NEURAL MODELLING OF PERIODICALLY MODULATED TIME-VARYING EFFECTS

Alec Wright and Vesa Välimäki [*]

Acoustics Lab, Dept. of Signal Processing and Acoustics
Aalto University
Espoo, Finland
alec.wright@aalto.fi

ABSTRACT

This paper proposes a grey-box neural network based approach to modelling LFO modulated time-varying effects. The neural network model receives both the unprocessed audio, as well as the LFO signal, as input. This allows complete control over the model's LFO frequency and shape. The neural networks are trained using guitar audio, which has to be processed by the target effect and also annotated with the predicted LFO signal before training. A measurement signal based on regularly spaced chirps was used to accurately predict the LFO signal. The model architecture has been previously shown to be capable of running in real-time on a modern desktop computer, whilst using relatively little processing power. We validate our approach creating models of both a phaser and a flanger effects pedal, and theoretically it can be applied to any LFO modulated time-varying effect. In the best case, an error-to-signal ratio of 1.3% is achieved when modelling a flanger pedal, and previous work has shown that this corresponds to the model being nearly indistinguishable from the target device.

1. INTRODUCTION

Virtual analog modelling is a popular research topic, which seeks to develop algorithms that can accurately imitate music hardware, such as guitar effects pedals, amplifiers or musical instruments [1, 2, 3, 4]. Time-varying audio effects are a family of audio effects, in which the behaviour of the system changes over time. This category includes phasing, chorus, flanging, Leslie speakers and more. This paper focuses on the modelling of such effects processing units using a neural network.

Virtual analog modelling research can broadly be divided into three approaches, "white-box" [5, 2, 6], "grey-box" [7, 8], and "black-box" [9, 10] modelling. In "white-box" modelling, circuit analysis or physical properties of the system are used to derive equations describing its behaviour. In "black-box" modelling, the relationship between inputs and outputs of the system are directly measured and subsequently emulated. Finally, "grey-box" models fall somewhere between the previous two approaches, requiring knowledge of how the system works, but also using data measured from it. This paper explores a generalised gray-box modelling approach for time-varying effects using a recurrent neural network (RNN).

[*] This research is part of the "Nordic Sound and Music Computing Network—NordicSMC," NordForsk project no. 86892.

In recent years, numerous studies on virtual analog modelling of guitar amplifiers [11, 12, 13, 14, 4, 15] and other nonlinear systems [16, 17] using neural networks have been published. Neural network modelling of time-varying audio effects has received less attention, with the first publications being published over the past year [18, 3]. Whilst Martínez *et al.* report that accurate emulations of several time-varying effects were achieved, the model utilises bi-directional Long Short Term Memory (LSTM) and is therefore non-causal and unsuitable for real-time applications.

In this paper we present a general approach for real-time modelling of audio effects with parameters that are modulated by a Low Frequency Oscillator (LFO) signal. Our approach is an adaptation of a technique used for modelling guitar amplifiers that we proposed last year [14]. We validate our approach by creating models of a digital phasing algorithm and an analog phaser and flanger pedal. The resulting models require relatively little processing power to run and are suitable for real-time operation.

The rest of this paper is organised as follows. Section 2 provides some background on time-varying audio effects. Section 3 describes the neural network architecture and training process used during this study. Section 4 describes preliminary experiments carried out to validate and test the limitations of our proposed approach. Section 5 details the method we applied to predict the LFO signal in audio effect pedals. Sections 6 and 7 describe the experiments and results, respectively, achieved when applying our model to phaser and flanger guitar effects pedals, and Section 8 concludes the paper.

2. TIME-VARYING AUDIO EFFECTS

This paper focuses on modelling time-varying audio effects, a family of audio effects where the parameters of the effect are varied, or *modulated*, over time.

A simple example of a time-varying audio effect is the *wah-wah* filter. This is just a bandpass filter with a variable resonant frequency [1]. The resonant frequency is usually modulated by a footpedal; however, in the *Auto-Wah* variant of the effect, the resonant frequency can be modulated either by the envelope of the input signal, or a Low-Frequency Oscillator (LFO). Figure 1 shows an example spectrogram of an Auto-Wah effect frequency response, along with the corresponding LFO signal modulating its resonant frequency. An LFO is a general term for a periodic signal, usually at a frequency below 20 Hz, that modulates the parameters of an audio effect or instrument.

In this paper, we propose a general approach for modelling audio effects with parameters that are modulated by an LFO signal. To demonstrate the viability of the method, we create models of two popular LFO modulated time-varying audio effects, *phasing* and *flanging*. As these effects share some similarities, the fol-

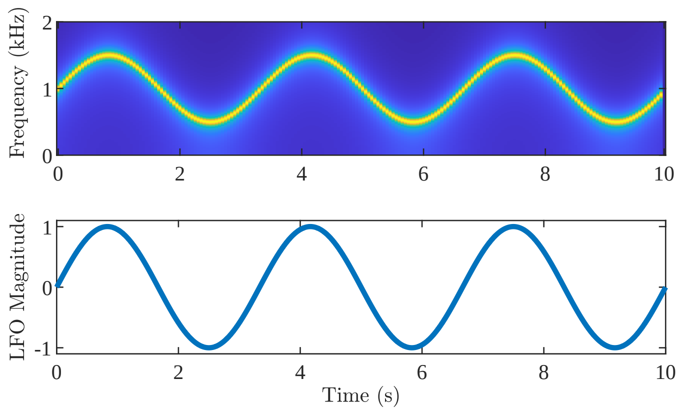

Figure 1: *Spectrogram of an Auto-Wah effect response (top) and corresponding 0.3 Hz sinusoidal LFO signal (bottom).*

