REAL-TIME BLACK-BOX MODELLING WITH RECURRENT NEURAL NETWORKS

Alec Wright, Eero-Pekka Damskägg, and Vesa Välimäki[*]

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

ABSTRACT

This paper proposes to use a recurrent neural network for black-box modelling of nonlinear audio systems, such as tube amplifiers and distortion pedals. As a recurrent unit structure, we test both Long Short-Term Memory and a Gated Recurrent Unit. We compare the proposed neural network with a WaveNet-style deep neural network, which has been suggested previously for tube amplifier modelling. The neural networks are trained with several minutes of guitar and bass recordings, which have been passed through the devices to be modelled. A real-time audio plugin implementing the proposed networks has been developed in the JUCE framework. It is shown that the recurrent neural networks achieve similar accuracy to the WaveNet model, while requiring significantly less processing power to run. The Long Short-Term Memory recurrent unit is also found to outperform the Gated Recurrent Unit overall. The proposed neural network is an important step forward in computationally efficient yet accurate emulation of tube amplifiers and distortion pedals.

1. INTRODUCTION

Virtual analog modelling is an active area of research, which seeks to create software that can accurately emulate popular music hardware, such as instruments, audio effects or amplifiers [1]. Nonlinear systems with memory, such as guitar amplifiers and distortion pedals are particularly challenging to emulate [2, 3]. Generally, the approaches to virtual analog modelling fall into three categories, "white-box" [2, 4, 5, 6], "grey-box" [7, 8], and "black-box" [9, 10] modelling. This paper is concerned with black-box modelling of tube amplifiers and distortion pedals using a recurrent neural network (RNN).

This is a very timely topic, as the first attempts to model nonlinear audio circuits with a Long Short-Term Memory (LSTM) neural network were published last year [11, 12]. Schmitz and Embrechts used a hybrid neural network consisting of a convolutional layer in front of an RNN [12]. Zhang *et al.* tested tube amplifier modelling using an LSTM neural network with many hidden layers but only a few units per layer. They reported that the sound quality of the emulation was not good enough, as there were clear audible differences with respect to the real amplifier [11].

In recent works [13, 14], we adapted the WaveNet convolutional neural network to model nonlinear audio circuits such as tube amplifiers and distortion pedals. In [14] it was shown that the WaveNet model of several distortion effects was capable of running in real time. The resulting deep neural network model, however, was still fairly computationally expensive to run.

In this paper, we propose an alternative black-box model based on an RNN. We demonstrate that the trained RNN model is capable of achieving the accuracy of the WaveNet model, whilst requiring considerably less processing power to run. The proposed neural network, which consists of a single recurrent layer and a fully connected layer, is suitable for real-time emulation of tube amplifiers and distortion pedals.

The rest of this paper is organized as follows. Section 2 discusses the two nonlinear circuits, which are used as target devices in this study, as well as the creation of the training data. Section 3 introduces the neural network architecture that we use for modelling. Section 4 focuses on the training of the RNNs. The real-time implementation of the RNN is presented in Section 5. Section 6 reports experiments and comparisons we have conducted to validate this work. Section 7 concludes the paper.

2. MODELLED DEVICES

Two commonly used types of nonlinear audio circuits are distortion pedals and guitar amplifiers. As such, for this study, we chose to model the Electro-Harmonix Big Muff Pi distortion/fuzz pedal and the Blackstar HT-1 combo guitar amplifier.

The Big Muff is a famous guitar pedal, whose first version was released by Electro-Harmonix in 1969 [15]. Since then, numerous versions of the pedal have appeared, each with slight differences to the original circuit. The Big Muff is known for its heavily distorted sound, which is produced by its two cascaded diode clipping stages. The pedal has a "sustain" knob for controlling the pre-gain, i.e. the gain applied to the signal before the clipping stages, a "volume" knob for controlling the post-gain, as well as a "tone" knob for controlling the shape of the filter in the tone stage. Several researchers have studied the digital modelling of the Big Muff distortion pedal prior to this work [16, 17, 14].

The Blackstar HT-1 is a small 1-Watt vacuum tube amplifier [18]. It has two channels: a high-gain and a low-gain channel. In this work, the high-gain channel was used, as it introduces more distortion to the signal. The amplifier has an unconventional tone stage. The "ISF" tone knob on the amplifier allows for continuous shifting between two distinct tone settings, which the manufacturer describes as the "American" and "British" tones. The amplifier also has a "gain" knob and a "volume" knob, which control the pre-gain and the post-gain respectively.

[*] This research belongs to the activities of the Nordic Sound and Music Computing Network—NordicSMC (NordForsk project number 86892).

