F3

Faculty of Electrical Engineering
Department of Cybernetics

Bachelor's Thesis

Learning the Musical Instrument Amplifier Model with Neural Networks

Jakub Lukeš

Supervisor: Ing. Patrik Vacek
Study program: Open Informatics
Branch of study: Computer and Information Science
August 2021

BACHELOR'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Lukeš Jakub** Personal ID number: **474725**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Open Informatics**

Branch of study: **Computer and Information Science**

II. Bachelor's thesis details

Bachelor's thesis title in English:

Learning the Musical Instrument Amplifier Model with Neural Networks

Bachelor's thesis title in Czech:

Učení modelu zesilovače hudebního nástroje pomocí neuronových sítí

Guidelines:

Outputting signal from electric musical instrument is constructed with complicated circuits inside a musical amplifier. These amplifiers are described with specific parameters and have unique sounds for each type and price category. The target of this thesis is a software model of a real amplifier.
1) Get familiar with the functioning of the musical amplifier (guitar for example), with its inputs, outputs and effects.
2) Design a way to model the effects of the amplifier from raw instrument signals.
3) Find a publicly available dataset of sounds for learning the model of the chosen amplifier.
4) Discuss the usage of different types of neural networks for learning the amplifier.
5) Compare the resulting similarity of the learned model to the sounds of the real amplifier.

Bibliography / sources:

[1] Wright, A.; Damskägg, E.-P.; Juvela, L.; Välimäki, V. Real-Time Guitar Amplifier Emulation with Deep Learning. Appl. Sci. 2020, 10, 766.
[2] Comunità, Marco & Stowell, Dan & Reiss, Joshua. (2020). Guitar Effects Recognition and Parameter Estimation with Convolutional Neural Networks.
[3] Sigtia, Siddharth & Benetos, Emmanouil & Dixon, Simon. (2015). An End-to-End Neural Network for Polyphonic Music Transcription. IEEE/ACM Transactions on Audio, Speech, and Language Processing. 24. 10.1109/TASLP.2016.2533858.

Name and workplace of bachelor's thesis supervisor:

Ing. Patrik Vacek, Vision for Robotics and Autonomous Systems, FEE

Name and workplace of second bachelor's thesis supervisor or consultant:

Date of bachelor's thesis assignment: **08.01.2021** Deadline for bachelor thesis submission: **13.08.2021**

Assignment valid until: **30.09.2022**

Ing. Patrik Vacek Supervisor's signature	prof. Ing. Tomáš Svoboda, Ph.D. Head of department's signature	prof. Mgr. Petr Páta, Ph.D. Dean's signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

Date of assignment receipt	Student's signature

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, 13 August 2021

Acknowledgements

I would like to thank my supervisor, Ing. Patrik Vacek, for the guidance and support I received during the preparation of this thesis.

Abstract

Guitar amplifiers play a key role in rock music because of the overdrive and other effects unique to each model. They are a costly piece of musical equipment, especially when considering the vacuum valve models. This work explores the options to learn and emulate any guitar amplifier using artificial neural networks. One recurrent and one convolutional neural network architectures were implemented and trained on real guitar recordings. Their accuracy was then evaluated objectively and with listening tests. It was shown that the presented neural networks can create accurate emulations of the overdrive effects with only small imperfections. This work did not aim to create a real-time application, but it could be useful for post-processing.

Keywords: recurrent neural network, convolutional neural network, WaveNet, LSTM, guitar amplifier

Abstrakt

Kytarové zesilovače hrají klíčovou roli v rockové hudbě kvůli tzv. overdrive a jiným efektům, které jsou unikátní pro každý model. Představují zásadní investici do vybavení hudebníků, především pokud se jedná o elektronkové modely. Tato práce zkoumá možnost učení a emulace libovolného kytarového zesilovače pomocí umělých neuronových sítí. Byly implementovány jedna rekurentní jedna konvoluční neuronová síť, které poté byly trénovány na skutečných kytarových nahrávkách. Jejich přesnost pak byla vyhodnocena objektivně i pomocí poslechových testů. Výsledky ukazují, že představené neuronové sítě dokáží věrně napodobit overdrive efekt jen s drobnými nedokonalostmi. Tato práce se nevěnovala použití v reálném čase, výsledky ale mohou být najít uplatnění v postprodukci.

