CONDITIONING METHODS FOR NEURAL AUDIO EFFECTS

Riccardo SIMIONATO (riccardo.simionato@imv.uio.no)[1] and
Stefano FASCIANI (stefano.fasciani@imv.uio.no)[1]

[1]*Department of Musicology*, **University of Oslo**, Norway

ABSTRACT

Machine learning techniques have become a common approach for modeling analog audio effects. Black-box and hybrid solutions have been applied to a large variety of audio effects. Audio effects usually incorporate user-controllable parameters, and how to infuse this information into the networks is still a challenge. Feature-wise Linear Modulation is a popular conditioning method, but its use in audio effect modeling is still limited. This technique involves an affine transformation via learnable coefficients based on conditioning information. This study compares this approach with other proposals used in this field, such as gated activation. In addition, the control parameters may present a nonlinear relationship with the effect's associated sonic response. Therefore, the investigation also considers nonlinear mapping. This case study investigates two types of analog audio effects: distortion and dynamic range compression. Results indicate the conditioning layer leads to better performance if placed at the end of the architecture, and the Feature-wise Linear Modulation method outperforms other approaches. In addition, nonlinear mapping can be beneficial for cases with strong nonlinear relationships between parameters, such as the overdrive effect.

1. INTRODUCTION

Machine learning has become widely used in audio modeling effects. In an early attempt, a multilayer feedforward network was utilized to learn a high-pass filter and the Ibanez Tube-Screamer distortion effect [1]. Subsequently, convolutional-based networks have been applied for a vacuum-tube amplifier [2] and distortion pedals [3]. Here, conditioning information was infused using a gated activation (GA) as used in the original Wavenet network [4]. Recurrent networks have also been applied for modeling distortion pedals and amplifiers [5–7]. In the case of [6, 7], the conditioning information was incorporated as an extra input to the network. The mentioned works considered one parameter control. A larger number of parameters are considered in [8, 9], where a sequence-to-sequence-based model, based on recurrent networks, exploited the sharing of internal states for the case of an optical compressor. One network encodes the conditioning information, and in this case, some past samples of the signal in its internal states are also shared with the decoder when inferring the output. Two parameters were initially used [8] and later extended to four [9], particularly including parameters determining a time-variant response, such as attack and release time. Still, in the case of another optical compressor, a Temporal Convolution network (TCN) has been used [10] together with the Feature-wise Linear Modulation (FiLM) method [11]. In this case, two parameters were included, but they did not have time characteristics. The FiLM method computed two vectors from the conditioning information to perform linear mapping of the input or some layer's output. FiLM layers learn how to manipulate the control parameters to create a map and decide how this information should influence the network. An activation function such as ReLU or sigmoid can be placed at the end to decide how much information should pass to the inferring network.

Machine learning modeling of nearly all types of analog effects has been investigated. However, not all the proposed models integrate the variable control parameters typically found on audio effect units. A comprehensive model of an audio effect should include user-controllable parameters that determine variations in the sound alteration process. These parameters should allow for continuous changes within their range, determining an immediate change in the model's response. To address this gap in machine learning modeling of audio effects, we further investigate approaches for conditioning with control parameters artificial neural network models of different effects. We compare three approaches used in the field, such as the FiLM and GA layers, against concatenating the conditioning information with the input samples representing the baseline. FiLM and GA are placed at different stages in the architecture to compare when the conditioning information is more beneficial for prediction accuracy. Finally, control parameters may present nonlinear relationships, resulting in a variation of the sound alteration process of the audio effect. Therefore, including nonlinearities in the conditioning block can be beneficial. We use two audio effects to evaluate the various conditioning approaches: overdrive and compressor. For each effect, we select only two variable parameters to condition the networks: the level and tone for the overdrive and the ratio and threshold for the compressor.

Section 2 presents our experiments' methodology, architectures, and datasets. Section 3 detailed the obtained results and the consequent discussion, while Section 4 concludes the paper with a summary of the findings.

2. METHODS

The experiments reported in this paper aim to explore different conditioning methods for artificial neural network modeling of audio effects. The architecture is based on state-space models (SSMs) [12]. In particular, we employ the diagonal state space model variant, the S4D layer [13].

We explore the placement of conditioning layers at different stages of the neural network. Specifically, we consider three configurations labeled *pre*, *post*, and *pre-post*. The first configuration presents the conditioning layer before the S4D layer, the second after the S4D layer, and the last one includes a conditioning layer before and after the S4D layer. At the same time, we evaluate different network architectures for the conditioning layer.

In particular, we use the FiLM and the GA layers. The first consists of computing two vectors from the conditioning information using a linear function to perform a linear mapping of input or some layer's output. The second involves learnable convolution weights for the conditioning vector and the latent space computed from the input vector. The sigmoid is applied to the outputs of the first convolution, while the hyperbolic tangent is applied to the second one. Finally, the two results are element-wise multiplied together. The FiLM layer originally used FC layers to predict the coefficient for the transformation. We expand the methods using convolution as in the GA. These approaches are compared with a baseline we obtain by training similar S4D-based architecture without conditioning layers and with conditioning values provided as input and the array of audio samples. Furthermore, we carry out experiments exploring different transformations inside the FiLM layer. Linear mapping preserves the dimension of any affine subspaces. Still, in the context of audio effect modeling, control parameters may present nonlinear relationships with the sound-altering process of the audio effect. For this reason, we explore nonlinear transformations to assert if increasing the transformation complexity benefits the conditioning process. In particular, we selected transformations of odd orders due to their anti-symmetric characteristic.

