TOWARDS NEURAL EMULATION OF VOLTAGE-CONTROLLED OSCILLATORS

Riccardo Simionato

Dept. of Musicology
University of Oslo
Oslo, Norway
riccardo.simionato@imv.uio.no

Stefano Fasciani

Dept. of Musicology
University of Oslo
Oslo, Norway
stefano.fasciani@imv.uio.no

ABSTRACT

Machine learning models have become ubiquitous in modeling analog audio devices. Expanding on this line of research, our study focuses on Voltage-Controlled Oscillators of analog synthesizers. We employ black box autoregressive artificial neural networks to model the typical analog waveshapes, including triangle, square, and sawtooth. The models can be conditioned on wave frequency and type, enabling the generation of pitch envelopes and morphing across waveshapes. We conduct evaluations on both synthetic and analog datasets to assess the accuracy of various architectural variants. The LSTM variant performed better, although lower frequency ranges present particular challenges.

1. INTRODUCTION

Analog electronic circuits play a crucial role in a significant category of musical equipment, like synthesizers and audio effects, offering a unique sound that enhances their appeal. The field of virtual analog modeling aims to digitally replicate these analog audio effects and synthesizers. Analog synthesizers create sound using Voltage-Controlled Oscillators (VCOs), which generate electrical signals through rapidly changing voltages, producing a variety of waveshapes with different timbres, including the common sawtooth, rectangular-pulse, and triangular waveforms. The frequency is controlled by an input voltage, typically adhering to the 1V/Oct tracking standard. The sound's timbre is shaped by Voltage-Controlled Filters (VCFs), and a Voltage-Controlled Amplifier (VCA) dynamically controls the synth voice. This method, known as subtractive synthesis, is characteristic of the 'East Coast' synthesis approach. In contrast, synthesizers using the 'West Coast' synthesis approach take an additive synthesis-like method. They incorporate a wavefolding circuit, which enriches waveforms by inverting or folding parts of the waveform to add harmonic content. In this approach, the synth voice is controlled through a Low Pass Gate (LPG), a circuit which combines filter and amplifier-like characteristics in order to simultaneously affect a sound's timbre and its loudness.

In these signal flows, VCO circuits are the primary signal generators, converting a voltage that represents frequency into an audio signal oscillating at that frequency. Noise generators can also be used to add extra sound coloration. Although VCO waveshapes are relatively simple, they are not band-limited. Thus, the primary challenge in their digital emulation is developing computationally efficient algorithms that minimize aliasing distortion To address this issue, several methods have been proposed, including wavetable synthesis [1], discrete summation formulas [2], frequency-domain methods [3], bandlimited impulse trains (BLITs) [4], bandlimited step functions (BLEPs) [5], differentiated polynomial waveforms (DPWs) [6], polynomial transition regions (PTRs) [7], and simple oversampling. The choice of algorithm depends on the specific application's efficiency and aliasing reduction requirements.

While artificial neural networks (ANNs) have been successfully employed to model analog audio effects over the past decade, their application in modeling audio oscillators, particularly VCOs, remains limited and underexplored. For instance, a hybrid approach employing a neural oscillator based on a Deep Convolutional Neural Network was designed to generate wavetable content from timbral descriptors [8].

This work investigates the use of ANNs to emulate VCOs that generate multiple waveforms at arbitrary frequencies. The approach involves training specifically designed autoregressive ANNs with data recorded from real VCOs, as preliminary tests suggest this approach is more effective than other generative models. The architecture generates audio samples one at a time, minimizing latency between control data—such as frequency and waveshape type—and audio output, theoretically supporting real-time interactive sound synthesis. Conceptually, this approach is similar to wavetable synthesis, where waveforms from real VCOs are stored and scanned to reproduce arbitrary frequencies. However, the ANN model learns how VCOs generate signals across all frequencies, enhancing synthesis by capturing waveshape variations due to the nonlinear characteristics of analog VCOs. This leads to improved accuracy, despite the high computational complexity involved. Furthermore, as a data-driven black-box synthesis approach, its performance concerning aliasing is predominantly influenced by any aliasing present in the recordings used to train the ANN, rather than by limitations or approximations of the synthesis algorithm.

Popular examples of autoregressive models are the Wavenet [9] and the SampleRNN [10], which were employed for various audio tasks. We designed autoregressive models based on the most popular types of neural layers: a fully connected (FC), Convolutional, Recurrent, and the Feature-wise Linear Modulation (FiLM) [11]. We show how the method allows continuous frequency control and morphing across waveshapes.

The remainder of the paper is organized as follows: Section 2 outlines the methodology and dataset used for our research. Section 3 presents and discusses the findings, and Section 4 provides concluding remarks.

2. METHODS

In this paper, we explore autoregressive neural networks for VCO emulation. For the experiments, we collected audio at different frequencies from the VCOs of an analog subtractive synthesizer. In addition, to further evaluate the neural network's ability to generate signals at unseen frequencies, we have collected equivalent synthetic datasets. When training the system, we use an input parameter to inform the network of the frequency to generate. This parameter allows for continuous pitch modulations. An additional parameter can be used to inform the waveshapes to be generated, allowing morphing among those available in the dataset.

2.1. Architecture

The architecture, illustrated in Figure 1, is composed of five layers: Backbone, Conditioning, Compression, State, and Output. The Backbone layer can be implemented using either temporal convolutional neural networks (TCNs) or recurrent neural networks, including the Elman network (vanilla RNN) and variants such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU). The Conditioning layer integrates a Feature-wise Linear Modulation (FiLM) [11] followed by a Gated Linear Unit (GLU) [12], as in [13]. The Compression layer consists of a FC layer, while the State layer includes two FCs, and the Output layer has one FC. Both the State and Output layers use a hyperbolic tangent activation function. The Compression layer is composed of four neurons, whereas the State layer contains the same number of neurons as the Backbone layer. The Output layer has a single neuron.

The Compression and State layers are added to the architecture only when the Backbone layer includes recurrent neural network configurations, as they feature internal states. The Compression layer reduces the number of recurrent steps performed by the subsequent layer, significantly lowering computational costs. The State layer recalculates the initial internal states of the recurrent layer each time a new input buffer is fed to the Compression layer, initiating the recurrent steps necessary to generate a new output sample. This approach differs from truncated back-propagation through time, where states would persist beyond a recurrent step count equal to the dimensionality following the Compression layer. While the architecture does not guarantee that the Compression layer functions along the time axis, experiments demonstrate that accuracy is either comparable to or better with the Compression layer than without it.

