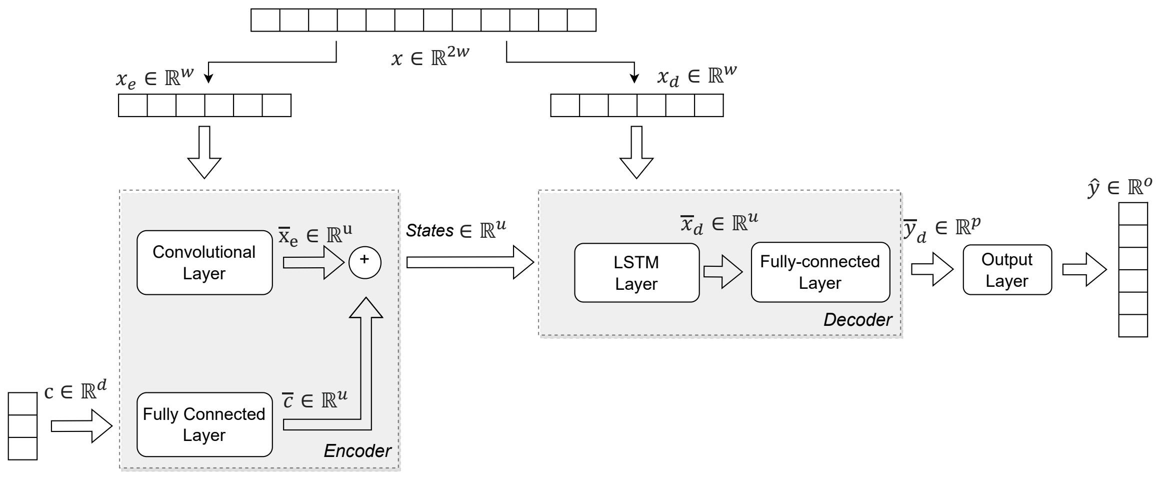

Figure 1: *Proposed architecture. Encoder consists of two layers: one taking the conditioning control parameters and the other one taking the past of the input signal. Control parameters are processed by a fully-connected layer, while the input is processed by the convolutional layer. The outputs of both layers are added to each other to compute the internal states that act as conditioning for the LSTM layer composing the decoder. The decoder takes the "current" input signal. w is the size in samples of the encoder and decoder input signals, and u is the number of units, representing as well dimension of the layer's outputs. The internal states dimension must match the number of units of the LSTM layer; for this reason, we selected the same number of units for all the layers. A fully-connected layer with p units and sigmoid activation function is placed after the LSTM layer, producing the decoder's output. This is fed to another fully-connected layer, including o neurons with linear activation function generating the o output samples of the model.*

to allow a comparison with [14]. Preliminary experiments varying input size $2w$ and the number of units u of the ED models use only half of the CL 1B dataset, still representing 625 combinations of conditioning parameters. The best-performing model, detailed in Sec. 4, is trained using the full dataset for 200 epochs in total. Results for the test loss are computed with the model's weights that minimize the validation loss throughout the training epochs. We have asserted the ability of the ED architecture to predict conditioning values never seen during the training in our previous investigation [16]. On the other hand, predicting abrupt and quick changes in the dynamic was challenging for the network. Similar behavior is also present in [15], where the prediction error increases with the increasing of the conditioning peak reduction value (i.e., in scenarios with larger dynamics change between input and output). Instead, conditioning values unseen during training do not determine a significant increase in the prediction error. For this reason, in this study, we test the generalization capability of the network using pairs of input signals and conditioning values unseen during the training phase, as detailed in the CL 1B dataset description.