As mentioned, the architecture selected for our experiments is based on SSMs, particularly the S4D model. These operate similarly to recurrent neural networks but showed remarkable results on sequence modeling tasks and improved accuracy also in audio effect modeling [14]. The remarkable ability of SSMs to capture long-range dependencies stems from their utilization of a specific state matrix known as the HiPPO matrix [15]. This matrix is designed to encode all the past input history, finding a map from the input to a higher dimensional space that represents the compression of the history. The matrix allows the model to be conceptualized as a convolutional model that decomposes an input signal onto an orthogonal system of smooth basis functions. In this way, the state encodes the history of the signal. Audio effects, to a different extent, present time dependencies crucial for accurately modeling them. The ability to track long dependencies in the input-output audio is beneficial in this context. In addition, The S4D model offers faster training times than classic RNNs and is defined by parameterizing its state matrix as a diagonal matrix. As RNNs, SSMs offer advantages due to their ability to capture dependencies in time-series data by utilizing internal states rather than solely relying on the input received at each iteration, such as a lengthy array of past input samples. This aspect is crucial when the goal is to achieve an implementation for live audio, which requires low input-output latency, where input refers to both audio and control parameters. We restrict the experimentation to relatively small networks with low computational complexity, allowing implementation on real-time consumer-grade digital audio systems. Experiments are based on models we have developed to exhibit minimal input-output latency, using small blocks of audio samples we typically find in computer-based audio processing dataflow as input.

The audio examples, dataset, source code, and trained models described in this paper are available online [1].

2.1 Architecture

The architecture of the artificial neural network used in this study is shown in Figure 1, and consists of fully connected (FC) layers and an S4D layer. The input is an array including the 64 most recent samples of the effect input signal $\boldsymbol{x_n}$. This choice is to give more context to the network but without introducing significant latency. The input is linearly projected with an FC layer to compute $\boldsymbol{u_n}$, which is in turn fed to the S4D layer to obtain $\boldsymbol{o_n}$. The S4D layer does not present nonlinearity; for this reason, an FC layer with a softsign activation function follows, obtaining $\hat{\boldsymbol{o}}_{\boldsymbol{n}}$. An FC layer with one unit is the output layer.

The conditioning block indicated in Figure 1 with a dashed line can be placed at different network points. In particular, before and after the S4D layer. The first variation utilizes a single conditioning layer after the linear projection performed by the linear FC layer on the input. The second presents the conditioning layer after the FC layer with nonlinearity. Lastly, the third one places conditioning on both the previously mentioned positions. These experiments should reveal the optimal positioning of the conditioning layer for the network processing of the audio samples.

2.2 Conditioning Block

The types of conditioning blocks we compare in this study are shown in Figure 2. These may include the FiLM layer or the GA layer. As mentioned earlier, the FiLM method applies an affine transformation to the vector $\boldsymbol{p_n}$ representing the conditioning information. The conditioning vector is fed to an FC layer to produce another vector $\boldsymbol{\eta_n}$ having double the size of the conditioning vector. The output vector is split into two equally sized vectors, and the following operation is then applied to $\boldsymbol{p_n}$:

$$\hat{\boldsymbol{p}}_{\boldsymbol{n}} = \boldsymbol{\eta_1 p_n} + \boldsymbol{\eta_0} \tag{1}$$

where η_1 and η_0 are the vectors obtained from the output of the FC layer. Following the projection, it is common to apply a function, such as ReLU or sigmoid, to determine

[1] https://github.com/RiccardoVib/Conditioning-Methods-for-Neural-Audio-Effects

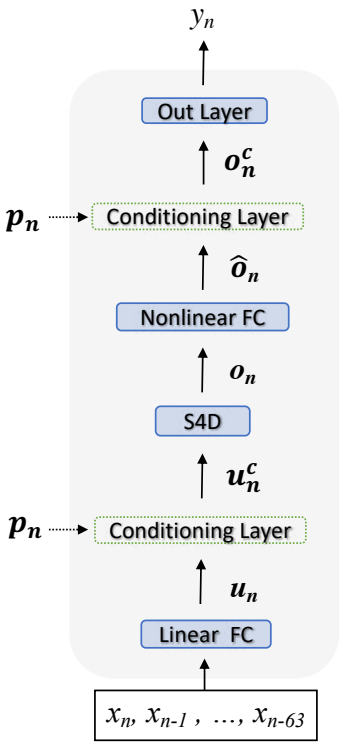

Figure 1. Architecture: the input is a vector including the 64 most recent input samples $[x_n, ..., x_{n-63}]$, which is fed to the network producing the output sample y_n. The network consists of a linear FC, S4D, and FC layer with softsign activation function adding the nonlinearity. The output layer is an FC layer with one unit. The dashed lines indicate the possible placement of conditioning layers: after the linear projection, just before the output layer, or in both positions.

the amount of information that should be passed. Our approach replaces the activation function with a Gated Linear Unit (GLU) [16]. This layer is more stable than ReLU and exhibits faster learning than sigmoid. Similarly to the previous step, the GLU layer consists of a linear FC layer that takes the FiLM output vector as input and computes a vector twice its length. The resulting output is split equally into two vectors: q_1 and q_2. A function is applied to q_2, and the resulting output is multiplied element-wise with q_1. The following equation describes the GLU layer:

$$o_n^c = q_1 \otimes softsign(q_2) \tag{2}$$

The GLU layer determines the flow of information through the network, acting as a logical gate. Typically, a sigmoid activation function is used for this purpose. However, in our approach, we introduce a softsign function instead. The softsign function controls the extent to which the control parameters should positively or negatively influence the final output. A value of 0 indicates a bypass, meaning no influence on the output. This allows for more flexibility.