2.1. Training Data

One issue which is common to many neural network applications is the collection of training data. Often this is a very labour intensive process, as a large collection of training examples needs to be gathered and labelled. For this application the process is relatively straightforward, as training data can be generated simply by inputting audio signals to each of the devices being modelled. The resulting output can then be recorded from the devices and used as the ground truth.

The input audio files were taken from the guitar and bass guitar datasets[1] [2] described in [19, 20], respectively. The full dataset used consists of 8 minutes and 10 seconds of audio. The dataset was split into a training set of 5 min and 42 s, a validation set of 1 min and 24 s and a test set of 1 min and 4 s. The audio was split so that each data subset contained approximately equal proportions of guitar and bass recordings. All of the training data audio used during this study was recorded at a sampling rate of 44.1 kHz.

The recording was carried out using a MOTU UltraLite-mk3 USB audio interface. One output of the audio interface was connected to the input of the device being measured. The output of the device being measured was recorded by connecting it to one of the inputs of the audio interface. The direct signal coming out of the audio interface was also measured by connecting one of its outputs to one of its inputs. The test audio was then output from both interface output channels and recorded through both input channels. The recorded direct signal from the audio interface and the recorded output signal from the device being measured makes up the input/output pairs which were used during network training and testing.

This process was adapted slightly for the HT-1 amplifier. The HT-1 amplifier has an emulated line out, which applies a speaker cabinet simulation to the signal. For the purposes of this study we are not interested in modelling this, so the speaker output of the amplifier was used instead. To allow for connection from the amplifier speaker output to the audio interface input, a Bugera PS1 power attenuator was used. The HT-1 amplifier speaker output was connected to the power attenuator and the line-out of the power attenuator was connected to the input of the audio interface. Additionally the speaker output of the power attenuator was also connected to a speaker cabinet.

The authors acknowledge that the choice of load will affect the output of the amplifier. Specifically whether the load is resistive or reactive has been shown to influence the output of triode tubes [21]. Whilst the amplifier is connected, indirectly, to a reactive load (the speaker cabinet), it is still thought that the output will be influenced by the presence of the power attenuator. Ideally the amplifier could be connected to a speaker cabinet, and the amplifier output could be recorded directly. This is difficult to achieve in practice. One option is to record the output of the speaker cabinet using a microphone, however this introduces a number of additional effects to the amplifier output. Namely the speaker cabinet nonlinearities and frequency response [22], as well as the coloration of the signal introduced by the microphone and its placement [23]. For these reasons, we chose to use the power attenuator for the collection of the training data.

For each device, the entire dataset was processed five times. Each time the user control being modelled was adjusted. For the HT-1 the control being modelled was the "ISF" control, and for the Big Muff it was the "Tone" control. For each measurement the control was set to one of five equally spaced settings, from the minimum (0) to the maximum (10) possible value.

[1]www.idmt.fraunhofer.de/en/business_units/m2d/smt/guitar.html
[2]www.idmt.fraunhofer.de/en/business_units/m2d/smt/bass_lines.html

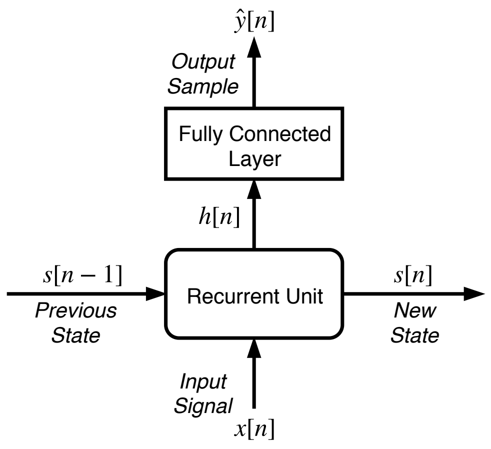


Figure 1: *Proposed neural network architecture, where $x[n]$ is the input signal, $s[n]$ and $s[n-1]$ are recurrent unit states, $h[n]$ is the recurrent unit output and $\hat{y}[n]$ is the neural network's predicted output sample.*

For all recordings the HT-1 "gain" and "volume" knobs were set to 5 and 10 respectively. For the Big Muff the "volume" and "sustain" knobs were set to 10 and 5 respectively.

3. RECURRENT NEURAL NETWORK MODEL

The proposed model is a gated RNN based on either LSTM or Gated Recurrent Units (GRUs). The precise behaviour of LSTM units and GRUs is described in Sections 3.1 and 3.2, this section describes the general behaviour of the model and applies whether the recurrent unit selected is an LSTM or a GRU.