4.3. Loss Function

As emerged in our previous work on CL 1B [16], error hikes during the attack phase of the compressor, triggered by fast changes in the input signal amplitude. To address this limitation, we investigate the use of different loss functions: we evaluate the Mean Absolute error (MAE), the Error-to-Signal Ratio (ESR), and a Short-Time Fourier Transform-based (STFT) loss function with different resolutions (window size of [8, 16, 32] and hop size of [2, 4, 8], respectively). As expressed in Eq. 1, the STFT-based loss function minimizes the spectral difference between the target and the predictions with a multi-resolution spectral loss. The component of the spectral loss with resolution m compares the two audio signals by summing the $L1$ differences between both their linear- and log-spectrograms. The models are trained for 50 epochs using different loss functions. Since absolute values returned by different loss functions are not comparable, results are qualitatively assessed by inspecting the waveforms of the predictions against the true outputs, focusing on the accuracy of the attack phase of the compression. Subsequently, we further explore combinations of the mentioned loss functions for training the model.

$$\begin{aligned}L_{STFT}(y, \hat{y}) = &|||STFT_m(y)| - |STFT_m(\hat{y})|||_1 \\ &+ || \log(|STFT_m(y)|) - \log(|STFT_m(\hat{y})|)||_1\end{aligned} \quad \text{(Eq. 1)}$$

5. RESULTS & COMPARISONS

This section details the performance of the proposed architecture with the datasets detailed in Sec. 4. Comparisons with the state-of-the-art are included in Sec. 5.1, where the best TCN architecture from [14] is used to model the CL 1B; in Sec. 5.4 where our best ED architecture is used to model the LA2A used in all previous works on deep-learning modeling of dynamic range compression; and in Sec. 5.5, where latency and computational cost of our best ED model are compared with the best TCN architecture from [14]. Datasets, source code, trained models, audio examples, and additional figures available online[8].

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

5.1. CL 1B

Tab. 2 details validation and test loss for models with different input segment size $2w$ and different numbers of units u. Accuracy is inversely proportional to the input size. Smaller sizes allow predicting output samples with a finer grain, resulting in smaller errors, although the network is fed with a smaller segment of audio samples. The accuracy is proportional to the number of units u, as larger networks appear to be beneficial to the model's accuracy. The most accurate model presents input segments $2w$ of 32 samples and 128 internal units u. Even if the best configurations resulted in 128 internal units, we used 64 hidden units for the final model. Therefore, we trained the model for 200 epochs, and we will refer to it for the figures and results detailed in the rest of this section. Associated validation and test loss are shown at the bottom of Tab 2. The model continues to learn at the increase of epochs, and conceivably losses could further decrease if training continues. The choice of 64 units is determined by our goal of minimizing computational complexity. A model with 128 units has almost four times the number of trainable parameters and brings improvements in the losses that do not justify the choice of having a bigger computational complexity. On the other hand, a bigger number of hidden units could lead to more remarkable improvements when increasing the number of training epochs. We noticed that the predicted signals include spurious tones generated by errors at the boundary of the output segments. We found that the frequency of such tones is equal to nF_r/o, where F_r is the sampling rate, o is the output size, and n is an integer representing the number of overtones, ranging from 1 to 6. The amplitude of the spurious tones does not exceed -70 dB; hence these are often masked by the compressed output audio signals within high-frequency contents. Cumulative figures on prediction error, such

Table 2: *Validation and test loss (MSE) for ED model against different input segment sizes $2w$ and the number of units u. The test loss refers to unseen audio samples during training but seen conditioning values. The number of trainable parameters and epochs is also detailed. The models are trained for 50 epochs, using half dataset and MSE as the loss function. The bottom section of the table refers to the selected best model, which has been trained using the full dataset for up to 200 epochs.*