Klíčová slova: rekurentní neuronová síť, konvoluční neuronová síť, WaveNet, LSTM, kytarový zesilovač

Překlad názvu: Učení modelu zesilovače hudebního nástroje pomocí neuronových sítí

Contents

Figures

Tables

Chapter 1

Introduction

Electric or bass guitar with applied sound effects has been a major theme of rock music ever since its beginnings in the 1950s. Back then, guitar amplifiers were built with vacuum valves, semiconductors were not yet available on the market. Despite the fact that all consumer electronics have switched to solid state designs long ago, vacuum valve amplifiers still remain popular among musicians. The limited production capacity of vacuum valves of course drives the price of such amplifiers up.

A solid state amplifier might be just enough for a beginner musician, but having only one can still be too restrictive. What if the musician wants some variety or even chain multiple effects together? Another interesting problem arises when someone likes particular guitar sound in a song, they want to imitate it, but it is not possible to find which device and which settings were used to make it. Figuring out the correct combination by hand is almost impossible, so it would be useful to get help from a computer.

Artificial neural networks have been on the rise over the last years and it seems like they can be the solution to many problems. The goal of this thesis is to find out whether neural networks can really learn a musical amplifier effect from a sound file, what network architecture is the most suited for this task, and whether an emulated effect can be distinguished from the real one.

This thesis is divided into five chapters. The following chapter briefly explains the inner workings of a valve guitar amplifier and other effects units. After that, the current state of research is summarised and the proposed neural networks are introduced. Chapter three tells everything about the practical execution, from dataset preparation and model implementations to outlining the experiments and introducing the evaluation criteria. In chapter four, the the models are compared through the experimental results. Finally, in chapter five the results are discussed and all findings are summarised.

Chapter 2

Theory

This chapter introduces the reader to the concepts that are later used in this work. First, the functioning of traditional valve guitar amplifiers and similar devices is described more in depth. After that, the state of research in the field of effects modelling is summarised. The following sections then further explain the methods.

2.1 Traditional Hardware

2.1.1 Vacuum Valve Amplifiers

Guitar amplifiers are used to boost and distort audio signal from an electric guitar or bass guitar. The traditional type is the valve 'combo' amplifier (also called tube amplifier), combining an amplifier based on vacuum valves (tubes) and a speaker in one cabinet. The purpose of valves in the circuit is to amplify input audio signal, which becomes distorted (clipped) when the signal strength exceeds a certain threshold. The amount of gain, also labelled as overdrive, can be regulated with a knob. Other properties, such as equaliser levels, master volume, and sometimes the amount of reverb, can be controlled as well. Different circuit designs and vacuum valve models significantly influence the sound of distortion.

Triode, tetrode, and pentode valves with three, four, or five active electrodes, respectively, are commonly used. Triodes have only one control grid; more grids were added to allow higher operating frequencies and higher gain without affecting stability. [1] The rest of the circuit consists just of passive components. Semiconductors are not present since they were not yet widely available around 1950s, when rock music came into existence.

2.1.2 Solid-state Amplifiers

Vacuum valves were technologically surpassed by transistors, which are smaller, more energy-efficient, and cheaper to manufacture. This also meant that solid-state amplifiers with semiconductor components were introduced. Nevertheless, even today some musicians still prefer vacuum valve models. Musical amplifiers are therefore one of the last applications where vacuum valves are still in use in the 21st century.