The proposed model consists of two hidden layers, a recurrent unit, followed by a fully connected layer. This architecture is shown in Figure 1. Unlike feedforward neural network layers, recurrent units have a *state*, which is used in the computation of the output and then updated at each time step. The model can be thought of as a function which is trained to predict the current output sample value, based on the current input signal value, the recurrent unit's state and the model's learned parameters:

$$\hat{y}[n] = f(x[n], s[n-1], \theta), \tag{1}$$

where n is the discrete time index, θ are the model's learned parameters and $s(n-1)$ is the recurrent unit's state at the previous time step. In this study, $\hat{y}[n]$ represents the model's prediction of a single output sample and $x[n]$ the unprocessed input signal.

At each time step an output is produced by the recurrent unit and fed into the fully connected layer. The size of the recurrent unit's output is determined by its *hidden size*, which is a model parameter defined by the user. Generally a greater hidden size produces a model capable of emulating more complex behaviour, but also requiring increased computational resources to train and run. The fully connected layer therefore consists of a single neuron, with a number of inputs equal to the recurrent unit hidden size. The output of this layer represents the networks predicted

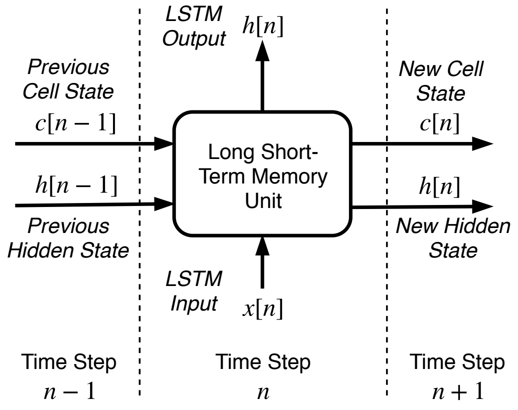


Figure 2: *Diagram of an LSTM unit, where c represents the cell state, h the hidden state and $x[n]$ is the input signal at time step n.*

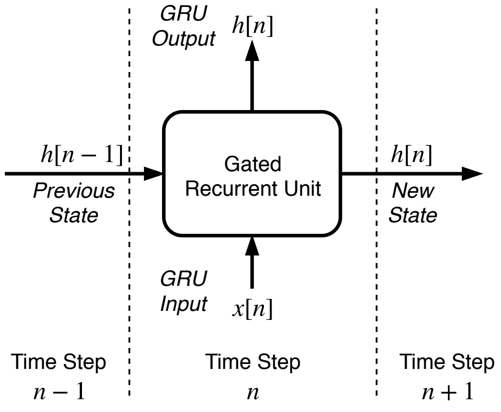


Figure 3: *Diagram of a GRU, where h represents the hidden state and $x[n]$ is the input signal at time step n.*

sample value for that time step. In addition to this, the recurrent unit's state is also updated at each time step, based on the previous state, current input and the model's learned parameters. As each state update is a function of the previous state, this means that the model has a potentially unlimited memory and in practice learns through training the effective memory required for the system being modelled.

The neural network model can also include conditioning, which allows for the emulation of the target device's user controls. To add conditioning, the input signal is extended to include an additional value representing the user control's setting, as suggested in [13]. At each time step the input signal will then be a vector containing the input sample and one or more conditioning values, depending on how many controls are included in the model.

It is possible to add additional recurrent layers before the final fully connected layer, however, preliminary testing indicated that this had little influence on the resulting model's accuracy. As such, the model was limited to just a single recurrent layer for this study.

3.1. Long Short-Term Memory

In an LSTM [24], the unit's state consists of two vectors, the *cell state*, c, and the *hidden state*, h. At each time step, the inputs are the current time step input, $x[n]$, the initial cell state, $c[n-1]$, and the initial hidden state, $h[n-1]$. The LSTM produces two outputs, the updated hidden state, $h[n]$, and the updated cell state, $c[n]$. An LSTM unit is depicted in Figure 2. The outputs are produced according to the following functions:

$$i[n] = \sigma(W_{ii}x[n] + b_{ii} + W_{hi}h[n-1] + b_{hi}), \tag{2}$$

$$f[n] = \sigma(W_{if}x[n] + b_{if} + W_{hf}h[n-1] + b_{hf}), \tag{3}$$

$$\tilde{c}[n] = \tanh(W_{ic}x[n] + b_{ic} + W_{hc}h[n-1] + b_{hc}), \tag{4}$$

$$o[n] = \sigma(W_{io}x[n] + b_{io} + W_{ho}h[n-1] + b_{ho}), \tag{5}$$

$$c[n] = f[n]c[n-1] + i[n]\tilde{c}[n], \tag{6}$$

$$h[n] = o[n]\tanh(c[n]), \tag{7}$$

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.