The gated activation (GA), instead, is described by the following expression:

$$o_n^c = tanh(h_n) \otimes sigmoid(h_n). \tag{3}$$

where

$$h_n = x_n * W_f + p_n * W_g \tag{4}$$

with W_f, W_g representing the learned convolution weights. Lastly, a modification of the GLU, named Gated Convolutional Unit (GCU), is proposed by adding two parallel convolutional layers instead of the linear FC layer. The primary objective of this study is to explore and extend the conditioning methods to investigate how effective they are in influencing inferring networks based on the conditioning information. For this reason, we applied and compared the previously mentioned methods, the GA, FiLM-GLU, and FiLM-GCU, placing them at different stages of the architecture (*post*, *pre*, *pre-post*). All the conditioning layers are designed to have a similar number of parameters. Finally, when using the FiLM layer, we experiment also with the following nonlinear transformations:

$$\begin{aligned} f(p_n) &= \eta_1 {p_n}^3 + \eta_0 \\ f(p_n) &= \eta_1 {p_n}^5 + \eta_0 \end{aligned} \tag{5}$$

2.3 Dataset

For this study, we selected two different audio effects: overdrive distortion and dynamic range compression. These types are representative of the effects already successfully modeled using black-box neural network techniques. For each effect unit, we selected only two variable control parameters. Our primary goal is to evaluate the prediction accuracy of the networks when influenced by control parameters used as conditioning information. We collected data from effects implemented as software plugins to have enough flexibility. The data were collected using a tool that facilitates and speeds up the collection of reproducible datasets from audio effects [2]. The parameters were sampled with 3 equally spaced values across the ranges.

The selected effects are FabFilter Pro-C 2 (C1) [3], SphereComp (C2) [4], Auditory Dr Drive (OD1) [5], Otto8 Overdrive (OD2) [6]. The selected variable parameters are the ratio and threshold for the compressor, drive, and tone levels for the overdrive. The ratio range is [1.5:1, 10:1], while the threshold is between [-3, -36] dB. The overdrive effect's parameters are percentages that do not relate to measurable quantities. The selected range is [10%, 100%]. For both effects, the selected range determines at least some compression or distortion (i.e., no settings generating a fully dry output signal). Table 1 summarizes the ranges and values for each audio effect. The dataset from the selected audio effects was developed by recording the effect's output for different combinations of the selected control parameters. Three equally spaced values were recorded for each parameter, resulting in 9 combinations for each effect. The audio data was recorded at a

[2] https://github.com/stefanofasciani/DGMD
[3] https://www.fabfilter.com/products/pro-c-2-compressor-plug-in
[4] https://www.waproduction.com/plugins/view/spherecomp
[5] https://www.audiority.com/shop/dr-drive/
[6] https://www.pluginboutique.com/products/8864-OTTO8-OVERDRIVE