The reason for that is very subjective: musicians claim that old amplifiers have a characteristic, so-called 'valve sound', which is described as

softer than the one produced by solid-state devices. This phenomenon was researched and even though some differences in sound were confirmed, audio experts suggest that output transformers might have a more significant impact on that than a mere presence of vacuum valves. [2]

2.1.3 Effects Units

Audio effects can also be produced by standalone effects units, often rack-mounted or in the form of a pedal (stompbox). They come in a wide variety of effects, such as distortions, modulations, delay, reverb, sustain etc. The devices are designed to be easily chained, which allows for complex effect modelling. Pedal units are very compact and can be easily regulated with a press of foot, while rack-mounted units offer more controls.

The most advanced devices in this category are multi-effect processors and modelling amplifiers. These contain digital signal processing circuits, a single device can model vast amount of effects and even valve amplifiers. The parameters can be very complex, therefore preset saving and loading is incorporated. An example of such unit is the Kemper Profiler [3].

2.2 Effects Modelling with Software

Two different approaches can be used for emulating a device that has some inputs and outputs. The 'white-box' modelling works with the knowledge of inner workings of the device and re-creates its analogue circuits with numerical methods. Such facsimile should operate very similarly to the original if the non-linear equations are solved correctly [4].

Contrary to that, the 'black-box' approach knows nothing about the device's internals, instead only the input-output mapping is known. The model then needs to be designed and tuned from scratch to produce similar results. This method gained on popularity particularly with the recent advances of artificial neural networks. Therefore, this is the approach that will be further explored in this work.

2.2.1 State of Research

If we further consider only black-box modelling methods with artificial neural networks, two trends in network designs can be seen. First, the Recurrent Neural Networks (RNN) are great for making time series predictions as they can process any sequence of inputs while remembering their internal state. The Long Short-Term Memory (LSTM), possibly the most popular subtype of RNNs, was used in [5, 6] where it was concluded that the models closely matched the real amplifier. It was also discovered that LSTM networks perform better than Gated Recurrent Unit (GRU) networks, which is

another vastly used type of RNNs.[7].

Another option is to use a Convolutional Neural Network (CNN), namely the feedforward WaveNet [8, 9]. Recently, the WaveNet and LSTM networks were compared with focus on real-time use [10]. No major differences in quality were found if the networks are sufficiently large.

It was also explored how different perceptual pre-emphasis filters, applied before computing a loss, affected the subjective accuracy of produced audio [11].

As of today, a few production-ready Virtual Studio Technology (VST) plugins with models learned by neural networks are already available. For example, the Neural DSP company uses proprietary neural networks to train their commercial models [12]. Some authors of [10] are affiliated with this company. On the other hand, the GuitarML [13] project offers open-source real-time models based on both the WaveNet and RNN. Users can either use a pre-trained model or train their own with provided scripts.

2.2.2 Recurrent Neural Networks

A Recurrent Neural Network (RNN) unit/cell iterates over a sequence of inputs and at the same time generates the output sequence step by step. The unit remembers its internal (hidden) state, which can be seen as another input/output when the operations are unrolled in time. The operation is illustrated in Figure 2.1. Multiple cells can be layered in more complex RNNs, i.e. the prediction of one cell is given as input to another one.

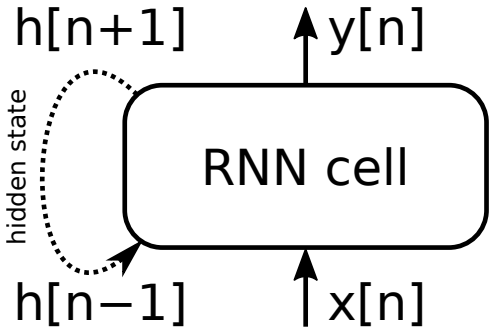

Figure 2.1: A simple recurrent neural network with one cell at time step n. Output sequence y is generated from input sequence x and hidden state h. The hidden state h is updated in each step.