The network operates in an autoregressive manner, using an input buffer as a shift register, which includes the past T generated audio samples. This approach provides the network with context to generate an audio signal that is continuous and consistent. In initial experiments, we set this value to approximately 5% of the number of samples in the longest period in the datasets, while in further experiments, we investigate the impact of this architectural parameter on the overall learning.

The frequency and shape parameters, both normalized between 0 and 1, serve as inputs to both the State layer, if present, and the Conditioning layer. If the Backbone layer is an LSTM, the State layer comprises two parallel FC layers, corresponding to the LSTM's two types of states. For GRU and RNN architectures, the State layer consists of a single FC layer, as these networks have only one type of state. The number of units in the State layer matches the number of units in the Backbone layer, ensuring compatibility between the internal blocks of the architecture.

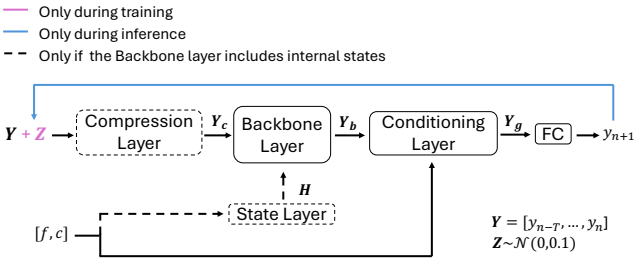

Figure 1: *The architecture uses a shift buffer that stores the last T input audio samples,* $[y_{n-T}, ..., y_n]$ *, which the network processes to generate the next output sample,* y_{n+1}*. These samples are fed into the Backbone layer, which is comprised of either a convolutional or recurrent layer. Two parameters,* f *(frequency) and* c *(shape), condition the wave generation. In recurrent-based architectures, these parameters help compute the internal states* H *using an FC network, while another FC layer compresses the input vector, reducing the number of recurrent steps required to compute the output and therefore decreasing the computational complexity. A final linear FC layer produces the output sample* y_{n+1}*. During training, white noise* $Z \sim \mathcal{N}(0, 0.1)$ *is added to the input signal to improve robustness during inference.*

2.2. Conditioning Techniques

The Conditioning layer is positioned after the Backbone layer, as this arrangement is more effective [13] and consists of a FiLM followed by a GLU having softsign as an activation function. The FiLM layer applies an affine transformation to the vector representing the conditioning information:

$$\boldsymbol{k} = \boldsymbol{f_1}(\boldsymbol{Y_b})\boldsymbol{w} + \boldsymbol{f_2}(\boldsymbol{Y_b}) \tag{1}$$

where $\boldsymbol{w}$ is the output vector of the backbone layer, $\boldsymbol{k}$ the result from the FiLM operations, and $\boldsymbol{f_1}$ and $\boldsymbol{f_2}$ are two vectors obtained from splitting the output of a linear FC layer, having many units double of the dimension of $\boldsymbol{w}$, fed with $[f, c]$.

A GLU layer follows the FiLM block to determine the information that should flow. Similarly to FiLM, the GLU layer consists of a linear FC layer that takes the FiLM output vector $\boldsymbol{k}$ and computes a vector twice its length. The resulting output is split equally into two vectors $\boldsymbol{g_1}$ and $\boldsymbol{g_2}$, to perform the following operation:

$$\boldsymbol{Y_g} = \boldsymbol{g_1}(\boldsymbol{k}) \odot \text{softsign}(\boldsymbol{g_2}(\boldsymbol{k})) \tag{2}$$

The GLU layer determines the flow of information through the network, acting as a logical gate. The softsign activation function controls the extent to which the control parameters should influence the final output, and it is defined as follows:

$$\text{softsign}\,(x) = \left(\frac{x}{|x| + 1}\right) \tag{3}$$

2.3. Datasets

The datasets include recordings from a real analog VCO and a digital emulation, featuring data from the Korg Monologue [1] and the Ableton Live Operator [2]. Three different waveshapes have been

[1] https://www.korg.com/de/products/synthesizers/monologue/

[2] https://www.ableton.com/en/packs/operator/

collected from the VCO of the Monologue: triangle, square, and sawtooth, while for the Operator, sinusoidal and sawtooth waveshapes. In the rest of the paper, we refer to the Monologue dataset as 'Analog' and to the Operator dataset as 'Synthetic'.

To collect the data from the hardware analog synthesizer, we used a MOTU M4 audio interface to record the output generated by triggering the integrated sequencer for all 72 pitches in the range $[E_0, E_6]$. All the VCO signal post-processing and modulations were deactivated in order to record the clean VCO output. This process was repeated for each waveshape. The audio was recorded at 96 kHz with a total duration of 8 seconds per pitch and subsequently downsampled to 48 kHz when training the models. The recordings were postprocessed, obtaining a separate audio file for each pitch and waveshape value, with their first sample aligned with phase 0. The length of the files was limited to $262,144$ samples at 48 kHz, equivalent to 5.41 seconds.

To collect the data from the Operator, we utilized the sequencing and recording features of the hosting Digital Audio Workstation, Ableton Live. Also in this case, the setting of the Operator was tuned to obtain an output including the clean virtual VCO signal. In this case, the generation was forced to start at phase 0, and the VCO frequency was controlled to generate signals between 100 and 300 Hz, with incremental steps of 1 Hz. This was repeated for each waveshape. The 'Synthetic' dataset presents a finer and linear pitch resolution across the examples, while in the 'Analog' dataset, the frequency increments are nonlinear and given by $f_p = 2^{p/12}$ Hz, where p represents the pitch index between 0 and 71. The amplitude of all recorded examples is individually normalized between 1 and -1. The waves' fundamental frequency for the analog dataset ranges between 20 Hz and 1318 kHz, meaning that the longest wavelength consists of $2,400$ samples, while the shortest is 36. For the synthetic, the collected frequency range is $100 - 300$ Hz, meaning the longest wavelength consists of 480 samples, and the shortest is 360.