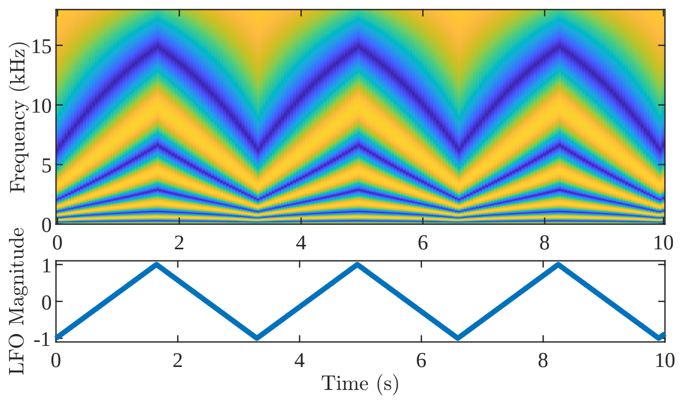

Figure 2: *Spectrogram of a phaser response (top) and corresponding triangular LFO signal (bottom).*

lowing sections will briefly describe both of the effects and how they are achieved, whilst highlighting the key differences between them.

2.1. Phasing

Both phasers and flangers work by introducing moving notches into the spectrum of the input signal [19]. In the case of phasing, the notches can be introduced using notch filters, or a series of low-order allpass filters [20]. In the case of the latter, the effect is achieved by mixing the unprocessed signal with the allpass chain filtered signal. The allpass filters introduce phase delay to the signal, and any frequencies where the phase delay is equal to an odd multiple of π, will be cancelled out when the dry and all-passed signals are mixed together. Modulation of the notch frequencies is achieved by varying the allpass coefficients using an LFO signal. The number of notches is determined by the maximum phase delay introduced to the signal, which is in turn determined by the number and order of the allpass filters. Figure 2 shows an example spectrogram of a phaser frequency response, along with the corresponding LFO signal modulating the allpass filter coefficients.

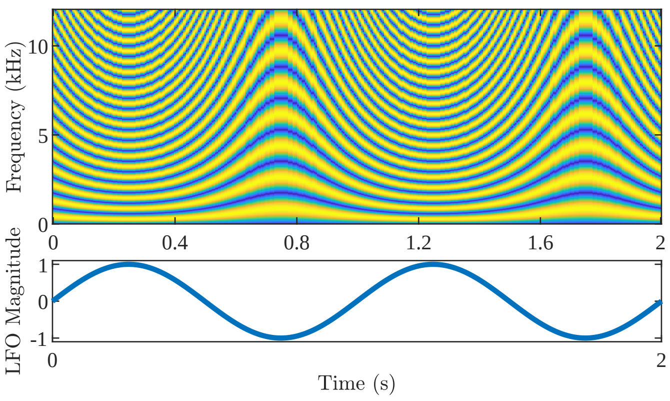

Figure 3: *Spectrogram of a flanger effect (top) and corresponding LFO signal (bottom).*

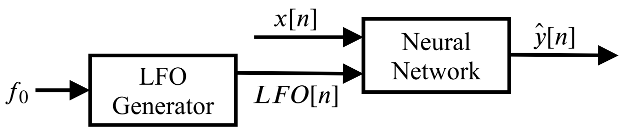

Figure 4: *Model structure, where $x[n]$ is the input signal, $LFO[n]$ is the LFO signal, f_0 is the desired frequency of the LFO and $\hat{y}[n]$ is the network's predicted output sample.*

2.2. Flanging

Flanging is achieved by mixing a signal with a delayed version of itself [19]. This is equivalent to applying an inverse comb filter to the signal. The effect is modulated by varying the length of delay. As the notches are achieved by an inverse comb filter, the notch locations are harmonically related, and are integer multiples of the lowest frequency notch. As the lowest frequency notch increases, the total number of notches decreases, as more of the higher frequency notches start to exceed the Nyquist frequency. Figure 3 shows an example spectrogram of a flanger response, along with the corresponding LFO signal that is modulating its delay line length.

The main difference between phaser and flanger can be seen in their respective spectrograms: compared to the phaser, the flanger introduces many more notches to the spectrum, and does not allow any control over the locations of each notch.

3. MODEL ARCHITECTURE AND TRAINING

The general idea proposed in this study is to model LFO modulated time-varying audio effects using a neural network. The proposed model processes the input audio on a sample-by-sample basis, with the time-varying aspects of the effect modelled using an LFO signal as an additional input to the model. The general form of the trained model is depicted in Fig. 4.

This approach to modelling time-varying effects should result in a smaller and less complex neural network model being required, as a key feature (the LFO) of the audio effect is provided to the network. Additionally the shape and frequency of the LFO can be controlled freely after the model is trained. One issue with this approach is that training the neural network requires

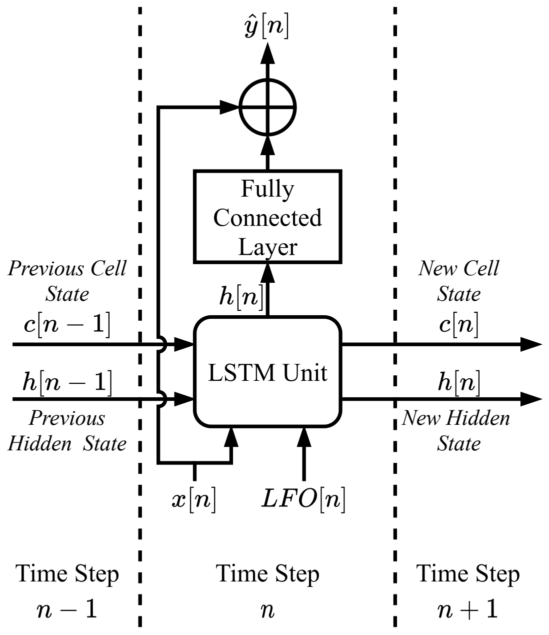

Figure 5: *RNN structure, where $x[n]$ is the input audio signal, $LFO[n]$ is the input LFO signal, h and c are the LSTM Hidden and Cell states, respectively, and $\hat{y}[n]$ is the network's predicted output sample.*

input/output data from the target device, annotated with the LFO signal. Predicting the LFO signal accurately is not trivial and this is discussed further in Section 5.