Long Short-Term Memory

So far, the internals of an RNN cell were not discussed, since several architectures exist. The Long Short-Term Memory (LSTM) network, first introduced in [14], was one of the chosen method for experiments in the thesis.

An LSTM cell keeps two internal states, referred to as the *hidden state* (h) and the *cell state* (c). Their size is determined by the *hidden size* hyperparameter. The main four internal building blocks (yellow color in Figure 2.2) are called the *forget gate* (f), the *input gate* (i), the *candidate cell state* ($\tilde{c}$), and the *output gate* (o). These operations are defined as:

$$f(t) = \sigma(W_f[h(t-1), x(t)] + b_f), \tag{2.1}$$

$$i(t) = \sigma(W_i[h(t-1), x(t)] + b_i), \tag{2.2}$$

$$\tilde{c}(t) = \tanh(W_{\tilde{c}}[h(t-1), x(t)] + b_{\tilde{c}}), \tag{2.3}$$

$$o(t) = \sigma(W_o[h(t-1), x(t)] + b_o), \tag{2.4}$$

where weights W and biases b are learnable parameters of the cell and σ is the logistic sigmoid function. Afterwards, new values of the cell state and hidden state are calculated:

$$c(t) = f(t)c(t-1) + i(t)\tilde{c}(t), \tag{2.5}$$

$$h(t) = o(t)\tanh(c(t)). \tag{2.6}$$

The hidden state $h(t)$ is also the cell's prediction $y(t)$. [15]

The forget gate prevents the network from being affected by an input value for too long, which is useful for modelling an overdrive effect without any echoes. In [7] the LSTM network was compared with the Gated Recurrent Unit (GRU) network

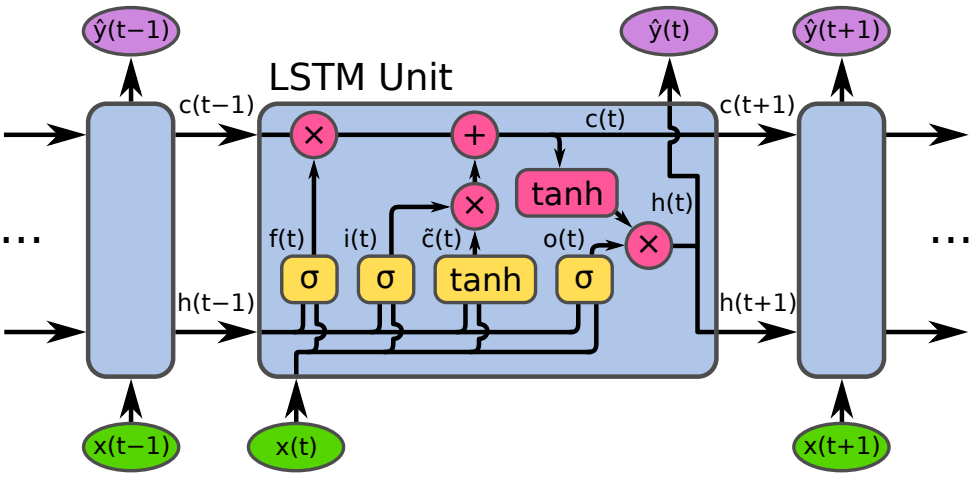

Figure 2.2: A diagram of an LSTM unit unrolled in time. In each step, a value from the input sequence x (green) and the cell state h are processed through gates (yellow). The gate outputs and cell state c are combined with pointwise operations (magenta) to get an updated cell state and hidden state. Finally, the new hidden state $h(t)$ is added to the predicted sequence $\hat{y}$ (violet).

2.2.3 Convolutional Neural Networks

Convolutional Neural Networks (CNN) belong in the category of feed-forward neural networks. There are no loops in the data flow, as opposed to the recurrent neural networks. The principle of CNNs is based on a mathematical operation called convolution.