$2w$	u	Val Loss	Test Loss	Params	Ep.	Data
32	32	$1.45e^{-5}$	$1.12e^{-5}$	8,912	50	50%
-	64	$1.09e^{-5}$	$8.75e^{-6}$	24,912	-	-
-	128	$1.00e^{-5}$	$8.08e^{-6}$	81,488	-	-
64	32	$6.01e^{-5}$	$5.39e^{-5}$	10,976	-	-
-	64	$1.60e^{-5}$	$1.29e^{-5}$	28,000	-	-
-	128	$1.03e^{-5}$	$8.76e^{-6}$	86,624	-	-
128	32	$3.28e^{-4}$	$3.47e^{-4}$	15,104	-	-
-	64	$3.60e^{-4}$	$3.79e^{-4}$	34,176	-	-
-	128	$3.13e^{-4}$	$3.20e^{-4}$	96,896	-	-
32	64	$1.42e^{-5}$	$1.74e^{-5}$	24,912	50	100%
-	**-**	$\mathbf{6.92e^{-6}}$	$\mathbf{8.21e^{-6}}$	**-**	**200**	**-**

as averages across the entire dataset, are poorly informative with respect to the model's conditioned behavior. For this reason, we analyze the trend of prediction accuracy against the four conditioning parameters. The two colormaps in Fig. 2 display the errors for all combinations of attack and release time with fixed ratio and threshold, as well as for all combinations of ratio and threshold with fixed attack and release time. The fixed parameters were set at the middle of their range. To provide a fair and informative representation of the model's conditioned behavior, the MSEs are computed using the same audio signal for all parameter combinations, which includes 10 second of percussive and bass sounds taken from the test set. The various conditioning scenarios determine major changes in the dynamic range of the output signal, which should be taken into account when comparing the MSEs. To overcome this challenge and allow direct comparison of the errors, we compute the MSEs represented in Fig. 2 after normalizing target outputs to $[-1, +1]$ and applying the same normalization factor to the model's predictions. From the left image, it is evident that the prediction accuracy drops with the growth of attack and, in particular, release time. This reflects that with longer temporal dependencies, accurate prediction is more challenging. The right image shows that heavy compression scenarios (i.e., higher ratio and lower threshold) are also more challenging to be predicted accurately. Since the dataset was randomly split, for some parameter combinations, the overall percentage of signal above the threshold, which triggers the compressor, may not be identical between the training and test set. This, in turn, could be the reason why the error variations are not monotonic, in particular for the left colormap. However, we should also consider that the MSE ranges illustrated in Fig. 2 are extremely small, as visible from the associated color-bar values. Fig. 3 shows an example with three different

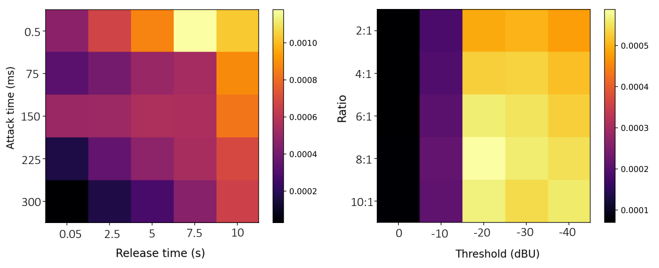

Figure 2: *ColorMaps of the test loss (MSE) for different combinations of conditioning parameters. Test errors for different values of attack and release times, with ratio and threshold fixed to* -20 dBu *and* 6:1 *(Left). Test errors for different values of ratio and threshold, with attack and release time fixed to the middle of their range (Right). Errors are computed after normalizing targets to* $[-1, +1]$ *and applying the same normalization factor to the predictions.*

settings of attack time, specifically for 0.5, 150, and 225 ms. Other parameters are fixed to ratio 6:1, threshold -30 dBu, and release time 0.05 s. It is visible how the model has learned different compression temporal profiles. The model handles the RMS envelope quite accurately, applying the gain reduction accordingly to the different attack time values. Lastly, Tab.3 compares the test losses (MSE and MAE) of the best TCN model from [14] and of our best ED model when trained with the CL 1B dataset. The losses after 50 epochs are reported, and the ED model shows better performance. In addition, from informal listening, it is evident that the TCN model introduces more audible artifacts than the ED model.