3.1. Neural Network Model

Validation of this approach was carried out using a Recurrent Neural Network (RNN) model similar to that which was used in our previous work [14]. The model consists of an LSTM unit, followed by a fully connected layer, and is depicted in Fig. 5.

An LSTM unit has two $State$ vectors, the $Hidden\ State$, h, and the $Cell\ State$, c, both of which are updated at each time step, with the $Hidden\ State$ also being used as the LSTM output. The LSTM states are updated according to the following equations:

$$i[n] = \sigma(W_{ii}x[n] + b_{ii} + W_{hi}h[n-1] + b_{hi}), \tag{1}$$

$$f[n] = \sigma(W_{if}x[n] + b_{if} + W_{hf}h[n-1] + b_{hf}), \tag{2}$$

$$\tilde{c}[n] = \tanh(W_{ic}x[n] + b_{ic} + W_{hc}h[n-1] + b_{hc}), \tag{3}$$

$$o[n] = \sigma(W_{io}x[n] + b_{io} + W_{ho}h[n-1] + b_{ho}), \tag{4}$$

$$c[n] = f[n]c[n-1] + i[n]\tilde{c}[n], \tag{5}$$

$$h[n] = o[n]\tanh(c[n]), \tag{6}$$

where $i[n]$ is the input gate, $f[n]$ is the forget gate, $\tilde{c}[n]$ is the candidate cell state, $o[n]$ is the output gate, tanh(.) is the hyperbolic tangent function and $\sigma(.)$ is the logistic sigmoid function. $x[n]$ is the LSTM input signal, consisting of the unprocessed guitar signal and the value of the LFO at time n. The LSTM learns the weight matrices and bias vectors, denoted by W and b in the above equations, during training. The LSTM $hidden\ size$ is a user-defined parameter which determines the size of the state vectors, and thus the required size of the weight matrices and bias vectors.

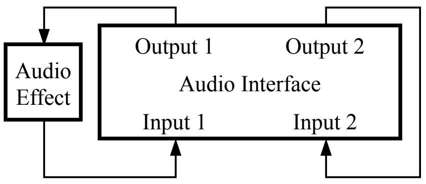

Figure 6: *Measurement setup, where Input 1 records the target signal, $y[n]$, and Input 2 records the corresponding input signal, $x[n]$.*

The fully connected layer performs an affine transformation of the hidden state vector, which is summed with the network input to give the network's predicted output sample:

$$\hat{y}[n] = W_{fc}h[n] + b_{fc} + x[n], \tag{7}$$

where W_{fc} and b_{fc} are the fully connected layer's weight matrix and bias vector respectively and $h[n]$ and $x[n]$ are the LSTM hidden state and network input at time n, respectively.

The model was implemented using PyTorch [21]. Whilst this study only used the RNN model, the method of including the LFO signal as a neural network input should work for other neural network structures, such as the WaveNet-style model proposed for guitar amplifier modelling in [13].

Whilst the required processing power for running the neural network model was not measured in this study, the model is very similar to a previously tested model [14], which, when run on an Apple iMac with 2.8 GHz Intel Core i5 processor, could process one second of audio in 0.12 s when an LSTM hidden size of 32 was used, or 0.24 s for a hidden size of 64.

3.2. Training Data

The datasets for each of the effects modelled in this study were created by processing clean guitar audio with the target effect, and recording the output. The training data used as input during this study was taken from the *IDMT-SMT-Guitar* database [22] for guitar transcription. This is a collection of clean guitar recording with no audio effects applied. To best capture the time-varying nature of the effect, audio clips containing minimal silence between notes were selected from the dataset. All audio was recorded at a sample rate of 44.1 kHz.

When a physical device was being modelled, a PC connected to a Focusrite Scarlett 2i2 digital audio interface was used to output and record the signals. The first output of the audio interface was connected to the input of target device, with the output of the target device then being connected to the first input of the audio interface. Additionally, to compensate for the latency introduced by the audio interface, the second output of the audio interface was connected directly to the second input of the audio interface. The unprocessed guitar audio was then sent through both outputs. The signals recorded from the first and second inputs make up the time-synchronised target, $y[n]$, and input, $x[n]$, data, respectively. The measurement setup is shown in Fig. 6

3.3. Training

The training process used was similar to that described in [14]. The Training Dataset was split into 1-second segments, and processed in mini-batches. During the processing of each mini-batch, 1000 samples were initially processed to allow the RNN states to initialise, then parameter updates were carried out every 1024 samples. It is noted that the length of the segments, as well as the time between parameter updates, is short in comparison to the LFO frequency. As such, at each parameter update, the model is learning from just a small segment of the LFO signal. However, as the full range of the LFO is included in the dataset, the model still learns to imitate the effect over its full range of LFO values.