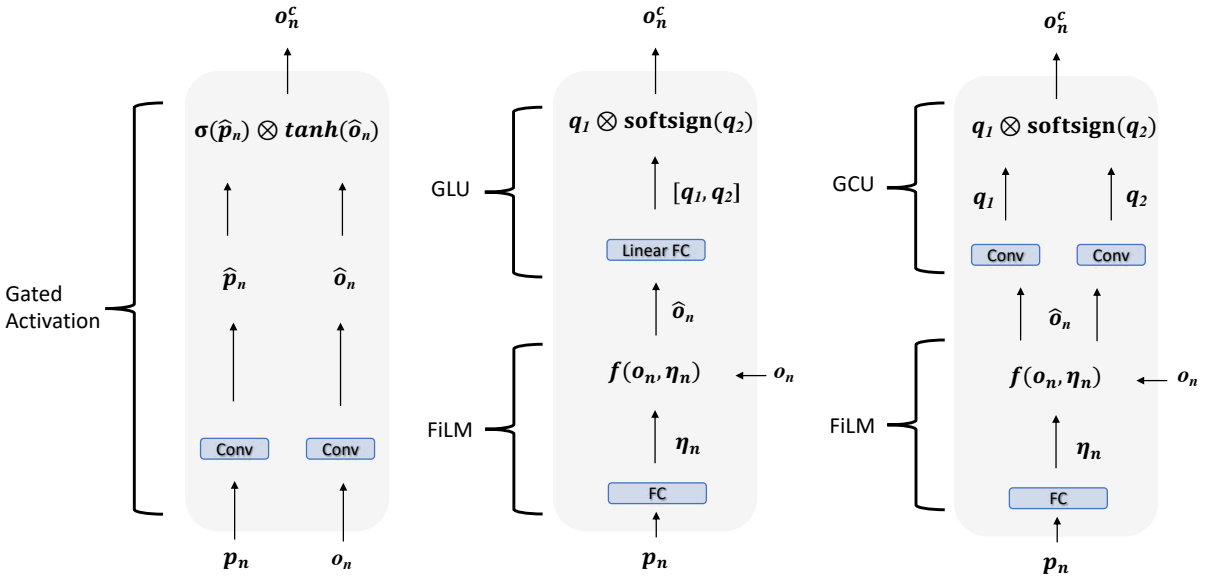

Figure 2. Conditioning layers. Gated Activation (left): the vector representing the conditioning information $\boldsymbol{p_n}$ and the input vector $\boldsymbol{o_n}$ are passed to two convolutional layers which to the outputs are applied the sigmoid function and the hyperbolic tangent, separately. Their results are element-wise multiplied together. FiLM-GLU (middle): the vector representing the conditioning information $\boldsymbol{p_n}$ fed an FC. The outcome is the vector with a length double the input size. This output vector is then split into two vectors that are used to apply a transformation (Equation 1 or 5) indicated with $f()$ to the vector $\boldsymbol{o_n}$. Subsequently, the transformed vector passes through another linear FC layer, producing another output vector double the length of its input. This output is split into two halves: the first one is applied to the softsign function, and the second one is multiplied element-wise. FiLM-GCU (right): similar to the FiLM-GLU case, but after the transformation $f()$, the vector is passed to two convolutional layers.

Software	Parameters	Range
Compressors	Ratio	[1.5:1, 4:1, 10:1]
	Threshold	[-3, -20, -36] dB
Overdrives	Drive	[10%, 45%, 100%]
	Bright/Tone	[10%, 45%, 100%]

Table 1. Software effects selected for the experiments, with parameters and their respective ranges.

sampling rate of 48 kHz. For each combination, the effects were fed with a mono input signal with a duration of 2 minutes. The input signal includes a variety of sounds, such as frequency sweeps covering a range of 20 Hz to 20 kHz, white noises with increasing amplitudes (both linear and logarithmic), recordings of instruments such as guitar, bass, drums (both loops and single notes), vocals, piano, pad sounds, and sections from various electronic and rock songs. The control parameters of each effect are mapped to the range $[0, 1]$ before using them as network conditioning information.

The C1 and OD1 datasets were used for the experiments investigating the conditioning methods and the placements, while the latter two were employed for further experiments aiming to confirm the findings of the previous experiments.

2.4 Experimenting and Learning

The models are trained for 60 epochs and use the Adam [17] optimizer with a gradient norm scaling of 1 [18]. The training was stopped earlier in case of no reduction of validation loss for 10 epochs. We design a time-based schedule for the learning rate as follows:

$$lr = \frac{LR}{e^4} \tag{6}$$

with LR the initial learning rate, that in our case is $3 \cdot 10^{-4}$ and e the number of the current epoch. The loss function used is the mean square error (MSE) and is computed using the model's weights that minimize the validation loss throughout the training epochs. The input signal is split into segments of 600 samples (equivalent to 12.5 ms) to be processed before updating the weights. The models are evaluated using the MSE, error-to-signal-ratio (ESR), and multi-resolution short-time Fourier transform (STFT) error with resolutions of [256, 512, 1024] samples.

Finally, the dataset is split into 80% for the training set, 10% for the validation set, and the remaining 10% for the test set. The data was divided into an $80 - 10 - 10\%$ split at the individual recording level, ensuring that all control parameter combinations were equally represented in each subset and that they were used consistently for both training and evaluating the model. The test set includes audio unseen during training, while all parameter combinations are included in the training set. We do not evaluate the

models' ability to predict audio for conditioning parameters never seen during training. Minor manual adjustments are made to ensure that splitting points for audio recordings fall within segments of silence.

3. RESULTS

Tables 2 and 3 report the prediction errors related to different types of conditioning layers and placements. These errors relate to models trained using the C1 and OD1 datasets. In both cases, the baseline model, as expected, shows lower performances. When considering the compressor, in Table 2, the GA method presents larger errors in all cases. The affine transformation of the FiLM method results is beneficial, especially if placed after the S4D layer, and even if with smaller differences, using fully-connected layers in the gated unit leads to generally better prediction accuracy. These performances can also be observed in Figure 3, where the compression curves of the target and the model's predictions are shown. In particular, the curves refer to the three different ratios against a threshold set to -20 dB. The FiLM layer shows the best performance, especially when encountering heavier compression. On the other hand, GA particularly underestimates the compression in all the cases. Conversely, for the overdrive, the GA performs generally better than when applied to the compressor. In addition, using convolution in the gated unit after the FiLM operation leads to the best results in all cases, indicating convolution is particularly beneficial in overdrive modeling. The best accuracy is found in the FiLM-GCU case when placed before the S4D layer, but it presents a small gap with the *post* placements. In addition, placing the conditioning after the S4D layer led to general improvements when considering the other methods. This suggests that also, in the overdrive case, it is more beneficial for the networks to use the information given by the control parameters to project the output of the S4D layer and determine the extent to which this information influences the final output rather than influencing the inference of the recurrent layer based on the control parameters. Lastly, we note that the *pre-post* configuration leads to the worst performance for the overdrive dataset. From Figure 4, we observe that the three methods perform similarly, predicting the conditioning case in the middle of the range. GA improves when encountering heavy distortion and degenerates in the light distortion case, while FiLM shows the opposite trend. FiLM-based layers showed the best overall modeling accuracy in both cases, especially if placed after the S4D layer. GLU and GCU performance depends on the dataset. The overdrive type works better when convolutional layers are used, while the compressor shows better accuracy using a linear FC layer. The same trend is evident when training models with C2 and OD2 datasets, as visible in Table 4. These experiments are performed only in the *post* configuration. Again, convolution-based conditioning resulted in more benefits for the overdrive, while GLU remains a better candidate for compressors. The choice may depend on the influence of the various parameters on the output sound, particularly their non-linear relationship between control parameters and their