2.4. Training, Validation and Test set

We have created four different training, validation, and test sets using the 'Analog' datasets to evaluate different aspects of the VCO neural model. First, we have created variations on the individual example durations. In particular, we truncated the recordings to $(2^{[15,16,17,18]})$ samples. The different dataset sizes were used to explore the extent to which the model learn with different example durations, which range between 0.68 and 5.41 seconds. Secondly, from each recording associated with a frequency-shape pair, 10% of the data is extracted, and the first 3276 samples are allocated to the test set, while the remaining are allocated to the validation set. This approach enables the creation of a test set of identical size, allowing for a fair comparison on the generation of equal-length signals. These wave samples originate from phases other than zero, ensuring a variety of waveshape starting points. This processing was applied to each 'Analog' dataset, which will be referred to as 'Analog-15', 'Analog-16', 'Analog-17', and 'Analog-18', with the numerical suffix indicating the example lengths.

We employed the 'Synthetic' datasets to generate five distinct variants. For each waveshape in these datasets, training sets were designed by including varying amounts of frequency examples. This was achieved by systematically excluding frequency examples at specific intervals within the frequency range [100, 300] Hz, which was sampled at steps of 1 Hz. Specifically, we removed an example every 2, 4, 8, 16, or 32 Hz starting from 100 Hz, thereby creating five unique training sets. These intervals corresponded to the exclusion of 50%, 25%, 12.5%, 6.25%, and 3.125% of the dataset from the training sets, respectively. The test set, shared across all five variants, includes 3276 samples of examples taken every 32 Hz, which is 3.125% of the dataset, which is never part of any of the five training sets described above. The validation set was constructed by slicing (5%) of each example from the training set, ensuring that each example starts from a phase other than zero, providing diversity to the validation process. In this case, the datasets are named 'Synthetic-2', 'Synthetic-4', 'Synthetic-8', 'Synthetic-16', and 'Synthetic-32', with the numerical suffix indicating the step value used to exclude frequency examples from the training set, as explained above. These sets are used to investigate the models' ability to generate audio at unseen frequencies. Table 1 summarizes the datasets.

Table 1: *Summary of the designed datasets, including the type of dataset, the individual example durations in samples, and shapes included in the dataset. Sin, Tri, Squ, and Saw refer to sine, triangle, square, and sawtooth waveshapes, respectively.*

Dataset	Durations	Waveshapes
Analog-[15, 16, 17, 18]	$2^{[15,16,17,18]}$	Tri, Squ, Saw
Synthetic-[2, 4, 8, 16, 32]	2^{18}	Sin, Saw

2.5. Experiments

The variants considered for the Backbone layer include vanilla RNN (Elman network), GRU, LSTM, and TCN. Experiments are carried out using various configurations, with unit numbers of $[16, 32, 64]$ and T values of $[64, 96, 128, 256]$ samples, which define the length of the input buffer. The TCN-based model features 4 layers, each with a kernel size of 3 and a dilation rate of 2^l, where l represents the layer number. On the other hand, RNN, GRU, and LSTM variants include only one layer.

To enhance the model's robustness against varying error levels encountered during inference, Gaussian noise $Z \sim \mathcal{N}(0, 0.1)$ is added to the input to provide robustness against errors that arise between predictions and ground truth. These errors can accumulate over inference iterations, generating one audio sample each, and impact subsequent predictions. By incorporating noise, the network becomes less dependent on error-free previous samples for generating the next output sample. Instead, it learns to maintain the integrity of the shape even when dealing with noisy samples within the input buffer.

The evaluation employs normalized mean squared error (NMSE) and single-resolution fast Fourier transform error (FFTE), using a resolution of 256. These metrics are calculated based on the model's weights that have minimized the validation loss during training epochs, ensuring an accurate assessment of model performance throughout the training process.

The initial experiments aimed to identify the architecture with the best accuracy, assessed through loss metrics and visual inspection of the predicted waveforms. Consequently, we trained all models using the 'Analog-18' dataset, waveshapes subset separately, training one model for each waveshape at a time. In this case, the c (shape) input is removed from the model as only the frequency is used as input. To ensure comparable expressivity, architectures were designed to have a similar number of parameters, as detailed in Table 2.

Table 2: *Units and number of parameters for the architectures used in the comparative experiments.*

Model	Units	Parameters	Model	Units	Parameters
TCN	38	25,309	RNN	92	26,793
GRU	70	26,149	LSTM	64	26,181

After identifying the best-performing architecture, we conducted further experiments exclusively with the two top-performing architectures. We explored the impact of training set examples' lengths on the performance by training the models on the 'Analog-15', 'Analog-16', 'Analog-17', and 'Analog-18' datasets. Also for this experiment, the c input is removed from the models, and training was performed for one waveshape at a time.

Given the significant number of audio samples per period in the lowest octave of the 'Analog' dataset, we further investigate how different sizes of T affect the model's learning for the lower frequencies when trained in a dataset including all the possible frequencies. This involved training the model for different values of T, with the 'Analog-18' dataset, utilizing the triangle, square, and sawtooth shapes separately.

Subsequently, we investigated the architecture's efficacy in predicting frequencies unseen during training compared to how many frequency examples has been included in the training. We exploited the 'Synthetic' datasets because they have linear and finer frequency resolution. Therefore, we trained the model using all the 'Synthetic-XX' datasets featuring different numbers of frequency wave examples in the training set. The buffer size T, in this case, is set to 32 since the lower frequency included is 100 Hz, which is not as low as in the 'Analog' dataset. This allows a significant speed-up in training time.

Finally, we investigated how the model's performance adjusts when tasked with learning all waveshapes. In this scenario, the additional conditioning parameter c, defining the shape, is included in the model. The parameter ranges from 0 to 1, where 0 is associated with triangle, 0.5 sawtooth, and 1 square waveshape.

2.6. Training Strategy

The models are trained for a maximum of 1000 epochs and use the Adam optimizer. The training was stopped earlier in case of no reduction of validation loss for 50 epochs. In addition, we implemented a time-based schedule for the learning rate, where it decreases by 75% each epoch, starting from an initial learning rate of $3 \cdot 10^{-4}$. The loss function used is the normalized mean square error (NMSE), defined as

$$NMSE = \frac{\sum_{n=0}^{N}(y_n - \hat{y_n})^2}{\sum_{0}^{N} y_n^2} \tag{4}$$

with y the ground-truth samples, $\hat{y}$ the predicted samples, and N the size of the batch. Lastly, the teaching forcing was exploited, which involves feeding the ground-truth samples back into the model after each step, thus forcing the model to stay close to the ground-truth sequence.