The loss function used was the error-to-signal ratio (ESR) with first-order highpass pre-emphasis filtering, summed with a DC loss term:

$$\mathcal{E}_{\text{ESR}} = \frac{\sum_{n=0}^{N-1} |y_p[n] - \hat{y}_p[n]|^2}{\sum_{n=0}^{N-1} |y_p[n]|^2}, \tag{8}$$

$$\mathcal{E}_{\text{DC}} = \frac{|\frac{1}{N}\sum_{n=0}^{N-1}(y[n] - \hat{y}[n])|^2}{\frac{1}{N}\sum_{n=0}^{N-1} |y[n]|^2}, \tag{9}$$

$$\mathcal{E} = \mathcal{E}_{\text{ESR}} + \mathcal{E}_{\text{DC}}, \tag{10}$$

where y_p is the pre-emphasised target signal, and $\hat{y}_p$ is the pre-emphasised neural network output. The first order pre-emphasis filter coefficient used was 0.85. The ESR loss is the squared error divided by the energy of the target signal, which effectively normalises the loss so it is not dominated by higher energy parts of the signal. The highpass pre-emphasis filter helps the model to more effectively learn the high-frequency behaviour of the target device. The training was carried out on a NVIDIA V100 Graphics Processing Unit (GPU) using the Adam optimizer [23]. The segments were shuffled at the end of each epoch. Training typically took about 6 hours to complete.

4. TOY-PROBLEM: MODELLING A DIGITAL MODEL

To provide initial validation of our approach, a digital model was used as a target device. Whilst it might seem slightly convoluted to train a neural network to model a digital model of an audio effect, this was deemed necessary as it allowed us to provide a completely accurate LFO signal during neural network training. Otherwise, when modelling an analog device, it would be difficult to determine if any failure of the neural network to emulate the target device is due to the model itself, or just inaccurate prediction of the LFO signal.

The gray-box model of the *Fame Sweet Tone Phaser PH-10*, proposed in [7], was used as the target device. It is implemented as a series of ten first-order allpass filters, with the first two and last two of them having a constant coefficient, and the remaining six filter's coefficients being modulated by an LFO.

The complete dataset used for modelling the digital phaser consists of a five-minute training dataset, a one-minute validation dataset, and a 50-second test dataset. Networks with LSTM hidden sizes of 8, 16, and 32 were trialled. To determine how much training data is needed to create an accurate model, neural networks were trained with varying amounts of the training dataset being excluded. The different training dataset lengths used, range from 10 seconds to five minutes in length. This resulted in a total of 30 networks being created. Identical validation and test datasets were used for all networks.

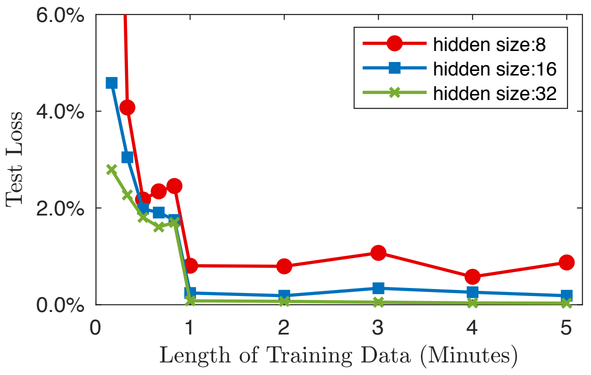

Figure 7: *Test loss with varying hidden size and training data length in the toy-problem of digital phaser modelling.*

For the training and validation datasets, the phaser LFO used was a rectified sine-wave with frequency of 0.34 Hz. For the test dataset, four separate test datasets were created, each with the same input audio but with the LFO frequency being varied. LFO frequencies of 0.1 Hz, 0.34 Hz, 1.1 Hz, and 3.8 Hz were used for creation of the test datasets.

The ESR loss achieved on the test dataset for the various training data lengths and network hidden sizes is shown in Fig. 7. It can be seen that one minute of training data is sufficient for the neural networks to model this system, with the test loss being relatively unaffected by any further increase in the training data length. It was also observed that model performed equally well on the test datasets that were generated with LFO frequencies that were unseen during neural network training. These initial tests indicate that the neural network is capable of modelling time-varying effects, when the true LFO signal is known, and furthermore can generalise the effects behaviour when given an unseen LFO frequency as input.

5. LOW FREQUENCY OSCILLATOR MEASUREMENT

To further validate the model, it is desirable to model analog devices instead of digital systems. This presents an extra challenge as the input data for the neural network requires an accurate prediction of the LFO signal that is modulating the parameters. Furthermore, there is no clear way of determining the accuracy of the LFO signal prediction. The approach to measuring the LFO signal is described in this section.

5.1. Measurement Signal

The LFO measurement signal used in this study is one previously proposed for phaser pedal LFO measurement [7]. As the system behaviour varies over time, it is necessary to use a relatively short test signal, that can still provide a fairly accurate picture of the system's frequency response at the time of measurement. It has previously been suggested that the impulse response of a cascade of first-order allpass filters can be used as a chirp signal [24], with the number of filters in the cascade determining the total length of the signal.

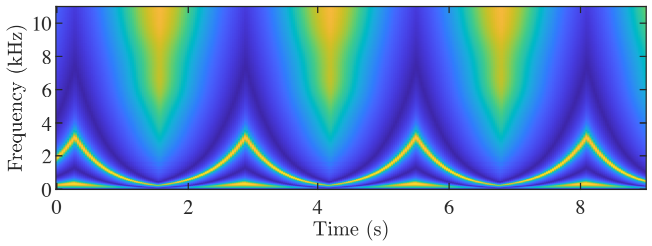

Figure 8: *Spectrogram of a phaser pedal measured using a chirp train.*

The full test signal used for this work is simply an impulse train, with a spacing of 30 ms, filtered by a cascade of 64 identical first-order allpass filters. An allpass filter coefficient of -0.9 is used, resulting each chirp being approximately 1200 samples, or 2.7 ms, long. The resulting chirp train is then input to target device, and the output of the device is recorded. As the length and location of the input chirps is known, the frequency response of the device at the time of each chirp can be found by transforming the time-domain chirps to the frequency domain.

Figure 8 shows an example spectrogram from a phaser pedal, measured using this method. Whilst the LFO can be clearly observed in the spectrogram, to obtain a prediction of the LFO that is sufficiently accurate for extrapolation to the rest of the dataset, further steps must be taken.