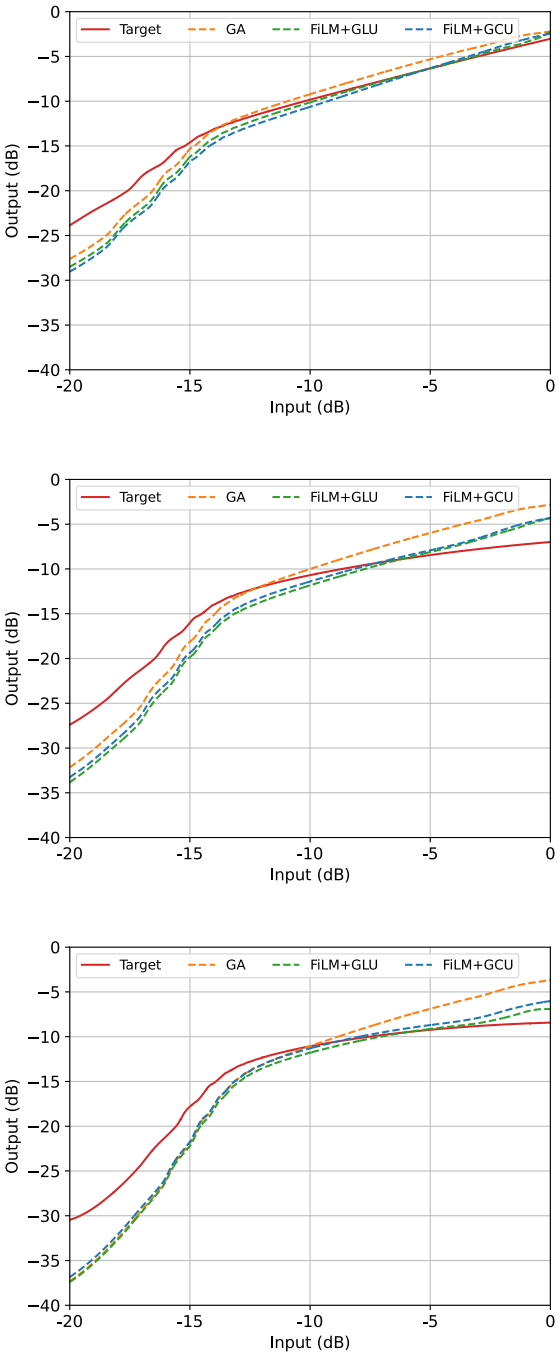

Figure 3. Compression curves of targets and predictions using the three different conditioning methods: GA, FiLM-GLU, and FiLM-GCU. The set threshold is -20 dB, while the ratio is 1.5:1 (top), 4:1 (middle), and 10:1 (bottom). The target example was not present in the training set. The plots refer to the C1 dataset.

impact on the effect's sound alteration process. To further investigate this aspect, we extend the transformation inside the FiLM layer with equations of order greater than one and odd. Even orders are discarded because of their symmetrical nature. The applied transformations are listed in Equation 5. Table 5 reports the errors, and as expected, in the case of the overdrive, a nonlinear transformation of or-

Method	Stage	MSE	ESR	STFT
Baseline	-	$8.27 \cdot 10^{-3}$	$2.98 \cdot 10^{-1}$	$6.84 \cdot 10^{-1}$
GA	Post	$7.23 \cdot 10^{-3}$	$2.60 \cdot 10^{-1}$	$5.51 \cdot 10^{-1}$
FiLM-GLU	**Post**	$\mathbf{2.00 \cdot 10^{-3}}$	$\mathbf{7.24 \cdot 10^{-2}}$	$\mathbf{1.82 \cdot 10^{-1}}$
FiLM-GCU	**Post**	$2.70 \cdot 10^{-3}$	$9.73 \cdot 10^{-2}$	$3.15 \cdot 10^{-1}$
GA	Pre	$7.92 \cdot 10^{-3}$	$2.85 \cdot 10^{-1}$	$8.41 \cdot 10^{-1}$
FiLM-GLU	Pre	$4.49 \cdot 10^{-3}$	$1.62 \cdot 10^{-1}$	$5.02 \cdot 10^{-1}$
FiLM-GCU	Pre	$\mathbf{4.42 \cdot 10^{-3}}$	$\mathbf{1.59 \cdot 10^{-1}}$	$\mathbf{4.81 \cdot 10^{-1}}$
GA	**Pre-Post**	$6.81 \cdot 10^{-3}$	$2.45 \cdot 10^{-1}$	$7.34 \cdot 10^{-1}$
FiLM-GLU	Pre-Post	$\mathbf{3.37 \cdot 10^{-3}}$	$\mathbf{1.21 \cdot 10^{-1}}$	$\mathbf{4.00 \cdot 10^{-1}}$
FiLM-GCU	Pre-Post	$5.51 \cdot 10^{-3}$	$1.98 \cdot 10^{-1}$	$6.23 \cdot 10^{-1}$

Table 2. MSE, ESR, and STFT errors referring to the different conditioning methods and placements, if applicable, and for the compressor C1 case. In bold are indicated the best results for each method and placement.