3. RESULTS

Table 3 summarizes the accuracy of models using the variants of the Backbone layer when trained using triangle, square, and sawtooth waveshapes separately from the 'Analog-18' dataset. The performance metrics are reported separately for each of the six octaves included in the dataset. Each wave example included in the test set is generated with the buffer initialized using the ground truth samples. It is evident that models using RNN and LSTM as Backbone layers consistently deliver superior accuracy across all datasets. Specifically, the LSTM case excels with the triangle waveform case, whereas the RNN and LSTM variants perform similarly with square and sawtooth datasets. Overall, recurrent-based architectures outperform the temporal convolutional model, with the RNN and LSTM variants being the most accurate.

Figure 2 shows target versus predicted waveforms for two examples for each variant of the Backbone layer. It is evident that only the LSTM variant could generate the period of the waveshape for the lowest E_0, which has a frequency of 20 Hz and a period of 2400 samples long. Other variants are unable to reproduce such a low frequency with the considered settings. In addition, Figure 3 shows target versus predicted waveforms for two square wave examples for the LSTM variant. The plots demonstrate how the waveshape of the analog VCO undergoes significant changes at various frequencies and how the model effectively learns these variations. Table 4 presents the training results for the top-performing variants, RNN and LSTM, evaluated across different training data sizes: 'Analog-15', 'Analog-16', 'Analog-17', and 'Analog-18'. The results indicate that increasing the length of the audio files generally improves performance, achieving the lowest error rates on the 'Analog-18' dataset. Notably, while the RNN Backbone layer outperforms the LSTM in scenarios with smaller training data sizes, the LSTM models demonstrate superior performance as the training data size increases.

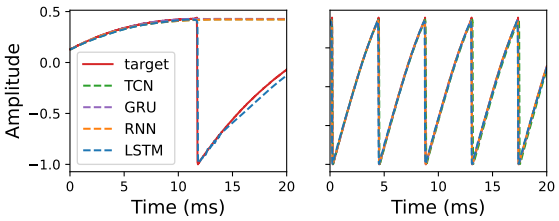

Figure 2: *Waveforms of target and prediction using the sawtooth wave shape included in the 'Analog-18' dataset for each Backbone layer variant. From left to right, the waveforms display the lower and higher frequencies included in the dataset:* 20 *Hz* (E_0) *and* 1318 *Hz* (E_6)*, respectively.*

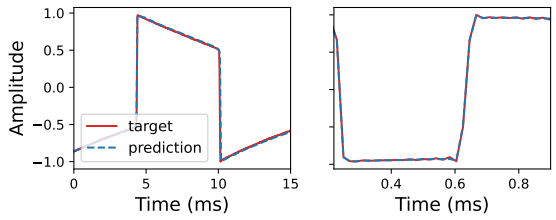

Figure 3: *Waveforms of target and prediction using the square wave shape included in the 'Analog-18' dataset for LSTM Backbone layer variant. From left to right, the waveforms represent* 41.20 *Hz* (E_1) *and* 659.25 *Hz* (E_5)*.*

The impact of the number of generated samples included in the input buffer T is shown in Table 5. As T increases, mod-

Table 3: *Performance metrics for each variant of the Backbone layer, reported separately for each of the six octaves included in the 'Analog-18' dataset, and related to models trained with triangle, square, sawtooth waveshapes separately. The lowest values for each performance metric are highlighted in bold.*

		TCN		RNN		GRU		LSTM	
Dataset	Oct	NMSE	FFT	NMSE	FFTE	NMSE	FFTE	NMSE	FFTE
'Analog-18' Triangle	$E_0 - D_1$	$3.66 \cdot 10^{-4}$	0.83	$6.15 \cdot 10^{-4}$	1.39	$1.91 \cdot 10^{-4}$	0.54	$\mathbf{1.89 \cdot 10^{-4}}$	**0.45**
	$E_1 - D_2$	$1.31 \cdot 10^{-6}$	0.13	$1.13 \cdot 10^{-6}$	0.09	$2.84 \cdot 10^{-7}$	0.05	$\mathbf{9.37 \cdot 10^{-8}}$	**0.05**
	$E_2 - D_3$	$1.72 \cdot 10^{-6}$	0.15	$1.10 \cdot 10^{-6}$	0.10	$1.23 \cdot 10^{-6}$	0.10	$\mathbf{4.85 \cdot 10^{-7}}$	**0.06**
	$E_3 - D_4$	$3.34 \cdot 10^{-6}$	0.17	$3.98 \cdot 10^{-6}$	0.20	$3.84 \cdot 10^{-6}$	0.20	$\mathbf{3.03 \cdot 10^{-6}}$	**0.16**
	$E_4 - D_5$	$\mathbf{7.46 \cdot 10^{-6}}$	**0.20**	$1.45 \cdot 10^{-5}$	0.28	$1.39 \cdot 10^{-5}$	0.25	$1.52 \cdot 10^{-5}$	0.26
	$E_5 - D_6$	$1.30 \cdot 10^{-4}$	0.93	$1.59 \cdot 10^{-4}$	1.03	$7.72 \cdot 10^{-5}$	0.71	$\mathbf{5.46 \cdot 10^{-5}}$	**0.48**
'Analog-18' Square	$E_0 - D_1$	$1.20 \cdot 10^{-4}$	0.85	$6.16 \cdot 10^{-5}$	0.74	$\mathbf{6.02 \cdot 10^{-5}}$	**0.76**	$6.75 \cdot 10^{-5}$	0.82
	$E_1 - D_2$	$8.22 \cdot 10^{-5}$	0.98	$8.82 \cdot 10^{-6}$	0.50	$1.16 \cdot 10^{-5}$	0.61	$\mathbf{6.29 \cdot 10^{-6}}$	**0.49**
	$E_2 - D_3$	$1.69 \cdot 10^{-4}$	1.23	$6.72 \cdot 10^{-5}$	0.85	$7.05 \cdot 10^{-5}$	0.92	$\mathbf{3.64 \cdot 10^{-5}}$	**0.72**
	$E_3 - D_4$	$3.27 \cdot 10^{-9}$	1.12	$\mathbf{1.52 \cdot 10^{-4}}$	**0.67**	$2.24 \cdot 10^{-4}$	0.76	$1.78 \cdot 10^{-4}$	0.74
	$E_4 - D_5$	$7.65 \cdot 10^{-9}$	0.01	$\mathbf{7.28 \cdot 10^{-9}}$	**0.01**	$9.76 \cdot 10^{-9}$	0.01	$1.02 \cdot 10^{-8}$	0.01
	$E_5 - D_6$	$5.84 \cdot 10^{-9}$	0.01	$\mathbf{3.26 \cdot 10^{-9}}$	**0.01**	$6.83 \cdot 10^{-9}$	0.01	$8.53 \cdot 10^{-9}$	0.01
'Analog-18' Sawtooth	$E_0 - D_1$	$1.54 \cdot 10^{-4}$	1.19	$1.60 \cdot 10^{-4}$	1.19	$1.06 \cdot 10^{-4}$	1.04	$\mathbf{9.97 \cdot 10^{-5}}$	**1.00**
	$E_1 - D_2$	$2.71 \cdot 10^{-5}$	0.74	$2.36 \cdot 10^{-5}$	0.74	$1.81 \cdot 10^{-5}$	0.68	$\mathbf{1.80 \cdot 10^{-5}}$	**0.53**
	$E_2 - D_3$	$7.03 \cdot 10^{-5}$	0.95	$\mathbf{4.19 \cdot 10^{-5}}$	**0.65**	$3.85 \cdot 10^{-5}$	0.64	$4.28 \cdot 10^{-5}$	0.74
	$E_3 - D_4$	$\mathbf{1.44 \cdot 10^{-5}}$	**0.22**	$3.39 \cdot 10^{-5}$	0.45	$4.76 \cdot 10^{-5}$	0.41	$4.25 \cdot 10^{-5}$	0.33
	$E_4 - D_5$	$4.12 \cdot 10^{-5}$	0.61	$1.49 \cdot 10^{-5}$	0.19	$4.26 \cdot 10^{-5}$	0.15	$\mathbf{1.36 \cdot 10^{-5}}$	**0.51**
	$E_5 - D_6$	$4.29 \cdot 10^{-8}$	0.01	$\mathbf{4.84 \cdot 10^{-9}}$	**0.01**	$1.85 \cdot 10^{-8}$	0.01	$1.86 \cdot 10^{-8}$	0.01