5.2. LFO Prediction

The measurement of the frequency response in this way can be viewed as sampling the frequency response at a sampling period equal to the spacing between the test chirps. The shape of the LFO can be inferred by observing the movement of the notches in the spectrum. This can be achieved by inverting the magnitude spectrum, so the notches instead appear as peaks, and using a peak picking algorithm.

For this work, we used the Matlab built-in `findpeaks` function. As the effects generally have multiple notches, this will result in a number of notches being located. Each notch frequency is assumed to change by a relatively small amount between each sample of the frequency spectrum, so the two closest notches between subsequent frames are assumed to belong to the same notch. The trajectory of each notch is tracked in this way over the duration of the test signal.

As LFOs are usually a simple waveform such as a rectified sine or triangle wave, the shape can be determined visually. This can be seen in Fig. 8, where the LFO appears to be a rectified sine wave. To predict the frequency and phase offset of the LFO signal, we used the Matlab `lsqnonlin` function, a nonlinear least-squares solver. For example, where the LFO is a rectified sine-wave, the solver would be given following equation to minimise:

$$f(f_0, \phi) = |\sin[2\pi t(f_0/2) + (\phi/2)]| - s_{LFO}(t), \qquad (11)$$

where f_0 and ϕ are the predicted LFO frequency and phase, s_{LFO} is the measured LFO signal, and t is the sampling times of the LFO.

From a visual comparison of the predicted and measured LFO, it was noted that the solver often got stuck in a local minimum. This can be observed in Fig. 9 (top), where the frequencies of the two waveforms clearly do not match. To improve the likelihood of the solver finding the global minimum, it was run multiple times, with a different initial frequency each time. The initial frequency values were chosen by taking the Fourier transform of s_{LFO} and finding the frequency bin with the highest energy, ω_k. The initial frequency was then varied linearly, starting at frequency ω_{k-1} and ending at frequency ω_{k+1}. The parameter predictions that result in the lowest error were then chosen as the LFO parameters.

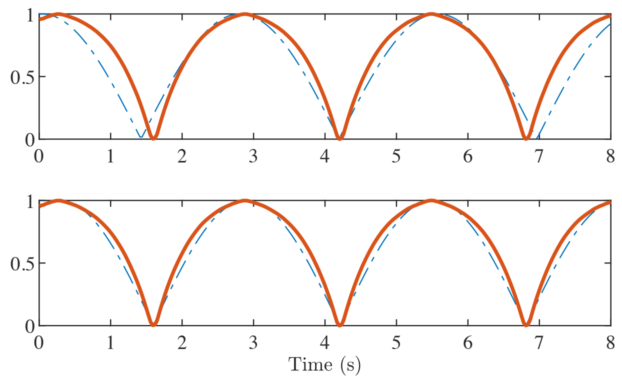

Figure 9: *LFO prediction (dashed line) and measured LFO (solid line), where the solver has found (top) a local minimum and (bottom) the global minimum.*

5.3. LFO Extrapolation

To create the dataset, the measurement signal is immediately followed by guitar audio from the dataset described in Sec. 3.2. The predicted LFO frequency and phase offset from the measurement signal can then be used to predict the LFO signal during the time following or preceding the measurement signal.

As the LFO signal is periodic, even a very small error in the predicted frequency will result in the predicted LFO signal quickly becoming out of sync with the actual LFO signal. Additionally, it is possible that the LFO signal frequency is subject to slight variations due to, for example, slight temperature changes in the circuit components. In light of this, the measurement signal was inserted periodically throughout the dataset. This effectively divides the input guitar audio into segments, with each segment being preceded and followed by a copy of the measurement signal. For each segment, the predicted LFO for the first half is then set according to the parameters predicted during the preceding measurement signal, and the second half is set according to the LFO parameters predicted during the following measurement signal.

6. EXPERIMENTS

Neural network models of two effects pedals were created, the Behringer VP-1 Vintage Phaser and the Donner Jet Convolution Flanger. The phaser pedal has a "Rate" control which sets the LFO frequency, and a "Tone" switch. The flanger pedal has a "Rate" control, "Color" and "Range" controls which adjust the flanger depth and center frequency, respectively, and a mode selector switch, which determines whether the LFO is on ("Normal" mode) or off ("Filter" mode).

To create the datasets, the LFO measurement and prediction method described in Sec. 5 was used, with the measurement signal

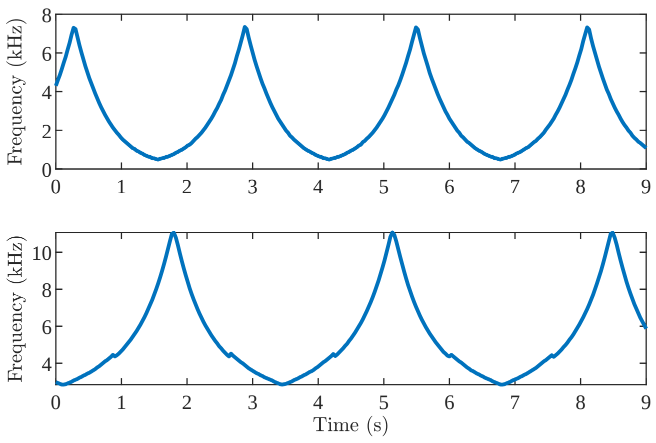

Figure 10: *Measured notch frequency over time for (top) the Vintage Phaser VP-1 and (bottom) the Jet Convolution Flanger.*

being inserted periodically throughout the training dataset.

6.1. Phaser Pedal

The phaser pedal was modelled with the "Rate" set to 3 and the "Tone" switch in the up position. For the phaser pedal LFO, a nine second chirp train, with half a second of silence at the beginning and end was used as the measurement signal. A total of seven minutes of guitar audio was used as the dataset, with the measurement signal being inserted at 20-second intervals throughout the dataset. An example of one of the phaser pedal's notch frequencies over time is shown in Fig. 10 (top).