Convolution

Let x be a discrete input, for example, sampled audio with a specific sample rate. Let t be an integer time argument. The *discrete convolution* of discrete functions x and w is

$$\begin{aligned}(w * x)(t) &= \sum_{n=-\infty}^{\infty} w(n)x(t-n) \\ &= \sum_{n=-\infty}^{\infty} w(t-n)x(n) = (x * w)(t).\end{aligned} \tag{2.7}$$

Function w is also called the *kernel*. [16]

Common intuition for convolution in machine learning is shown in **??**: The kernel is represented as a finite matrix, which is being moved on top of the input. For each kernel position, a dot product of the kernel and the overlapping section of input is calculated. Finally, the dot product is written to the output in the same position where the kernel is currently placed.

From the mathematical perspective, it is important to note that **??** does not actually represent convolution, but a similar operation called cross-correlation. The main difference is that in convolution, the kernel is flipped (see Equation 2.7, notice the minus sign in the index). Convolution is also commutative, while cross-correlation is not. [16] Functions referred to as *convolution* are implemented as a cross-correlation in many popular machine learning libraries. [17, 18]

Given discrete real functions w and x, the *discrete cross-correlation* $w \star x$ is defined as

$$\begin{aligned}(w \star x)(t) &= \sum_{n=-\infty}^{\infty} \overline{w(n)}x(t+n) \\ &= \sum_{n=-\infty}^{\infty} w(n)x(t+n),\end{aligned} \tag{2.8}$$

where $\overline{w[n]}$ is the complex conjugate of $w[n]$.

If we restrict the kernel w_N to have a finite size N, Equation 2.8 can also be rewritten as

$$(w_N \star x)(t) = \sum_{n=0}^{N-1} w_N(n)x(t+n). \tag{2.9}$$

WaveNet

WaveNet is a generative convolutional neural network that was proposed in [19]. It was originally designed for text-to-speech tasks, but authors suggest that the network could be used for many other audio generation problems, since it works with the same input signals. Suitability of this network for guitar amplifier emulation was later confirmed by Damskägg and others in [8] and in the followups [9, 10].

The distinct feature of the WaveNet are dilated convolutions, which are visualised in Figure 2.3. The first layer is an input time series. In each further layer, the space between inputs from the previous layer is doubled, which increases the receptive field without significantly increasing computational complexity. The example pattern in this figure can be described as $d_k = \{1, 2, 4, 8\}$.

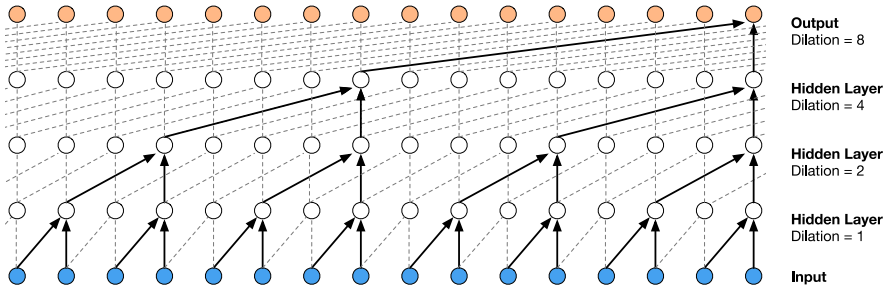

Figure 2.3: Dilated convolutional layers of the WaveNet neural network. Figure taken from [19].

To further improve model performance, the dilation pattern can be repeated: e.g. $d_k = \{1, 2, 4, \ldots, 128, 1, 2, 4, \ldots, 128\}$.

The WaveNet uses a gated activation function

$$z = \tanh(W_f * x) \odot \sigma(W_g * x), \tag{2.10}$$

where $*$ is the convolution operation, $\odot$ is the element-wise multiplication, $\sigma(\cdot)$ is the logistic sigmoid function, and W_f, W_g are the filter and gate learnable convolution kernels, respectively.

Chapter 3

Methodology

This chapter describes the practical approach – implementation of the neural networks, the training process, and the evaluation experiments. Some interim results are already presented in this chapter, while the final comparison is done in chapter 4.