The LSTM unit state consists of the hidden state and the cell state. The state update of the LSTM unit is determined by the eight weight matrices and eight bias vectors, denoted by W and b respectively in the above equations. It should be noted that the biases in the neural network are simply constant offsets applied during the unit computation, and are in no way related to bias in electronic circuits. These weights and biases make up the learnable parameters of the LSTM unit, which are learned during training. The size of the weight matrices and bias vectors is determined by the input size and the LSTM's hidden size. The updated cell state, $c[n]$, and the updated hidden state, $h[n]$, are used as the initial state for the next time step, $n+1$. The updated hidden state is also used as the output of the LSTM for the current time step, n, which is input into the fully connected layer in our model.

3.2. Gated Recurrent Unit

A GRU [25] is an alternative recurrent unit, which can be used in the proposed architecture shown in Figure 1. In a GRU, the unit state consists of a single *hidden state* vector, h. At each time step, the inputs are the current time step input, $x[n]$ and the initial hidden state, $h[n-1]$. The GRU produces a single output, the updated hidden state, $h[n]$. A GRU unit is depicted in Figure 3. The hidden state is calculated according to the following functions:

$$r[n] = \sigma(W_{ir}x[n] + b_{ir} + W_{hr}h[n-1] + b_{hr}), \tag{8}$$

$$z[n] = \sigma(W_{iz}x[n] + b_{iz} + W_{hz}h[n-1] + b_{hz}), \tag{9}$$

$$\tilde{h}[n] = \tanh(W_{ih}x[n] + b_{ih} + r[n](W_{hh}h[n-1] + b_{hh})), \tag{10}$$

$$h[n] = (1 - z[n])\tilde{h}[n] + z[n]h[n-1], \tag{11}$$

where $r[n]$ is the reset gate, $z[n]$ is the update gate and $\tilde{h}[n]$ is the candidate hidden state.

The GRU unit state consists of the hidden state vector, which is determined by the six weight matrices and six bias vectors, denoted by W and b respectively in the above equations. These weights and biases are the learnable parameters of the GRU. The size of the weight matrices and bias vectors is determined by the input size and the GRU's hidden size. The updated hidden state, $h[n]$, is used as the initial state for the time step, $n+1$, as well as being used as the output of the GRU for the current time step n.

3.3. Fully Connected Layer

The fully connected layer, which can be seen in Figure 1, is not followed by an activation function, so the output sample predicted by the network is simply an affine transformation of the hidden state vector:

$$\hat{y}[n] = W_{fc}h[n] + b_{fc}, \tag{12}$$

where W_{fc} and b_{fc} are the fully connected layer's weight matrix and bias vector respectively and $h[n]$ is the recurrent unit hidden state at time n. As the model outputs a single value at each time step, the fully connected layer consists of just a single neuron. As such the weight matrix reduces to a vector and the bias vector consists of a single value in this case.

4. TRAINING

Recurrent Neural Networks are generally considered to be harder to train effectively in comparison to feedforward networks [26]. As such this section describes, in detail, the training process and techniques used in the completion of this study. All the RNN models produced for this paper were trained using the Adam optimizer [27], with an initial learning rate of 5×10^{-4}.

4.1. Loss Function

The loss function used during training is based on the error-to-signal ratio (ESR) with pre-emphasis filtering, as used in [13, 14]. The ESR is the squared error divided by the energy of the target signal. A pre-emphasis filter was applied to the output and target signals before computing the ESR. The pre-emphasis filter helps the network learn to model the high-frequency content. For a segment of training signal of length N, the pre-emphasised ESR is given by:

$$\mathcal{E}_{\text{ESR}} = \frac{\sum_{n=0}^{N-1} |y_p[n] - \hat{y}_p[n]|^2}{\sum_{n=0}^{N-1} |y_p[n]|^2}, \tag{13}$$

where y_p is the pre-emphasised target signal, and $\hat{y}_p$ is the pre-emphasised neural network output. The pre-emphasis filter was chosen to be a first-order high-pass filter with transfer function

$$H(z) = 1 - 0.85z^{-1}. \tag{14}$$

Additionally, a DC term was added to the loss to reduce the DC offset in the model outputs. The DC loss term is given by:

$$\mathcal{E}_{\text{DC}} = \frac{|\frac{1}{N}\sum_{n=0}^{N-1}(y[n] - \hat{y}[n])|^2}{\frac{1}{N}\sum_{n=0}^{N-1} |y[n]|^2}. \tag{15}$$

The DC term was introduced as early tests showed the model outputs contained a DC offset. The final loss function used for training is given by:

$$\mathcal{E} = \mathcal{E}_{\text{ESR}} + \mathcal{E}_{\text{DC}}. \tag{16}$$

4.2. Truncated Back-Propagation Through Time

When training RNNs it is possible to update the network parameters at each time step, however this comes with a very high computational cost. An alternative approach is to allow the RNN to process an entire sequence and then update the parameters. This involves backpropagating through the entire sequence to find the required gradients, as such this is often referred to as *Backpropagation Through Time* (BPTT) [28]. When modelling very long sequences, updating once per sequence results in slow training as the network parameters are updated infrequently. In addition to this, each update requires backpropagation through the entire sequence, which is computationally expensive.