6.2. Flanger Pedal

For the flanger, the pedal was modelled with the "Color" and "Rate" set to the 12 o'clock position, the "Range" set to the 9 o'clock position, and the mode selector set to "Normal". During measurement of the LFO, we found that it was much harder to get a clean measurement of a notch position over time, most likely due to the large number of notches present. The parameters for the flanger pedal were chosen as they resulted in a cleaner measurement of the LFO. An example of one of the flanger pedal's notch frequencies over time is shown in Fig. 10 (bottom).

A total of seven minutes of guitar audio, with the measurement signal inserted every 20 seconds, was processed through the flanger pedal. From visually inspecting the resulting measured LFOs, many segments contained clearly erroneous data. To ensure accuracy of the training data, only parts of the dataset where a plausible LFO signal were measured were used. The final dataset then consisted of two minutes of audio, which was divided into an 80-second training dataset, a 20-second validation dataset and a 20-second test dataset.

7. RESULTS

For both of the pedals being modelled, four neural networks were trained, with hidden sizes of 8, 16, 32, and 64. The ESR loss achieved on the test dataset for each network is shown in Table 1. For the Phaser, the best ESR loss of 3.1% was achieved by the RNN with hidden size of 64, and the worst ESR loss of 6.7% was achieved by the RNN with hidden size of 8. For the Flanger, the best ESR loss of 1.3% was achieved by the RNN with hidden size of 64, and the worst ESR loss of 11.7% was achieved by the RNN with hidden size of 8. Formal listening tests were not carried out during the completion of this study, but previous listening tests for guitar amplifiers models [4] suggest that an ESR loss of 3% corresponds to very minor differences being perceived between the model and the target device. Informal listening tests also indicate that the models sound very similar to the target devices, and audio examples are available at the accompanying web page [25].

A comparison of spectrograms of the outputs of the phaser and flanger pedals, and the neural network models of them, with hidden sizes of 8 and 64, are shown in Fig. 11 and Fig. 12, respectively. For the phaser, the spectrograms show good agreement between both of the neural network models and the pedal, and the time-varying notches can be seen clearly. For the flanger, there are some visible differences between the smaller neural network and the pedal, where it appears some of the higher frequencies shown are either missing or smeared over time. However, for the larger model, no clear differences can be observed.

Additionally, a short dataset was created for each pedal, with the LFO rate set to a higher frequency that was unseen during training. The spectrograms of the outputs of the phaser and flanger pedals and their models are shown in Fig. 13 and Fig. 14, respectively. Again the spectrograms for each effect type appear to be very similar, indicating that the neural network model has generalised well to the unseen LFO frequency.

A further example in which the LFO frequency is varied continuously from 0.1 Hz to 3 Hz is also included in the accompanying web page [25]. For the Flanger effect, the neural network produces a reasonable sounding effect for the full range of the LFO frequencies tested. However, for the Phaser, some undesired behaviour occurs at the higher LFO frequencies, with the output becoming distorted and the volume fluctuating. It is not clear why this behaviour occurs, but it is likely that including training data with a higher frequency LFO would mitigate this.

8. CONCLUSION

This paper has shown how a neural network can be used to model LFO-modulated effects, such as phaser and flanger pedals. The method was first tested on the toy-problem of a digital phaser model, in which case the same LFO signal used by the effect could be shown to the neural network during training. This test proved that a faithful model of the digital phaser effect can be learned by an RNN using only one minute of training data. Adding more training data does not reduce the error, but increasing the neural network size does.

Real analog phaser and flanger pedals were then modelled by

Table 1: *Error-to-signal ratio for the RNN models of the Vintage Phaser VP1 and the Jet Convolution Flanger pedals. The best results are highlighted.*

Hidden	Number of	ESR	
Size	Parameters	Phaser	Flanger
8	361	6.7%	11.7%
16	1223	3.4%	5.8%
32	4513	3.3%	1.6%
64	17217	**3.1%**	**1.3%**

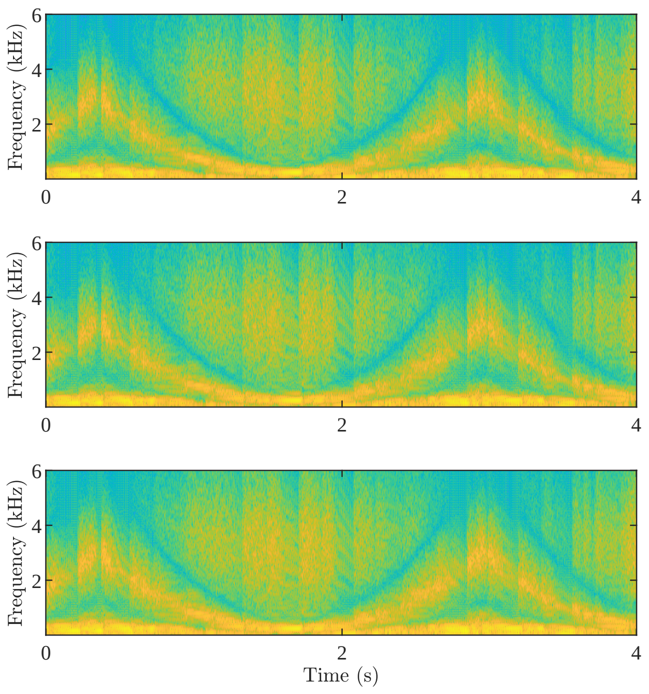

Figure 11: *Spectrogram of phaser effect with guitar signal as input, for (top) the Vintage Phaser VP1 and neural network models with hidden size (middle) 8 and (bottom) 64.*