5.2. Loss Functions

Fig. 4 shows prediction versus target output for a plucked bass example using models trained with four different loss functions. We

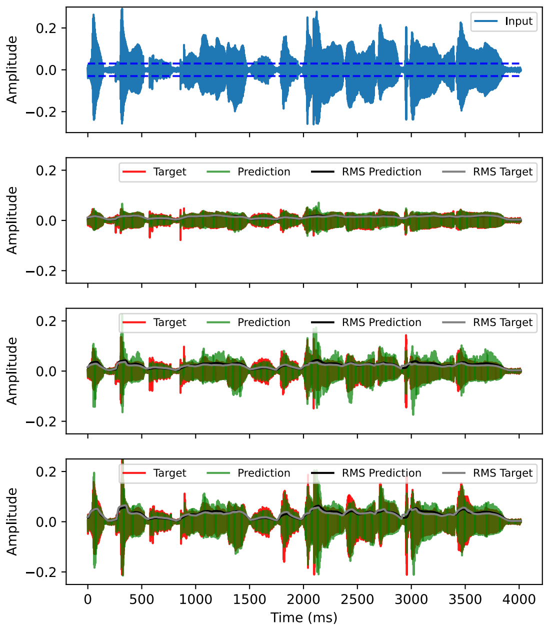

Figure 3: *Example of input waveform (first row) and associate output predicted versus target waveforms (second to fourth rows) for increasing values of attack time:* 0.5, 150, *and* 225 ms. *The ratio, threshold, and release time are set to* 6:1, −30 dBu, *and* 0.05 s, *respectively. The horizontal lines on the input waveform represent the threshold level.*

Table 3: *Test losses (MSE and MAE) for ED and TCN model. Losses refer to unseen audio samples during training but seen conditioning values. The number of trainable parameters and units is also detailed. The kernel size k, dilation factor d, and number of TCN blocks are reported for the case of TCN networks. The models are trained for* 50 *epochs with MSE as the loss function.*

Models	k	u	d	n	MSE	MAE	Params
TCN	13	32	10	4	$9.54e^{-4}$	$1.33e^{-2}$	51,464
ED-32	-	64	-	-	$1.74e^{-5}$	$1.93e^{-3}$	24,912

select a plucked bass sound because its amplitude envelope (sharp attack and decaying amplitude) is representative of scenarios that the model struggles to cope with, in particular during the initial phase of the compression. The plots refer to heavy-compressed scenarios (−40 dBu as threshold and 10:1 as ratio). In general, we found that models introduce more audible artifacts when these are accurate with onsets crossing the threshold and vice versa. When using MAE as the loss function, the models generate fewer artifacts but fail to learn the compression attack phase, applying the gain reduction instantaneously. On the other hand, when using MSE, the models learn the compressor attack and release phases more accurately but produce more audible artifacts. Using ESR or STFT-based loss functions provided poor performances, in particular, the latter one. The STFT-based's poor performance could be affected by the small output segment sizes that do not allow adequate frequency resolution. No combinations of loss functions led to advantages with respect to the attack phase. For this reason, we use the MSE only as the loss function for the training of our models, whose performance is detailed in previous sections.

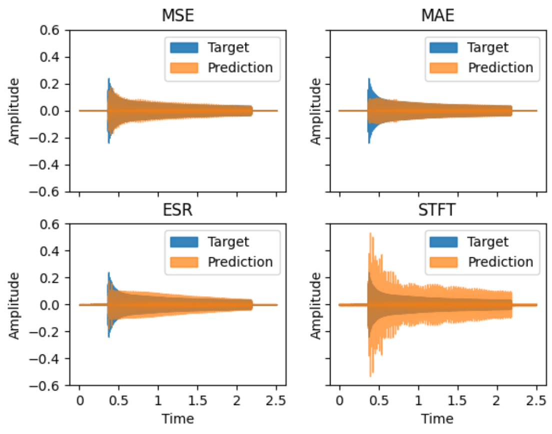

Figure 4: *Predicted waveform against the target for models trained using different loss functions: MSE, MAE, ESR, and STFT-based. The target example refers to a heavy-compressed plucked bass scenario (*−40 dBu *as threshold and 10:1 as ratio). These results are used to determine the influence of the loss functions on the predictions and are not representative of the accuracy of the final model.*