Method	Stage	MSE	ESR	STFT
Baseline	-	$2.99 \cdot 10^{-2}$	$4.59 \cdot 10^{-1}$	$7.17 \cdot 10^{-1}$
GA	**Post**	$7.41 \cdot 10^{-3}$	$1.13 \cdot 10^{-1}$	$3.79 \cdot 10^{-1}$
FiLM-GLU	**Post**	$7.92 \cdot 10^{-3}$	$1.21 \cdot 10^{-1}$	$\mathbf{3.13 \cdot 10^{-1}}$
FiLM-GCU	Post	$\mathbf{7.17 \cdot 10^{-3}}$	$\mathbf{1.10 \cdot 10^{-1}}$	$3.31 \cdot 10^{-1}$
GA	Pre	$7.93 \cdot 10^{-3}$	$1.21 \cdot 10^{-1}$	$4.01 \cdot 10^{-1}$
FiLM-GLU	Pre	$8.24 \cdot 10^{-3}$	$1.26 \cdot 10^{-1}$	$3.77 \cdot 10^{-1}$
FiLM-GCU	**Pre**	$\mathbf{6.86 \cdot 10^{-3}}$	$\mathbf{1.05 \cdot 10^{-1}}$	$\mathbf{3.28 \cdot 10^{-1}}$
GA	Pre-Post	$9.24 \cdot 10^{-3}$	$1.41 \cdot 10^{-1}$	$4.05 \cdot 10^{-1}$
FiLM+GLU	Pre-Post	$8.39 \cdot 10^{-3}$	$1.28 \cdot 10^{-1}$	$3.97 \cdot 10^{-1}$
FiLM-GCU	Pre-Post	$\mathbf{8.00 \cdot 10^{-3}}$	$\mathbf{1.22 \cdot 10^{-2}}$	$\mathbf{3.65 \cdot 10^{-1}}$

Table 3. MSE, ESR, and STFT errors referring to the different conditioning layers and placements, if applicable, and for the overdrive OD1 case. In bold are indicated the best results for each method and placement.

Dataset	Method	Stage	MSE	ESR	STFT
C2	GA	Post	$8.31 \cdot 10^{-4}$	$8.71 \cdot 10^{-2}$	$3.49 \cdot 10^{-1}$
	FiLM-GLU	Post	$\mathbf{6.72 \cdot 10^{-4}}$	$\mathbf{7.04 \cdot 10^{-2}}$	$2.76 \cdot 10^{-1}$
	FiLM-GCU	Post	$7.11 \cdot 10^{-4}$	$7.45 \cdot 10^{-2}$	$\mathbf{2.75 \cdot 10^{-1}}$
OD2	GA	Post	$1.82 \cdot 10^{-3}$	$1.87 \cdot 10^{-1}$	$\mathbf{3.92 \cdot 10^{-1}}$
	FiLM-GLU	Post	$2.01 \cdot 10^{-3}$	$2.06 \cdot 10^{-1}$	$4.46 \cdot 10^{-1}$
	FiLM-GCU	Post	$\mathbf{1.71 \cdot 10^{-3}}$	$\mathbf{1.76 \cdot 10^{-1}}$	$4.06 \cdot 10^{-1}$

Table 4. MSE, ESR, and STFT errors referring to the C2 and OD2 datasets, with the different conditioning layers and *post* configuration. In bold are indicated the best results for each method and placement.

ders 3 and 5 results beneficial. A nonlinear function more accurately represents the distortion introduced as a function of the two parameters. However, when an equation of the fifth order in the opposite degenerates the performances in the model trained with the OD1 dataset, a fifth-order transformation may provide excessive steepness in the response of the conditioning layer for the OD1. Still, the behavior is the opposite for the OD2 dataset, which improves when increasing the order. Differently, the compressors do not benefit from nonlinear transformations, which is consistent with the previous results where GLU showed the best modeling accuracy.

4. CONCLUSION

Artificial neural networks have been used to model various audio effects, showing sufficient accuracy in most cases. At the same time, real-time and low-latency capability is still a challenge as many proposed models use large artificial neural networks. Most models are also limited in emulating the variable response of an audio effect, which generally occurs when tuning the parameters exposed to users. How to feed parameter-related information into the modeling networks is fundamental to achieve an accurate fully-conditioned emulation. In this study, we investigated this aspect, starting from an architecture based on a state-space layer. We compared two conditioning methods, the first based on gated activation and the other on feature-wise linear modulation. The latter is commonly used for