Table 4: *Performance metrics for RNN and LSTM-based models trained with the 'Analog-18' dataset utilizing examples with different lengths. The models are trained with triangle, square, and sawtooth waveshapes separately. The training data size resulting in the lowest metrics is highlighted in bold.*

Length	RNN		LSTM	
Samples	NMSE	FFTE	NMSE	FFTE
$29,491$	$2.45 \cdot 10^{-4}$	1.09	$9.33 \cdot 10^{-4}$	1.65
$58,982$	$2.23 \cdot 10^{-4}$	0.77	$1.42 \cdot 10^{-4}$	0.71
$117,965$	$2.20 \cdot 10^{-4}$	0.88	$1.32 \cdot 10^{-4}$	0.68
$235,929$	$\mathbf{4.82 \cdot 10^{-5}}$	**0.47**	$\mathbf{4.80 \cdot 10^{-5}}$	**0.46**

Table 5: *Performance metrics for the RNN and LSTM-based models trained with the 'Analog-18' dataset when varying the size of the buffer T. The models are trained with triangle, square, and sawtooth waveshapes separately. The T sizes resulting in the lowest metrics are highlighted in bold.*

		RNN		LSTM	
T	Dataset	NMSE	FFTE	NMSE	FFTE
64	'Analog'	$1.04 \cdot 10^{-4}$	0.62	$7,74 \cdot 10^{-5}$	0.38
96	-18	$9.82 \cdot 10^{-5}$	0.52	$7.07 \cdot 10^{-5}$	0.57
128	Triangle	$8.40 \cdot 10^{-5}$	0.56	$5.81 \cdot 10^{-5}$	0.41
256		$\mathbf{3.94 \cdot 10^{-5}}$	**0.22**	$\mathbf{3.29 \cdot 10^{-5}}$	**0.26**
64	'Analog'	$1.68 \cdot 10^{-4}$	0.83	$1.33 \cdot 10^{-4}$	0.77
96	-18	$4.82 \cdot 10^{-5}$	0.47	$4.80 \cdot 10^{-5}$	0.46
128	Square	$7.49 \cdot 10^{-5}$	0.86	$1.75 \cdot 10^{-4}$	0.75
256		$\mathbf{7.99 \cdot 10^{-5}}$	**0.62**	$\mathbf{2.37 \cdot 10^{-5}}$	**0.42**
64	'Analog'	$1.12 \cdot 10^{-4}$	0.82	$1.69 \cdot 10^{-4}$	1.01
96	-18	$8.07 \cdot 10^{-5}$	0.71	$5.29 \cdot 10^{-5}$	0.57
128	Sawtooth	$8.17 \cdot 10^{-5}$	0.70	$4.06 \cdot 10^{-5}$	0.47
256		$\mathbf{4.12 \cdot 10^{-5}}$	**0.49**	$\mathbf{3.35 \cdot 10^{-5}}$	**0.40**

els show improved accuracy as they can use a longer temporal context to predict the next sample. This is particularly beneficial for low frequencies, where a single period includes a larger number of samples. It is important to note that although the size of the Compression layer's input differs, the input to the recurrent layers (utilized for the Backbone layer) consistently remains a 4-dimensional vector. Figure 4 shows how increasing T enhances the model's ability to generate low frequencies with greater accuracy. In particular, only the model trained with $T = 256$ succeeds in generating a triangle and square waveshape at 20 Hz from the 'Analog-18' dataset. This suggests that for the triangle and square wave, it is sufficient to have T approximately 10% of the number of samples in the period of the lowest frequency to generate, which is 20 Hz in this case. This also highlights the higher complexity associated with triangle and square wave generation compared to sawtooth. Indeed, accurate sawtooth generation requires setting T to approximately 4% of the number of samples in the period of the lowest frequency.

Table 6 provides insights into the impact of varying model sizes on performance using the 'Analog-18' dataset. As expected, increasing the number of units leads to improvements, especially for the square and sawtooth datasets, which present more complex challenges. Interestingly, increasing T leads to greater benefits to model performance than increasing the number of units.