5.3. Software Compressors

Tab. 4 reports the validation and test loss associated with the software compressor datasets. These are obtained training for 50 epochs, the best model configuration derived from the CL 1B experiments, which uses 32 samples as the input segment $2w$, 64 hidden units u, and MSE as the loss function. The order of magnitude of the losses for all software compressors is similar to the CL 1B case, proving that the ED model is able to learn different compression profiles.

Table 4: *Validation and test losses (MSE) for ED model against software compressors. The models have* 64 *as the number of units* u*, and the input segment size* $2w$ *is equal to* 32 *samples. The models are trained for* 50 *epochs and use MSE as the loss function.*

Dataset	Val Loss	Test Loss
MicroComp	5.85^{-5}	$8.35e^{-6}$
Fet Compressor	$1.25e^{-4}$	$6.68e^{-5}$
Presswerk	$4.42e^{-5}$	$5.29e^{-5}$

5.4. LA-2A

Tab. 5 shows different test losses of our model trained on the LA-2A dataset for 60 epochs using different input segment sizes. The trend is similar to CL 1B, smaller input/output size turns in more accurate results. The MAE loss is also reported in the table in order to have a direct comparison with the TCN models in [14]. The number of hidden units is set to 64 to limit the number of trainable parameters, which are detailed in the table as well. Our proposed architecture trained on the LA-2A dataset presents an MSE two orders of magnitude greater than the one obtained with the CL 1B dataset. Although the ED model presents a higher loss, it is still competitive with the TCN-based stat-or-the-art black-box model of the LA-2A, which presents a lower MAE equal to $7.66e^{-3}$ but uses almost twice as many trainable parameters and has a latency of 302 ms. Convolutional models generally converge quicker than LSTMs, which need a considerably larger number of training epochs to achieve similar losses. Hence this comparison based on an equal number of training epochs may be unfair to our ED model, which features an LSTM layer in the decoder. In addition, the LA-2A device presents faster attack and release time and fewer conditioning parameters than CL 1B. As stated before, the LA-2A has no tunable attack and release times, but these depend on the past state of the luminescent element. This can motivate the slight drop in accuracy of our architecture, which may not be able to infer the right compression temporal profile from the audio signal without explicit input data on attack and release time. The difference in the MSE can also be determined by differences between the datasets, such as type and levels of input signals, which have an impact on the overall amount and distribution of the audio compression.

Table 5: *Test loss (MAE, MSE, and ESR) for the ED model against different window input sizes. Both the encoder and decoder have* 64 *as the number of units. Input size refers to the number of total input samples used to compute the outputs; the encoder and decoder input sizes has to be considered half of this value. The number of trainable parameters is also reported. The models are trained for* 50 *epochs and use MSE as the loss function.*

$2w$	MAE	MSE	ESR	Params
32	2.48e^{-2}	1.59e^{-3}	2.43e^{-1}	24,656
64	2.54e^{-2}	1.63e^{-3}	2.49e^{-1}	27,744
128	2.65e^{-2}	1.64e^{-3}	2.51e^{-1}	33,920