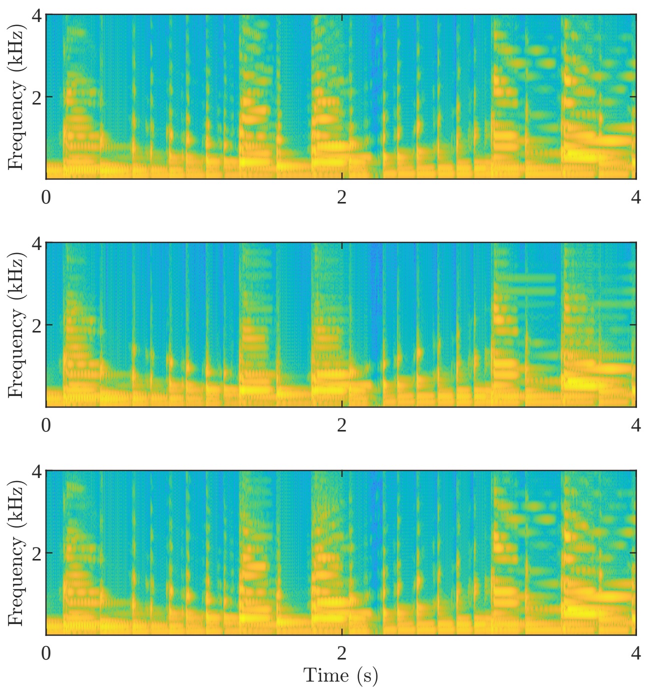

Figure 12: *Spectrogram of flanger effect with guitar signal as input, for (top) the Jet Convolution Flanger and neural network models with hidden size (middle) 8 and (bottom) 64.*

first estimating the frequency and phase of their LFO signal. A sufficiently dense allpass-chirp train was fed to the pedal to sample its frequency response, the frequency trajectories of notches were estimated, and a nonlinear solver was used to fit a signal model to the LFO. This turned out to be more reliable in the case of the phaser pedal than the flanger. The most reliable data obtained from the LFO estimation were used for training neural network models having hidden sizes between 8 and 64.

Using the largest RNN, ESRs of about 3% and 1% were obtained for the phaser and flanger pedal models, respectively. According to our previous studies, these values correspond to hardly audible modelling errors. However, listening confirms that also the smaller models, which have larger errors, sound quite realistic. The RNN models were further validated by modifying the LFO frequency and analyzing their performance. In most cases tested, the networks generalised well to conditions unseen during training. However, for the Phaser model undesirable volume fluctuation was introduced when the LFO rate was increased. The results of this study suggest that many time-varying effects may be modelled successfully with neural networks by first separating the LFO behaviour, with the main challenge being in measuring the LFO signal accurately.

9. REFERENCES

[1] V. Välimäki, S. Bilbao, J. O. Smith, J. S. Abel, J. Pakarinen, and D. Berners, “Virtual analog effects,” in *DAFX: Digital Audio Effects*, U. Zölzer, Ed., pp. 473–522. Wiley, Chichester, UK, second edition, 2011.

[2] A. Bernardini, P. Maffezzoni, and A. Sarti, “Linear multistep discretization methods with variable step-size in nonlinear wave digital structures for virtual analog modeling,” *IEEE/ACM Trans. Audio, Speech, and Lang. Process.*, vol. 27, no. 11, pp. 1763–1776, Nov. 2019.

[3] M. A. Martínez Ramírez, E. Benetos, and J. D. Reiss, “Deep learning for black-box modeling of audio effects,” *Applied Sciences*, vol. 10, no. 2, pp. 638, Jan. 2020.

[4] A. Wright, E.-P. Damskägg, L. Juvela, and V. Välimäki, “Real-time guitar amplifier emulation with deep learning,” *Applied Sciences*, vol. 10, no. 3, Jan. 2020.

[5] F. Eichas, M. Fink, M. Holters, and U. Zölzer, “Physical modeling of the MXR Phase 90 guitar effect pedal,” in *Proc. Int. Conf. Digital Audio Effects (DAFx-14)*, Erlangen, Germany, Sept. 1–5, 2014, pp. 153–158.

[6] K. J. Werner, V. Nangia, A. Bernardini, J. O. Smith III, and A. Sarti, “An improved and generalized diode clipper model for wave digital filters,” in *Proc. Audio Eng. Soc. 139th Conv.* New York, NY, USA, 29 Oct.–1 Nov. 2015.

[7] R. Kiiski, F. Esqueda, and V. Välimäki, “Time-varying graybox modeling of a phaser pedal,” in *Proc. Int. Conf. Digital Audio Effects (DAFx-16)*, Brno, Czech Republic, Sept. 5-9, 2016, pp. 31–38.

[8] C. Darabundit, R. Wedelich, and P. Bischoff, “Digital grey box model of the Uni-Vibe effects pedal,” in *Proc. Int. Conf. Digital Audio Effects (DAFx-19)*, Birmingham, UK, Sept. 2-6, 2019.