Table 7 displays the performance of the models predicting unseen frequencies, as the density frequency examples in the training sets vary. It is observed that the LSTM-based model performs particularly well with synthetic sine waves, and generally better than the RNN variants. 'Synthetic-2' dataset, including the smaller number of frequency examples in the training set and only 50% of frequency examples, poses the greatest challenges considering both variants and wave shapes. On the other hand, the models

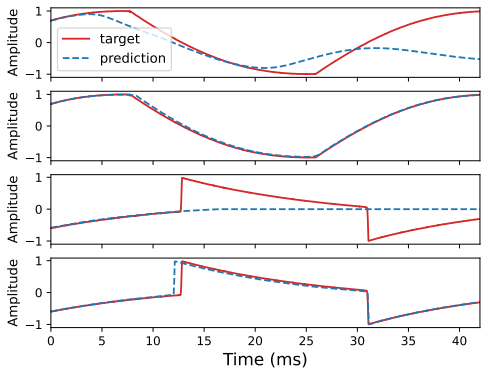

Figure 4: *Waveforms comparing target versus prediction for models trained with the 'Analog-18' dataset, related to triangle and square waveshapes under variations of buffer size T: triangle with T=128 (top), triangle with T=256 (mid-top), square with T=128 (mid-bottom) and square with T=256 (bottom). The waves refer to the lower pitch E_0, having 20 Hz, and are generated by a model using LSTM as the Backbone layer.*

Table 6: *Performance metrics of RNN and LSTM-based models trained with the 'Analog-18' dataset under variations of the number of units u. The models have been trained with triangle, square, and sawtooth waveshapes separately. The numbers of units resulting in the best metrics are highlighted in bold.*

		RNN		LSTM	
Dataset	u	NMSE	FFTE	NMSE	FFTE
'Analog'	16	$8.92 \cdot 10^{-5}$	0.48	$7.57 \cdot 10^{-5}$	0.48
-18	32	$8.45 \cdot 10^{-5}$	0.45	$7.87 \cdot 10^{-5}$	0.47
Triangle	**64**	$\mathbf{6.85 \cdot 10^{-5}}$	**0.40**	$\mathbf{6.15 \cdot 10^{-5}}$	**0.32**
'Analog'	16	$2.16 \cdot 10^{-4}$	0.91	$1.74 \cdot 10^{-4}$	0.81
-18	32	$7.37 \cdot 10^{-5}$	0.57	$1.55 \cdot 10^{-4}$	0.74
Square	**64**	$\mathbf{5.63 \cdot 10^{-5}}$	**0.50**	$\mathbf{4.76 \cdot 10^{-5}}$	**0.45**
'Analog'	16	$8.74 \cdot 10^{-5}$	0.70	$5.50 \cdot 10^{-5}$	0.55
-18	32	$4.75 \cdot 10^{-5}$	0.47	$4.60 \cdot 10^{-5}$	0.46
Sawtooth	**64**	$\mathbf{4.39 \cdot 10^{-5}}$	**0.59**	$\mathbf{4.38 \cdot 10^{-5}}$	**0.55**

present comparable results to the other 'Synthetic' datasets.

Figure 5 illustrates the results for two frequency cases included in the sines and sawtooth test sets, generated by the models trained with the 'Synthetic-32' dataset. The results indicate that the sine waves are accurately produced in both cases, showing strong alignment with the target. This suggests that the models can learn in scenarios where there is a step size of 32 Hz between frequency examples. However, the sawtooth wave poses a more challenging case, as we generally observe that the predicted signal tends to misalign from the target after a few cycles. This may occur because the target frequency is not accurately synthesized, leading to cumulative misalignments during generation.

Figure 6 shows an example of a 500 ms square-wave sweep from the 'Analog-18' dataset, generated by a LSTM-based model with $T = 96$, and 64 units. The sweep begins at 20 Hz and progresses to 500 Hz, with the frequency linearly increasing at each iteration. The actual dataset waveforms of 20 Hz and 500 Hz are also presented as a reference. Notably, this continuous frequency variation does not introduce artifacts, as demonstrated by both the

Table 7: *Performance metrics of RNN and LSTM-based models trained with the 'Synthetic-XX' (Synth-XX) sine and sawtooth datasets. The lowest metrics are highlighted in bold.*

	RNN		LSTM	
Dataset	NMSE	FFTE	NMSE	FFTE
Synth-2 Sine	$\mathbf{4.82 \cdot 10^{-5}}$	**0.41**	$8.57 \cdot 10^{-6}$	0.26
Synth-4 Sine	$5.05 \cdot 10^{-5}$	0.63	$6.47 \cdot 10^{-6}$	0.23
Synth-8 Sine	$4.85 \cdot 10^{-5}$	1.46	$\mathbf{5.35 \cdot 10^{-6}}$	**0.21**
Synth-16 Sine	$5.77 \cdot 10^{-5}$	1.83	$6.24 \cdot 10^{-6}$	0.22
Synt-32 Sine	$4.78 \cdot 10^{-4}$	1.40	$5.61 \cdot 10^{-6}$	0.22
Synth-2 Saw	$\mathbf{4.48 \cdot 10^{-5}}$	**1.78**	$3.83 \cdot 10^{-4}$	1.75
Synth-4 Saw	$4.12 \cdot 10^{-4}$	1.76	$\mathbf{2.11 \cdot 10^{-4}}$	**1.36**
Synth-8 Saw	$4.25 \cdot 10^{-4}$	1.80	$2.60 \cdot 10^{-4}$	1.56
Synth-16 Saw	$4.40 \cdot 10^{-4}$	1.77	$3.43 \cdot 10^{-4}$	1.64
Synth-32 Saw	$4.24 \cdot 10^{-4}$	1.80	$2.14 \cdot 10^{-4}$	1.46

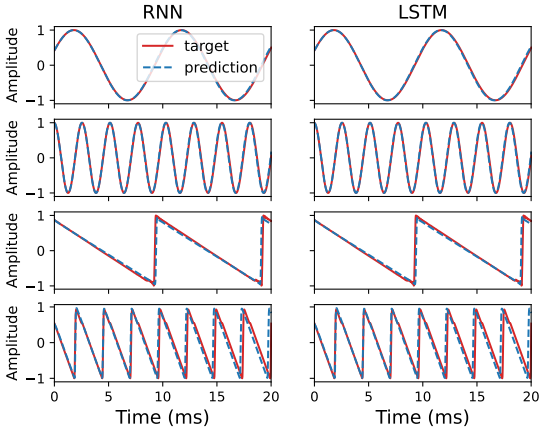

Figure 5: *Waveforms of target and prediction of the 'Synthetic-2' sines and sawtooth datasets for unseen frequencies. The plot refers to the model using RNN (left) and LSTM (right) as the Backbone layer trained with 50% of the dataset examples. The shown frequencies are 101 Hz and 299 Hz.*

visual representation in the figure and the audio examples provided in the online companion page [3].