5.5. Efficiency

The architecture we propose is designed to minimize latency and computational complexity, aiming at live audio applications. Considering the best model with 32 samples as input segment $2w$, the ED model has a latency of 16 samples, equivalent to 0.33 ms samples at 48 kHz sampling rate. The total latency in real-time audio stream processing application, including the double input-output buffering, is equal to 80 samples, equivalent to 1.66 ms. Tab. 7 reports the Floating Point Operations (FLOPs) required for each layer of the ED model. Breaking down the network, we have three Fully Connected (FC), one convolutional, and one LSTM layer. Each layer requires a different number of FLOPs, depending on the encoder and decoder input sizes w, number of conditioning parameters d, and output size o. The convolutional layer's FLOPs are influenced by the length of the kernel k and the number of filters f, which in our architecture are equal to w and u, respectively. In Tab. 7, we detail FLOPs for ED models with different input-output sizes and the respective intrinsic latencies, which are a key difference from the state-of-the-art TCN models. In [14], the most accurate TCN model has a latency of 302 ms, whereas experiments are detailed with other TCN models with latency as small as 101 ms. In our case, the best ED model presents a latency of only 0.33 ms. On the other hand, the most accurate TCN model is less computationally demanding since the inference predicts significantly longer segments of audio than our ED model. In particular, the inference requires $215,664$ FLOPs, equivalent to just 4 FLOPs per sample. Finally, we have also experimented using a TCN architecture that can deliver latency as low as our best ED model, using input and output segments of 32 and 16 samples, respectively. Results show that the TCN model trained on the CL 1B dataset is less accurate than the equivalent ED model. In particular, the MSE after 50 epochs is $4.20e^{-5}$ (TCN) versus $1.74e^{-5}$ (ED) as reported in Tab.2.

Table 6: *Number of Floating Point Operations (FLOPs) for the different layers of the ED model. d is the number of conditioning parameters, w is the input size, o is the output size, u is the number of hidden units, k is the kernel shape, and f is the number of filters.*

Layer	FLOPs
FC (conditioning)	2 x d x u
FC (decoder)	2 x u x u
FC (output)	2 x u x o
Convolutional	2 x f x k
LSTM	8 x (w + u) × u
Sigmoid function	4 x u

Table 7: *FLOPs for ED models with different input-output sizes (total FLOPs for inference, FLOPs per audio samples, and required GFLOPS for real-time implementation) and associated intrinsic latency (without accounting for the audio input-output buffering), considering* 48 kHz *sampling rate.*

$2w$	u	FLOPs	FLOPs/smp	GFLOPS	Latency
32	32	16,256	1,016	0,49	0.33 ms
32	64	52,480	3,280	1.57	-
32	128	186,368	11,648	5.59	-
64	32	22,912	716	0.34	0.66 ms
64	64	64,256	2,008	0.96	-
64	128	208,384	6,512	3.13	-
128	32	39,296	614	0.29	1.33 ms
128	64	90,880	1,420	0.68	-
128	128	255,488	3,992	1.92	-

6. CONCLUSION

We proposed a deep-learning architecture for black-box modeling of audio dynamic range compression. The model is in an Encoder-Decoder based on LSTM and convolutional layers. The encoder processes the near-past audio samples together with the values of the control parameters, which are encoded in a state helping the decoder to infer the output given the associated input samples. We demonstrated that the proposed architecture is able to model different types of compressors, three software, and two analog ones, with different compression characteristics. The model is designed to minimize latency and computational complexity and ease implementation in real-time audio stream processing applications. We conditioned the model against variations of the compressor's control parameters, including compression ratio, threshold, attack, and release time. The latter two change the system's temporal characteristics because the gain reduction is applied and removed in a different amount of time. Therefore, the architecture is able to learn nonlinear time-varying characteristics of the dynamic range compression, subject to attack and release time values, passed as conditioning parameters to the network. The model is competitive with state-of-the-art works presenting similar accuracy while using shorter input signal segments. Prediction accuracy drops slightly with heavy compression settings. In particular drastic dynamic changes during the attack phase of the compression are the most challenging to predict and produce audible artifacts. We have also investigated the influence of different training loss functions, showing how the MSE is the most suitable to model the initial phase of the compression. Finally, we detailed the latency and computational complexity of the proposed architecture. Future work includes investigations to reduce the presence of audible artifacts, which are also clearly visible from spectrograms of the predicted output. Preliminary experiments suggest that ED models with smaller output sizes may significantly contribute to minimizing the presence of audible artifacts while slightly penalizing efficiency in terms of computational cost and intrinsic latency.