Dataset	Method	Order	MSE	ESR	STFT
C1	FiLM-GLU	**1**	$\mathbf{2.00 \cdot 10^{-3}}$	$\mathbf{7.24 \cdot 10^{-2}}$	$\mathbf{1.82 \cdot 10^{-1}}$
-	FiLM-GLU	3	$2.21 \cdot 10^{-3}$	$7.99 \cdot 10^{-2}$	$2.15 \cdot 10^{-1}$
-	FiLM-GLU	5	$2.51 \cdot 10^{-3}$	$9.05 \cdot 10^{-2}$	$2.64 \cdot 10^{-1}$
-	FiLM-GCU	**1**	$\mathbf{2.70 \cdot 10^{-3}}$	$\mathbf{9.73 \cdot 10^{-2}}$	$\mathbf{3.15 \cdot 10^{-1}}$
-	FiLM-GCU	3	$3.98 \cdot 10^{-3}$	$1.43 \cdot 10^{-1}$	$4.65 \cdot 10^{-1}$
-	FiLM-GCU	5	$5.20 \cdot 10^{-3}$	$1.87 \cdot 10^{-1}$	$4.44 \cdot 10^{-1}$
C2	FiLM-GLU	**1**	$\mathbf{6.72 \cdot 10^{-4}}$	$\mathbf{7.04 \cdot 10^{-2}}$	$\mathbf{2.76 \cdot 10^{-1}}$
-	FiLM-GLU	3	$8.88 \cdot 10^{-4}$	$9.31 \cdot 10^{-2}$	$3.39 \cdot 10^{-1}$
-	FiLM-GLU	5	$8.69 \cdot 10^{-4}$	$9.11 \cdot 10^{-2}$	$3.40 \cdot 10^{-1}$
-	FiLM-GCU	**1**	$\mathbf{7.11 \cdot 10^{-4}}$	$\mathbf{7.45 \cdot 10^{-2}}$	$\mathbf{2.76 \cdot 10^{-1}}$
-	FiLM-GCU	3	$7.77 \cdot 10^{-4}$	$8.15 \cdot 10^{-2}$	$3.03 \cdot 10^{-1}$
-	FiLM-GCU	5	$9.19 \cdot 10^{-4}$	$9.64 \cdot 10^{-2}$	$3.60 \cdot 10^{-1}$
OD1	FiLM-GLU	1	$7.92 \cdot 10^{-3}$	$1.21 \cdot 10^{-1}$	$3.13 \cdot 10^{-1}$
-	FiLM-GLU	**3**	$\mathbf{7.32 \cdot 10^{-3}}$	$\mathbf{1.12 \cdot 10^{-1}}$	$\mathbf{3.10 \cdot 10^{-1}}$
-	FiLM-GLU	5	$9.08 \cdot 10^{-3}$	$1.39 \cdot 10^{-1}$	$3.23 \cdot 10^{-1}$
-	FiLM-GCU	**1**	$\mathbf{7.17 \cdot 10^{-3}}$	$\mathbf{1.10 \cdot 10^{-1}}$	$\mathbf{3.31 \cdot 10^{-1}}$
-	FiLM-GCU	3	$7.57 \cdot 10^{-3}$	$1.16 \cdot 10^{-1}$	$3.55 \cdot 10^{-1}$
-	FiLM-GCU	5	$8.96 \cdot 10^{-3}$	$1.23 \cdot 10^{-1}$	$3.67 \cdot 10^{-1}$
OD2	FiLM-GLU	1	$2.01 \cdot 10^{-3}$	$2.06 \cdot 10^{-1}$	$4.46 \cdot 10^{-1}$
-	FiLM-GLU	**3**	$\mathbf{1.30 \cdot 10^{-3}}$	$\mathbf{1.34 \cdot 10^{-1}}$	$\mathbf{3.07 \cdot 10^{-1}}$
-	FiLM-GLU	5	$1.38 \cdot 10^{-3}$	$1.42 \cdot 10^{-1}$	$3.24 \cdot 10^{-1}$
-	FiLM-GCU	1	$1.71 \cdot 10^{-3}$	$1.76 \cdot 10^{-1}$	$4.06 \cdot 10^{-1}$
-	FiLM-GCU	3	$1.28 \cdot 10^{-3}$	$1.31 \cdot 10^{-1}$	$3.07 \cdot 10^{-1}$
-	FiLM-GCU	**5**	$\mathbf{1.19 \cdot 10^{-3}}$	$\mathbf{1.22 \cdot 10^{-1}}$	$\mathbf{3.05 \cdot 10^{-1}}$

Table 5. MSE, ESR, and STFT errors referring to FiLM-GLU and FiLM-GCU conditioning methods, with different transformations as listed in Equation 5. In all cases, the conditioning layer is placed after the S4D layer.

conditioning neural networks in other application domains. The technique includes performing an affine transformation via learnable coefficients based on conditioning information. We improved the methods by adding gated linear and gate convolutional units. The study is based on two types of audio effects, distortion, and compression, which have proven to be accurately modeled using artificial neural networks. Distortion and compression have been chosen due to their significantly different sound-altering characteristic. Results showed the FiLM outperforms other approaches, and its placement after the S4D layer leads to more accurate emulations of the audio effect response. Both are compared when placed at different stages of the architecture and used as baseline feeding the conditioning information concatenated to the input samples. The gated linear unit is beneficial in the compressor case, while the convolutional counterpart is more suitable for the overdrive. In addition, the FiLM layer considering nonlinear transformations is beneficial for the overdrive effect, which parameters may present a stronger nonlinear relationship with the associated sonic response of the effect. In particular, a nonlinear transformation of the third order led to better results for the overdrive dataset, while no improvements were noted for the compressor.

The study will be extended to various effects in future work, incorporating control parameters with different characteristics. In particular, this study does not include parameters changing the temporal profile of the compression, namely the attack and release time. In addition, nonlinear mappings in the conditioning layer have been found beneficial when modeling overdrive. Future studies will be extended to include more refined transformations.

5. REFERENCES

[1] D. S. Mendoza, *Emulating electric guitar effects with neural networks*. Barcelona, Spain: Universitat Pompeu Fabra, 2005.