Truncated BPTT [29] is a method in which parameter updates are carried out during the processing of a sequence. This means that updates can be carried out frequently and the recurrent unit state can persist between updates. The frequency of parameter updates and the number of time steps to run BPTT through for each update are training hyperparameters that are chosen by the user [26]. Our model is trained on audio sampled at a rate of 44.1 kHz, so the sequence length is very long even for a second of audio. As such truncated BPTT was used during the training of the RNNs in this study. The parameter updates were carried out every 2048 samples, with BPTT being run over 2048 time steps each update, as this was found to be an effective update frequency during our early experiments.

4.3. Batch-Processing

The training dataset was split into half-second segments. This was done for two reasons, firstly, by creating a large number of separate sequences, the short sequences can be processed in parallel, greatly reducing the time required to process the entire dataset. Secondly, having short segments allows the dataset to be shuffled at each epoch, which is known to improve network convergence rates [30]. The training data segments were shuffled at the beginning of each epoch and processed in mini-batches of 40 segments. At the start of each mini-batch the recurrent unit's initial state is set to 0. The first 1000 samples are then processed without updating the network parameters, to allow the recurrent unit state to initialise. The remaining samples are then processed, with backpropagation and parameter updates being carried out every 2048 samples. This is repeated until the entire training dataset is processed. The training dataset is then shuffled for the next training epoch.

5. REAL-TIME IMPLEMENTATION

A real-time implementation of the RNN was developed in C++. The implementation was built using the JUCE framework and the Eigen library for matrix and vector operations. JUCE can be used to build real-time audio plugins in the common VST, AU, and AAX formats supported by modern digital audio workstations. The developed plugin can be used to process audio through the trained models.

5.1. Recurrent Layer Computations

Algorithm 1 shows how the state update for the LSTM, as given by Equations (2)–(7), is carried out in practice. The eight matrix multiplications in Equations (2)–(5), which involve the input x and hidden state h, are performed as two bigger matrix multiplications. Furthermore, the eight bias terms in these equations can be combined to a single bias term.

The result of these matrix multiplications and the bias addition are stored in the vector v, which contains the non-activated values of the input gate $i[n]$, forget gate $f[n]$, output gate $o[n]$, and the candidate cell state $\tilde{c}[n]$. The hyperbolic tangent and logistic sigmoid activation functions are applied to v elementwise

Algorithm 1 LSTM State Update

```
Require: Layer input $x$, hidden state $h$, cell state $c$, hidden size $N$, input weight $W_i$, state weight $W_h$, bias $b$, [conditioning term $b_{\text{cond}}$]
1: $v \leftarrow W_i x + W_h h + b$
2: if $b_{\text{cond}}$ was given then $v \leftarrow v + b_{\text{cond}}$
3: for each $i$ in $[0, N[$ do
4:     $c[i] \leftarrow \sigma(v[N+i])c[i] + \sigma([v[i]) \tanh(v[2N+i])$
5:     $h[i] \leftarrow \sigma(v[3N+i]) \tanh(c[i])$
6: return $h, c$
```

when computing the new cell state c and the new hidden state h. The GRU computations are carried out in a similar fashion.

In the real-time implementation, the conditioning is not given to the layer in the input vector x in the same way as it was described in Section 3. Instead, since the conditioning is not typically updated at audio rate, processing power is saved when the effect of the conditioning in the layer activation is computed separately, and stored into a vector b_{cond}, which is then added to v at each time step.

5.2. Computational Load

The computational load of the implementation was tested with different RNN configurations. Models using LSTM units and GRUs were tested with different hidden state sizes. The results are shown in Figure 4. The processing speed is reported in terms of compute time required to process one second of audio, which was estimated by running the models on an Apple iMac with an 2.8 GHz Intel Core i5 processor. The GRU models run faster than LSTM models using the same hidden size. Using a single recurrent layer, an LSTM network runs faster than real time up to a hidden size of 160, whereas a GRU runs faster than real time up to a hidden size of 192.

6. COMPARISON OF MODELS

Models of the devices were created using two neural network architectures, the RNN model described in this paper, and the convolutional WaveNet-like neural network described in [13, 14].