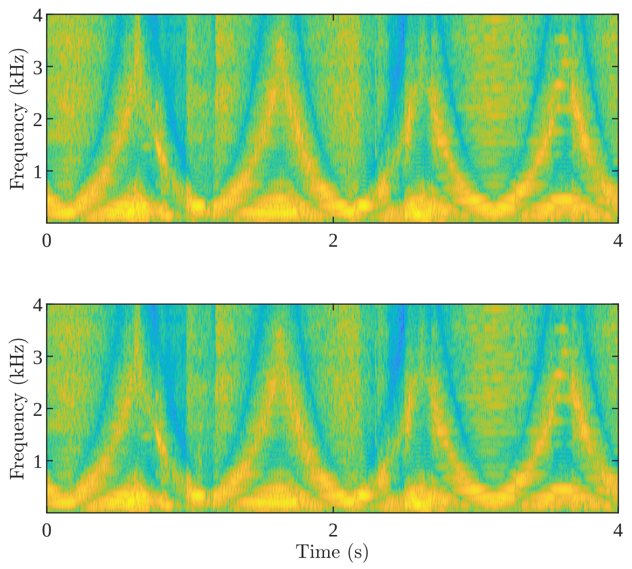

Figure 13: *Spectrogram of (top) the phaser pedal and (bottom) the neural network model outputs with guitar signal as input and an LFO frequency of 1 Hz.*

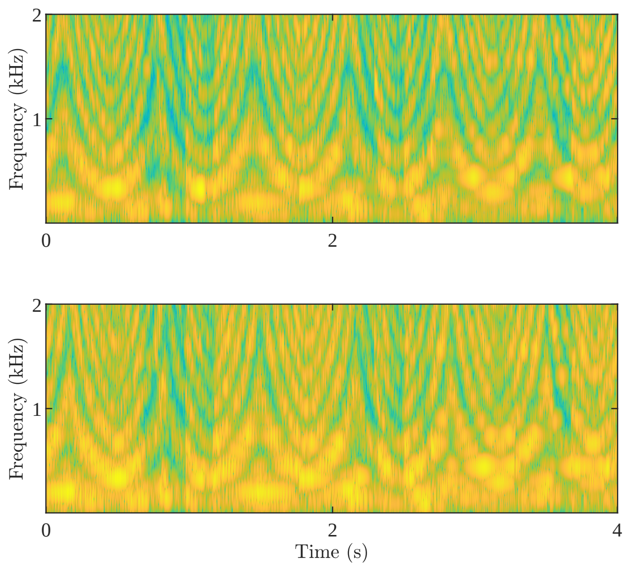

Figure 14: *Spectrogram of (top) the flanger pedal and (bottom) the neural network model outputs with guitar signal as input and an LFO frequency of 1.5 Hz.*

[9] S. Orcioni, A. Terenzi, S. Cecchi, F. Piazza, and A. Carini, "Identification of Volterra models of tube audio devices using multiple-variance method," *J. Audio Eng. Soc.*, vol. 66, no. 10, pp. 823–838, Oct. 2018.

[10] A. Novak, L. Simon, P. Lotton, and J. Gilbert, "Chebyshev model and synchronized swept sine method in nonlinear audio effect modeling," in *Proc. Int. Conf. Digital Audio Effects (DAFx-10)*, Graz, Austria, Sept. 2010, pp. 423–426.

[11] T. Schmitz and J.-J. Embrechts, "Nonlinear real-time emulation of a tube amplifier with a long short time memory neural-network," in *Proc. Audio Eng. Soc. 144th Conv.*, Milan, Italy, May 2018.

[12] Z. Zhang, E. Olbrych, J. Bruchalski, T. J. McCormick, and D. L. Livingston, "A vacuum-tube guitar amplifier model using long/short-term memory networks," in *Proc. IEEE SoutheastCon*, Saint Petersburg, FL, USA, Apr. 2018.

[13] E.-P. Damskägg, L. Juvela, E. Thuillier, and V. Välimäki, "Deep learning for tube amplifier emulation," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP-19)*, Brighton, UK, May 2019, pp. 471–475.

[14] A. Wright, E.-P. Damskägg, L. Juvela, and V. Välimäki, "Real-time black-box modelling with recurrent neural networks," in *Proc. Int. Conf. Digital Audio Effects (DAFx-19)*, Birmingham, UK, Sept. 2-6, 2019.

[15] E.-P. Damskägg, L. Juvela, and V. Välimäki, "Real-time modeling of audio distortion circuits with deep learning," in *Proc. Int. Sound and Music Computing Conf. (SMC-19)*, Malaga, Spain, May 2019, pp. 332–339.

[16] M. Martínez Ramírez and J. Reiss, "Modeling nonlinear audio effects with end-to-end deep neural networks," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, Brighton, UK, 12–17 May 2019, pp. 171–175.

[17] J. Parker, F Esqueda, and A. Bergner, "Modelling of nonlinear state-space systems using a deep neural network," in *Proc. Int. Conf. Digital Audio Effects (DAFx)*, Birmingham, UK, 2–6 Sept. 2019.

[18] M. Martínez Ramírez, E. Benetos, and J. Reiss, "A general-purpose deep learning approach to model time-varying audio effects," in *Proc. Int. Conf. Digital Audio Effects (DAFx-19)*, Birmingham, UK, Sept. 2-6, 2019.

[19] W. M. Hartmann, "Flanging and phasers," *J. Audio Eng. Soc.*, vol. 26, no. 6, pp. 439–443, Jun. 1978.

[20] J. O. Smith, "An allpass approach to digital phasing and flanging," in *Proc. Int. Computer Music Conf.*, Paris, France, 1984.

[21] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, 2019, pp. 8024–8035.

[22] C. Kehling, J. Abeßer, C. Dittmar, and G. Schuller, "Automatic tablature transcription of electric guitar recordings by estimation of score- and instrument-related parameters," in *Proc. Int. Conf. Digital Audio Effects (DAFX-14)*, Erlangen, Germany, Sept. 2014, pp. 219–226.

[23] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learning Representations (ICLR-15)*, San Diego, CA, USA, May 2015.

[24] V. Välimäki, J. S. Abel, and J. O Smith, "Spectral delay filters," *J. Audio Eng. Soc.*, vol. 57, no. 7/8, pp. 521–531, Jul./Aug. 2009.

[25] A. Wright and V. Välimäki, "Neural Modelling of Time-Varying Effects," accompanying webpage, available online at: `http://research.spa.aalto.fi/publications/papers/dafx20-tvfx/`, Accessed: 2020-04-18.