7. REFERENCES

[1] Jussi Pekonen and Vesa Välimäki, "The brief history of virtual analog synthesis," in *Proc. 6th Forum Acusticum. Aalborg, Denmark: European Acoustics Assoc.*, 2011, pp. 461–466.

[2] 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.

[3] 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 Int. Conf. on Acoustics, Speech and Signal Processing*. IEEE, 2020, pp. 241–245.

[4] Marco A 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.

[5] 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.

[6] Eero-Pekka Damskägg, Lauri Juvela, Etienne Thuillier, and Vesa Välimäki, "Deep learning for tube amplifier emulation," in *ICASSP 2019-2019 IEEE Int. Conf. on Acoustics, Speech and Signal Processing*. IEEE, 2019, pp. 471–475.

[7] 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.

[8] 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)*, 2019, pp. 332–339.

[9] Alec Wright, Eero-Pekka Damskägg, Vesa Välimäki, et al., "Real-time black-box modelling with recurrent neural networks," in *Proc. Int. Conf. on Digital Audio Effects (DAFx-19)*, 2019.

[10] Julian D. Parker, Fabián Esqueda, and André Bergner, "Modelling of nonlinear state-space systems using a deep neural network," in *Proc. Int. Conf. on Digital Audio Effects (DAFx-19)*, 2019, pp. 2–6.

[11] Marco A Martínez Ramírez, Emmanouil Benetos, and Joshua D. Reiss, "A general-purpose deep learning approach to model time-varying audio effects," *Proc. Int. Conf. on Digital Audio Effects (DAFX-19)*, 2019.

[12] Udo Zölzer, Xavier Amatriain, Daniel Arfib, Jordi Bonada, Giovanni De Poli, Pierre Dutilleux, Gianpaolo Evangelista, Florian Keiler, Alex Loscos, Davide Rocchesso, et al., *DAFX-Digital audio effects, Second Edition*, chapter 5, Wiley & Sons, 2011.

[13] Scott Hawley, Nenjamin Colburn, and Stylianos Ioannis Mimilakis, "Profiling audio compressors with deep neural networks," *Journal of the Audio Engineering Society*, 2019.

[14] 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.

[15] Alec Wright, Vesa Välimäki, et al., "Grey-box modelling of dynamic range compression," in *Proc. Int. Conf. on Digital Audio Effects (DAFX-22)*, 2022, pp. 304–311.

[16] Riccardo Simionato and Stefano Fasciani, "Deep learning conditioned modeling of optical compression," in *Proc. Int. Conf. on Digital Audio Effects (DAFX-22)*, 2022, pp. 288–295.

[17] Stefano Fasciani, "Tsam: a tool for analyzing, modeling, and mapping the timbre of sound synthesizers," pp. 129–136, 2016.

[18] Russell Reed, "Pruning algorithms-a survey," *IEEE transactions on Neural Networks*, vol. 4, no. 5, pp. 740–747, 1993.

[19] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. of the IEEE Conf. on computer vision and pattern recognition*, 2018, pp. 2704–2713.

[20] Andros Tjandra, Sakriani Sakti, and Satoshi Nakamura, "Tensor decomposition for compressing recurrent neural network," in *2018 Int. Joint Conf. on Neural Networks (IJCNN)*. IEEE, 2018, pp. 1–8.

[21] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean, "Distilling the knowledge in a neural network," *Conf. on Neural Information Processing Systems*, 2015.

[22] Víctor Campos, Brendan Jou, Xavier Giró-i Nieto, Jordi Torres, and Shih-Fu Chang, "Skip RNN: Learning to skip state updates in recurrent neural networks," *Int. Conf. on Learning Representations*, 2017.

[23] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *Int. Conf. on Learning Representations*, 2014.

[24] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio, "On the difficulty of training recurrent neural networks," in *Int. Conf. on machine learning*. Pmlr, 2013, pp. 1310–1318.