For the RNN model, both the LSTM and GRU recurrent unit types were tested, with the hidden size ranging from 32 to 96. Each model was trained for 20 hours on a Nvidia Tesla V100 Graphics Processing Unit (GPU). The validation error was calculated every other epoch. Once the training was complete, the test loss was calculated using the model parameters from the epoch with the lowest validation loss. An example of the validation and training loss during training is shown in Figure 5. It can be seen that around epoch 250 there is a large increase in both the training and validation loss. It is not entirely clear why this sudden increase in loss occurs, however this was found to be fairly common during the training of the RNN models. As the spike includes both the training and validation loss, it does not indicate that the network is overfitting to the training dataset.

For the WaveNet model, the three configurations presented in [14] were used. The models vary in the number of convolutional layers and in the number of channels in the convolutional layers, i.e. the hidden size. All models use gated activations. Training the WaveNet models took approximately two to three hours on the

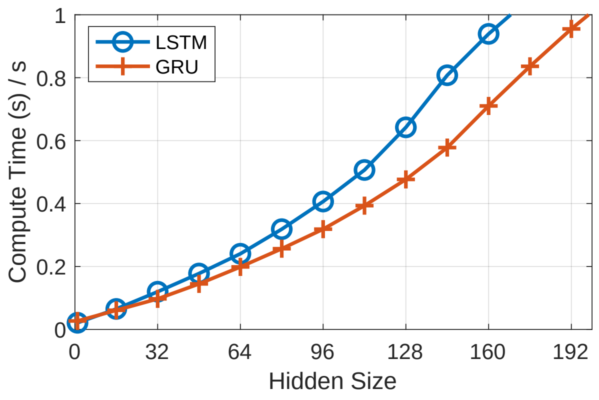


Figure 4: *Compute time of the real-time implementation for processing 1 s of audio at a 44.1-kHz sample rate, using different hidden sizes.*

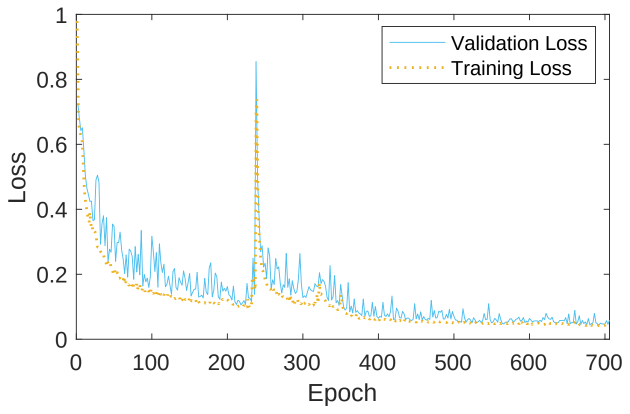


Figure 5: *Validation and Training loss for an RNN model of the Big Muff pedal.*

GPU. The processing speed of the WaveNet models was estimated using the C++ implementation presented in [14].

Model accuracy is evaluated in terms of the ESR loss, computed without pre-emphasis filtering, achieved on the unseen test dataset. The processing speed is reported in terms of compute time required to process one second of audio. A comparison between the RNN and WaveNet models of the HT-1 and the Big Muff is shown in Tables 1 and 2 respectively.

In terms of speed the results show a clear improvement for the RNN in comparison to the WaveNet model. The fastest RNN requires just 0.097 s to process a second of audio. The slowest RNN takes 0.41 s, which is less than the fastest WaveNet which takes 0.53 s to process a second of audio.

In terms of accuracy, the results vary depending on the device being modelled. In the case of the Big Muff pedal, the most accurate RNN model shows a considerable improvement over the most accurate WaveNet model. For the HT-1 amplifier the most accurate WaveNet outperforms the RNN. A comparison of audio processed by the best performing RNN model and the HT-1 amplifier is shown in the time domain in Figure 6 and in the frequency domain in Figure 7. The plots show good agreement between the

Table 1: *Error-to-signal ratio and processing speed for the Wavenet and the proposed GRU/LSTM models of the HT-1 Amplifier. The best results are highlighted.*

Model	Hidden Size	Layers	Number of Parameters	ESR	Time (s) / s of Output
WaveNet	16	10	24065	2.2%	0.53
WaveNet	8	18	11265	1.2%	0.63
WaveNet	16	18	43265	**0.79%**	0.91
GRU	32	1	3393	3.3%	**0.097**
LSTM	64	1	17217	1.8%	0.24
LSTM	96	1	38113	1.1%	0.41

Table 2: *Error-to-signal ratio and processing speed for the Wavenet and proposed LSTM models of the Big Muff pedal. The best results are highlighted.*

Model	Hidden Size	Layers	Number of Parameters	ESR	Time (s) / s of Output
WaveNet	16	10	24065	11%	0.53
WaveNet	8	18	11265	9.9%	0.63
WaveNet	16	18	43265	9.2%	0.91
LSTM	32	1	4513	10%	**0.12**
LSTM	48	1	9841	6.1%	0.18
LSTM	64	1	17217	**4.1%**	0.24

model output and the target device, however in future work listening tests should be conducted to verify the performance of the models.