The proposed model learns the target waveshape and replicates any aliasing present in the recordings. Therefore, the quality of the analog antialiasing filter used during the recording of the analog VCO and the digital antialiasing filter applied during data downsampling significantly affect the model's aliasing performance. Frequency domain plots are available on the companion online page, demonstrating that the model accurately reproduces the frequency content of the target recordings without introducing additional aliasing.

Lastly, Table 8 shows the outcomes of training the LSTM variant model, set up with 64 units and $T = 256$, with all waveshapes in the analog dataset. The model uses the input conditioning parameter c to specify the waveshape to be generated, which also enables it to morph between them. In this case, errors are slightly higher due to the higher task's complexity, but they are still close to those from models trained with individual waveshapes. Figure 7 shows generated examples of triangle, square, and sawtooth wave-

[3] https://riccardovib.github.io/NeuralOSC_pages/

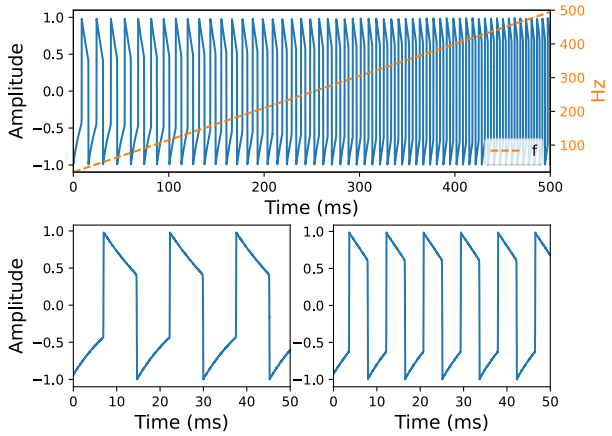

Figure 6: *Waveforms of a generated* 500 *ms square-wave sweep from the 'Analog-18' dataset. The sweep begins at* 20 *Hz and progresses to* 500 *Hz, with the frequency linearly increasing at each iteration. Waveforms of the dataset recordings at* 20 *Hz and* 500 *Hz are shown at the bottom for reference.*

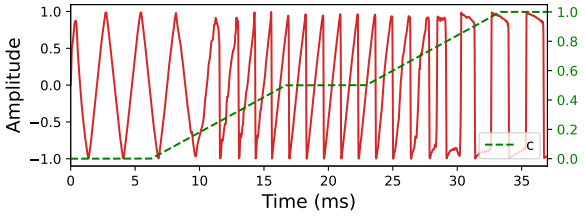

Figure 7: *Waveforms of* 37 *m of generated signal morphing between triangle, sawtooth, and square waveshape. The* c *input parameter is varied linearly to condition the generation across waveshapes. The waveform is generated using the LSTM variant with* 64 *units and* $T = 256$*, which is trained with the 'Analog-18' dataset. The signal has a frequency of* 164.81 *Hz* (E_3)*.*

shapes from the 'Analog-18', demonstrating the model's ability to adapt to different waveshape characteristics. The example is limited to 37 ms to better distinguish the waveshapes.

Table 8: *Performance metrics for the LSTM variant model trained on the full 'Analog-18' dataset. The model has* 64 *units and* 256 *as* T*. The losses are reported separately on all the wave shapes.*

Dataset	Waveshape (c)	NMSE	FFT
'Analog-18'	Triangle (0)	$4.73 \cdot 10^{-4}$	1.36
-	Sawtooth (0.5)	$3.69 \cdot 10^{-4}$	1.33
-	Square (1)	$2.97 \cdot 10^{-4}$	0.86

3.1. Ablation Experiments

The Compression layer was introduced to decrease the number of recurrent steps, thereby reducing the overall operations required to generate each audio sample. Computation costs are approximately reduced by a factor of T/U_c, where T represents the number of audio samples in the input buffer, used as a shift register, and U_c denotes the number of units in the compression layer. As summarized in Table 9, the addition of the Compression layer also improves modeling accuracy, indicated by a slight reduction in NMSE loss and FFT metrics. However, this enhancement is not consistently seen with RNN-based models. It's important to note that while the Compression layer reduces inference operations, it also increases the number of trainable parameters in the model.

4. CONCLUSIONS

This paper investigated the use of artificial neural networks (ANNs) for emulating Voltage-Controlled Oscillator (VCO) components in analog synthesizers. We gathered data from both real and synthetic VCOs to train models capable of generating waveform shapes at any frequency. Our findings suggest that ANNs have the potential to learn waveshape variations across frequencies, thereby enhancing emulation accuracy beyond traditional methods. Utilizing black-box autoregressive architectures, we successfully modeled various waveform types, including triangle, square, and sawtooth waves. The models functioned with a shift buffer, taking wave frequency and shape as inputs, demonstrating a promising approach for advanced sound synthesis.

We evaluated various architectural configurations in different modeling scenarios, varying on dataset size, input buffer size, number of computational units, and density of training frequency examples. Recurrent variants, especially LSTM, showed superior performance. Extended durations of individual waveform examples proved beneficial during training, as did larger buffer sizes, which enhance the network's ability to emulate lower frequencies. Overall, the models effectively emulate basic waveshapes, although they encountered significantly greater difficulty with lower frequency ranges. In addition, increasing the number of units can further enhance performance, albeit at the cost of higher computational demands. LSTM variants also demonstrated efficacy in predicting unseen frequencies, although the target frequency is not always perfectly matched, leading, at times, to cumulative misalignments over time. Ultimately, a single model, using the best configuration identified in previous experiments, successfully emulated all waveshapes by employing an additional input parameter to morph between them.

While the architecture allows real-time synthesis without latency, it demands a considerable number of floating-point operations (FLOPs). The LSTM variant with 64 units and a buffer size of 96 requires $141,312$ FLOPs per sample. Smaller networks show errors similar to larger ones, suggesting a smaller model might suffice. Specifically, the LSTM variant with 32 units requires $34,816$ FLOPs, and with 16 units, it needs $9,216$ FLOPs. Additionally, wave generation needs the buffer initialized with samples from real recordings to provide context, although this is only a fraction of the complete wave cycle.