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

[3] E.-P. Damskägg, L. Juvela, V. Välimäki *et al.*, "Real-time modeling of audio distortion circuits with deep learning," in *Proc. of 16th Int. Sound and Music Computing Conf. (SMC)*, Malaga, Spain, 2019.

[4] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *Speech Synthesis Workshop*.

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

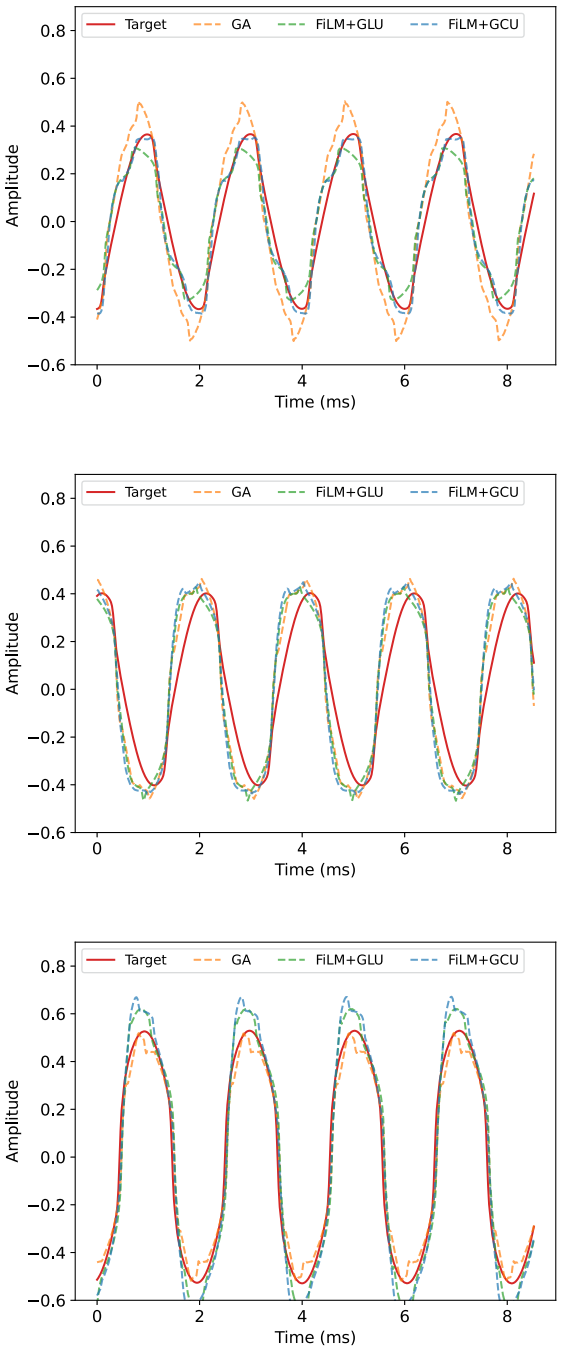

Figure 4. Target distorted 1000 Hz sine wave against the respective prediction using the three different conditioning methods: GA, FiLM-GLU, and FiLM-GCU. The drive knob is set to 45%, while the bright knob is to 10% (top), 45% (middle), and 100% (bottom). The sine wave is not present in the training set. The plots refer to models trained using the OD1 dataset.

[6] A. Wright, E.-P. Damskägg, and V. Valimaki, "Real-time black-box modelling with recurrent neural networks," in *Proc. of 22nd Int. Conf. Digital Audio Effects (DAFx)*, Birmingham, UK, 2019.

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

[8] R. Simionato and S. Fasciani, "Deep learning conditioned modeling of optical compression," in *Proc. of 25th Int. Conf. Digital Audio Effects (DAFx)*, Vienna, Austria, 2022.

[9] ——, "Fully conditioned and low-latency black-box modeling of analog compression," in *Proc. of 26th Int. Conf. Digital Audio Effects (DAFx)*, Copenaghen, Denmark, 2023.

[10] C. J. Steinmetz and J. D. Reiss, "Efficient neural networks for real-time analog audio effect modeling," in *152nd Audio Engineering Society Convention*, The Hague, Netherlands, 2022.

[11] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville, "Film: Visual reasoning with a general conditioning layer," in *Proc. of 2018 Conference on Artificial Intelligence (AAAI)*, New Orleans, Louisiana, USA, 2018.

[12] A. Gu, K. Goel, and C. Ré, "Efficiently modeling long sequences with structured state spaces," *Proc. of 2022 Int. Conf. Learning Representations (ICLR)*, 2022.

[13] A. Gu, K. Goel, A. Gupta, and C. Ré, "On the parameterization and initialization of diagonal state space models," *Proc. of 36th Int. Conf. Neural Information Processing Systems (NIPS)*, 2022.

[14] R. Simionato and S. Fasciani, "Comparative study of recurrent neural networks for virtual analog audio effects modeling," 2024.

[15] A. Gu, T. Dao, S. Ermon, A. Rudra, and C. Ré, "Hippo: Recurrent memory with optimal polynomial projections," *Proc. of 34th Int. Conf. Neural Information Processing Systems (NIPS)*, 2020.

[16] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language modeling with gated convolutional networks," in *International conference on machine learning*, Sydney, Australia, 2017.

[17] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *Proc. of 2015 Int. Conf. Learning Representations (ICLR)*, 2015.

[18] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International conference on machine learning*, Atlanta, USA, 2013.