6.1. Conditioning

The test loss was also computed separately for each of the five conditioning values the RNN models were trained on. For each model, the conditioning value with the greatest test loss was compared to the average test loss over all the conditioning values. The most extreme deviation from the average test loss was 16%, with the average deviation being 8%. This demonstrates that the models achieve a similar level of accuracy for each of the conditioning values.

On the devices modelled, the control knobs are continuous, allowing the user to select any value between the minimum and maximum setting. The models have been trained on data sampled at just five of the possible control knob settings, however the trained model is not limited to just these five settings. In order to test how well the RNNs can extrapolate to conditioning values not seen in training, a model was trained with the center conditioning value removed from the training dataset. The test lost was then computed for the unseen center conditioning value, and compared to the average test loss. For both the HT-1 and Big Muff the test loss for the unseen conditioning value was within 0.2% of the average. Informal listening tests also indicate that the models produces realistic outputs when the conditioned value is set to a value not seen during model training.

Audio examples of the models are available at the accompanying web page [31].

7. CONCLUSIONS

This work has compared the performance of two types of neural network for the real-time emulation of a distortion pedal and a vacuum tube amplifier. The single layer RNN model presented in this paper requires much less processing power to run in comparison to a previously presented WaveNet-like convolutional neural network, and can be run in real time. The accuracy of the RNN model

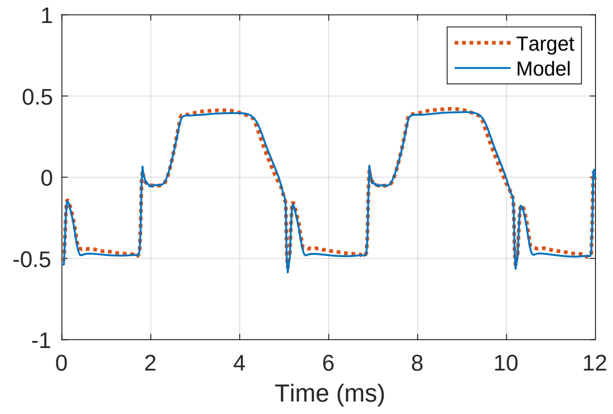
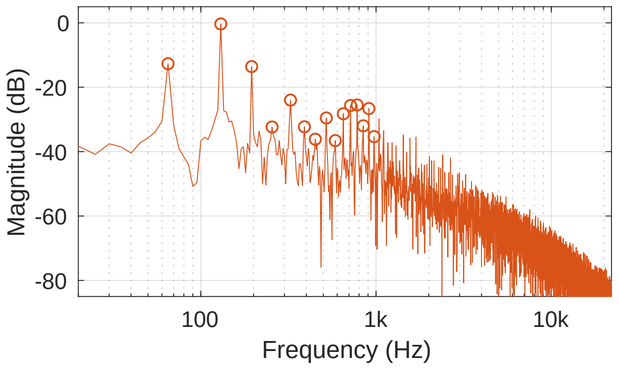


Figure 6: *Waveform of a guitar sound processed through the HT-1 guitar amplifier, and through the most accurate RNN model.*



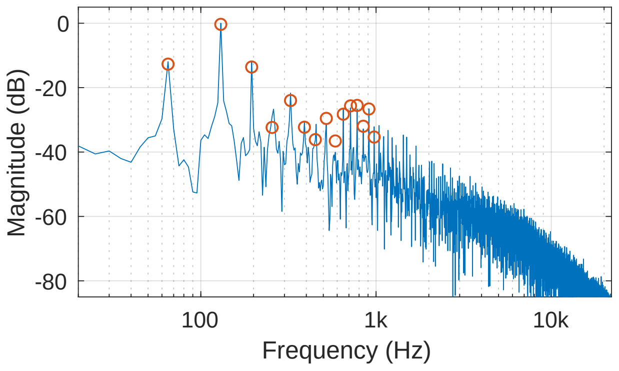


Figure 7: *Spectrum of a guitar sound processed (top) through the HT-1 guitar amplifier, and (bottom) through the most accurate RNN model. The circles indicate the level of the first 15 harmonics in the upper spectrum.*

was also comparable to or better than the WaveNet, depending on the device being modelled. Whilst listening tests have been carried out previously for the WaveNet-like model [13], future work should include listening tests to validate the proposed RNN model.