Preliminary experiments with fully stateful architectures, previously used for emulating analog audio effects [14], yielded unsatisfactory results compared to the proposed models. Additionally, deep state-space models [15], when used in the backbone layer, are constrained to track short temporal dependencies, regardless of compression layer usage. Therefore, these models are less suited to the current architecture, as they perform best in scenarios with particularly long temporal dependencies. In future research, we aim to explore how SSM-based models can further enhance analog VCO modeling and investigate architectural variations that may lead to reduced computational complexity.

Table 9: *Performance metric variations with the introduction of the Compression Layer for models using recurrent-based architectures in the Backbone layer. Negative numbers indicate a reduction in the associated loss, which signifies increased accuracy.*

	RNN		GRU		LSTM	
Dataset	NMSE	FFT	NMSE	FFT	NMSE	FFT
'Triangle Analog-18'	$-4.57 \cdot 10^{-5}$	-0.03	$-5.28 \cdot 10^{-5}$	-0.10	$-2.37 \cdot 10^{-5}$	-0.03
'Sawtooth Analog-18'	$3.66 \cdot 10^{-4}$	0.61	$-2.37 \cdot 10^{-5}$	-0.06	$-6.16 \cdot 10^{-5}$	-0.21
'Square Analog-18'	$2.29 \cdot 10^{-4}$	0.54	$-7.79 \cdot 10^{-6}$	-0.09	$-1.14 \cdot 10^{-5}$	-0.01

5. REFERENCES

[1] Phil Burk, K Greenebaum, and R Barzel, "Band limited oscillators using wave table synthesis," *Audio Anecdotes II—Tools, Tips, and Techniques for Digital Audio*, pp. 37–53, 2004.

[2] James A Moorer, "The synthesis of complex audio spectra by means of discrete summation formulas," *Journal of the Audio Engineering Society*, vol. 24, no. 9, pp. 717–727, 1976.

[3] Glen Deslauriers and Colby Leider, "A bandlimited oscillator by frequency-domain synthesis for virtual analog applications," in *Audio Engineering Society Convention 127*. Audio Engineering Society, 2009.

[4] Timothy Stilson and Julius O Smith III, "Alias-free digital synthesis of classic analog waveforms," in *ICMC*, 1996.

[5] Vesa Valimaki and Antti Huovilainen, "Antialiasing oscillators in subtractive synthesis," *IEEE Signal Processing Magazine*, vol. 24, no. 2, pp. 116–125, 2007.

[6] Vesa Valimaki, Juhan Nam, Julius O Smith, and Jonathan S Abel, "Alias-suppressed oscillators based on differentiated polynomial waveforms," *IEEE Transactions on audio, speech, and language processing*, vol. 18, no. 4, pp. 786–798, 2009.

[7] Dániel Ambrits and Balázs Bank, "Improved polynomial transition regions algorithm for alias-suppressed signal synthesis," in *Proc. of the Int. Conf. on Sound and Music Computing (SMC)*, 2013, pp. 561–568.

[8] G. Kreković, "Deep convolutional oscillator: Synthesizing waveforms from timbral descriptors," in *Proc. of the Int. Conf. on Sound and Music Computing (SMC)*, 2022, pp. 200–206.

[9] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[10] Soroush Mehri, Kundan Kumar, Ishaan Gulrajani, Rithesh Kumar, Shubham Jain, Jose Sotelo, Aaron Courville, and Yoshua Bengio, "SampleRNN: An unconditional end-to-end neural audio generation model," *Proc. of Int. Conf. on Learning Representations (ICLR)*, 2017.

[11] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville, "Film: Visual reasoning with a general conditioning layer," in *Proceedings of the AAAI conference on artificial intelligence*, Feb 2018, vol. 32.

[12] Y.N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language modeling with gated convolutional networks," in *International conference on machine learning*. PMLR, Dec 2017, pp. 933–941.

[13] Riccardo Simionato and Stefano Fasciani, "Conditioning methods for neural audio effects," in *Proc. of the Int. Conf. on Sound and Music Computing (SMC)*, 2024.

[14] Riccardo Simionato and Stefano Fasciani, "Modeling time-variant responses of optical compressors with selective state space models," *Journal of Audio Engineering Society*, vol. 73, no. 3, pp. 144–165, 2025.

[15] Albert Gu, Karan Goel, and Christopher Ré, "Efficiently modeling long sequences with structured state spaces," in *Proc. of Int. Conf. on Learning Representations (ICLR)*, 2022.

[16] 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.

[17] Alfred Fettweis, "Wave digital filters: Theory and practice," *Proceedings of the IEEE*, vol. 74, no. 2, pp. 270–327, 1986.

[18] Alberto Bernardini, Riccardo Giampiccolo, Enrico Bozzo, and Federico Fontana, "Wave digital extended fixed-point solvers for circuits with multiple one-port nonlinearities," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2024.

[19] Tara Vanhatalo, Pierrick Legrand, Myriam Desainte-Catherine, Pierre Hanna, Antoine Brusco, Guillaume Pille, and Yann Bayle, "A review of neural network-based emulation of guitar amplifiers," *Applied Sciences*, vol. 12, no. 12, pp. 5894, 2022.

[20] Etienne Gerat, Purbaditya Bhattacharya, and Udo Zölzer, "A differentiable digital moog filter for machine learning applications," in *Proc. of the Int. Conf. on Digital Audio Effects (DAFx)*, 2023.

[21] Julian Parker and Stephano D'Angelo, "A digital model of the buchla lowpass-gate," in *Proc. Int. Conf. Digital Audio Effects (DAFx), Maynooth, Ireland*, 2013, pp. 278–285.

[22] Riccardo Simionato and Stefano Fasciani, "Hybrid neural audio effects," in *Proc. of the Int. Conf. on Sound and Music Computing (SMC)*, 2024.

[23] Fabián Esqueda, Henri Pöntynen, Vesa Välimäki, Julian D Parker, et al., "Virtual analog buchla 259 wavefolder," in *Proc. of the Int. Conf. on Digital Audio Effects (DAFx), Edinburgh, UK*, 2017, pp. 5–9.

[24] Fabián Esqueda, Henri Pöntynen, Julian D Parker, and Stefan Bilbao, "Virtual analog models of the lockhart and serge wavefolders," *Applied Sciences*, vol. 7, no. 12, pp. 1328, 2017.