Of the two types of the recurrent unit tested, the GRU was shown to run slightly faster than the LSTM, for equivalent hidden sizes. However, we recommend the LSTM network over the GRU, as our tests indicate that for LSTM and GRU networks of roughly equivalent running speed, the LSTM generally achieved higher accuracy.

8. ACKNOWLEDGMENTS

We acknowledge the computational resources provided by the Aalto Science-IT project.

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] M. Karjalainen and J. Pakarinen, "Wave digital simulation of a vacuum-tube amplifier," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP-06)*, Toulouse, France, May 2006, vol. 5, pp. 153–156.

[3] J. Pakarinen and D. T. Yeh, "A review of digital techniques for modeling vacuum-tube guitar amplifiers," *Computer Music J.*, vol. 33, no. 2, pp. 85–100, 2009.

[4] F. Santagata, A. Sarti, and S. Tubaro, "Non-linear digital implementation of a parametric analog tube ground cathode amplifier," in *Proc. Int. Conf. Digital Audio Effects (DAFx-07)*, Bordeaux, France, Sept. 2007, pp. 169–172.

[5] R. C. D. Paiva, S. D'Angelo, J. Pakarinen, and V. Välimäki, "Emulation of operational amplifiers and diodes in audio distortion circuits," *IEEE Trans. Circ. Syst. II*, vol. 59, no. 10, pp. 688–692, Oct. 2012.

[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, Oct. 2015.

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

[8] F. Eichas and U. Zölzer, "Gray-box modeling of guitar amplifiers," *J. Audio Eng. Soc.*, vol. 66, no. 12, pp. 1006–1015, Dec. 2018.

[9] 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.

[10] 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.

[11] 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, Apr. 2018.

[12] 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.

[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] 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.

[15] The Big Muff Pi Page, "Evolution of the Big Muff Pi circuit," Available online at: http://www.bigmuffpage.com/Big_Muff_Pi_versions_schematics_part1.html, Accessed: 2019-04-05.

[16] K. J. Werner, V. Nangia, J. O. Smith, and J. S. Abel, "Resolving wave digital filters with multiple/multiport nonlinearities," in *Proc. Int. Conf. Digital Audio Effects (DAFX-15)*, Trondheim, Norway, Nov.–Dec. 2015, pp. 387–394.

[17] F. Eichas and U. Zölzer, "Black-box modeling of distortion circuits with block-oriented models," in *Proc. Int. Conf. Digital Audio Effects (DAFx-16)*, Brno, Czech Republic, Sept. 2016, pp. 39–45.

[18] Blackstar Amplification, "HT-1 guitar amplifier – Product page," Available online at: https://www.blackstaramps.com/uk/ranges/ht-1, Accessed: 2019-04-05.

[19] 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.

[20] J. Abeßer, P. Kramer, C. Dittmar, and G. Schuller, "Parametric audio coding of bass guitar recordings using a tuned physical modeling algorithm," in *Proc. Int. Conf. Digital Audio Effects (DAFx-13)*, Maynooth, Ireland, Sept. 2013, pp. 154–159.

[21] J. Pakarinen and M. Karjalainen, "Enhanced wave digital triode model for real-time tube amplifier emulation," *IEEE Trans. Audio Speech Lang. Process.*, vol. 18, no. 4, pp. 738–746, May 2010.

[22] D. T. Yeh, B. Bank, and M. Karjalainen, "Nonlinear modeling of a guitar loudspeaker cabinet," in *Proc. Int. Conf. Digital Audio Effects (DAFx-08)*, Espoo, Finland, Sept. 2008, pp. 89–96.

[23] A. Roginska, A. U. Case, A. Madden, and J. Anderson, "Measuring spectral directivity of an electric guitar amplifier," in *Proc. Audio Eng. Soc. 132nd Conv.*, Budapest, Hungary, Apr. 2012.

[24] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Dec. 1997.

[25] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint*, Dec. 2014, arXiv:1412.3555 [cs.NE].

[26] I. Sutskever, *Training Recurrent Neural Networks*, Ph.D. thesis, University of Toronto, Ontario, Canada, 2013.

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

[28] P. J. Werbos, "Backpropagation through time: What it does and how to do it," *Proc. IEEE*, vol. 78, no. 10, pp. 1550–1560, Oct. 1990.

[29] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, no. 2, pp. 179–211, Mar. 1990.

[30] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," in *Neural Networks: Tricks of the Trade*, pp. 437–478. Springer, 2012.

[31] A. Wright, E.-P. Damskägg, and V. Välimäki, "Real-time black-box modeling with recurrent neural networks," accompanying webpage, available online at: http://research.spa.aalto.fi/publications/papers/dafx19-rnn/, Accessed: 2019-18-06.