3.1 Dataset

To train an artificial neural network, an appropriate dataset is required. During supervised learning, the network's goal is to tune its internal parameters so that the predictions are close to the provided target predictions. The training set should be sufficiently large and diverse so that the network can become well generalised, meaning that it has a good performance even on input it has never seen. If the predictions on validation or test set have significantly worse accuracy than on the training set, the network is overfitted.

Since no real devices were available during the thesis preparation, a publicly available dataset was used. The Guitar Amplifier dataset [20] contains recordings of chromatic scales, chords, and a few short songs, which were processed through 5 different guitar amplifiers with several settings. MATLAB scripts, provided with the dataset, can be used to extract a smaller subset.

Two overdrive effects were chosen for modelling: the Mesa Boogie Express 5:50+ channel Crunch at gain 5, and the Blackstar HT-5 Metal channel Disto at gain 3. For both of them, the chromatic scale, chords, and *La Bamba* song were used for training, the blues scale for validation and the blues song for testing. This yields 2 min 37 s of training data.

Two more audio effects, Delay and Tremolo, were added in order to test the models' adaptability to other than distortion effects. Initially it was considered to use the IDMT-SMT-Audio-Effects dataset [21], which contains guitar sounds and 11 audio effects, but the downside is that only single tones and single chords are present in it. Therefore, the two desired effects were designed in Audacity editor and applied to the aforementioned Guitar Amplifier dataset.

The Delay effect was created by using the Echo plugin with the delay time of 0.1 s and decay factor 0.5, which means that the sound is repeated after 100 ms at half volume. The Tremolo effect is a volume function, defined as a sinusoid with the frequency of 5.5 Hz. Its *wet level* parameter, which

sets the mute intensity in each period, was set to 60 %.

All WAV files were converted to NumPy arrays with librosa [22] audio processing library. To allow training in batches, each array was split into 100 ms (4410 sample) sequences. The resulting shape of the 3-dimensional array is *(batches, 1, 4410)*, the middle dimension represents number of channels. Normalisation was then performed by applying formula

$$x'(i) = \frac{x(i) - \mu}{\sigma} \tag{3.1}$$

to the inputs x, where μ is the mean value (average) and σ the standard deviation of the training set. The new standardised time series x' now has 0 mean and standard deviation 1. This technique is commonly used to aid the neural network with learning.

3.2 Understanding the Effects

To better understand what happens to the sound after it passes through a guitar amplifier, a simple visualiser was developed in Python. Such spectrogram can be even used for automatic effects recognition [23]. First, Short-time Fourier transform (STFT) is computed on the audio signal, which means that a Fast Fourier transform (FFT) is performed on short overlapping windows. The chosen parameters for STFT were 1024 samples window length ($\sim$ 23 ms at a sample rate of 44 100 Hz) and 512 samples window hop length. From this we get a time-frequency spectrogram, which states what amplitude each frequency has at each time frame.

Since human ear is more sensitive to pitch changes at lower frequencies, the Mel scale was designed in such a way that an equal distance in pitch anywhere on it is always perceived as equally distant by a listener. By specifying a number of Mel bands, in this case 128, the frequency domain of Fourier transform is divided into equally sized bins on the Mel scale. Finally, the amplitude (power) should be converted to logarithmic dB scale, because otherwise the quiet sections would not be visible.

The resulting spectrogram of the guitar sound before and after passing through an amplifier, created with help of the librosa Python package, can be seen in Figure 3.1. At first glance, it is obvious that the overdrive effect adds some noise well above the original maximal frequency. The sound of Mesa 5:50+ feels more sharp than of the HT-5 Metal, which is caused by higher frequencies up until 20 kHz being present. Interestingly, there is a visible spike in the final reverberation of Mesa 5:50+ at 11.5 s, not present in other audio samples. The change in timbre is audible, but an explanation for it is unknown.

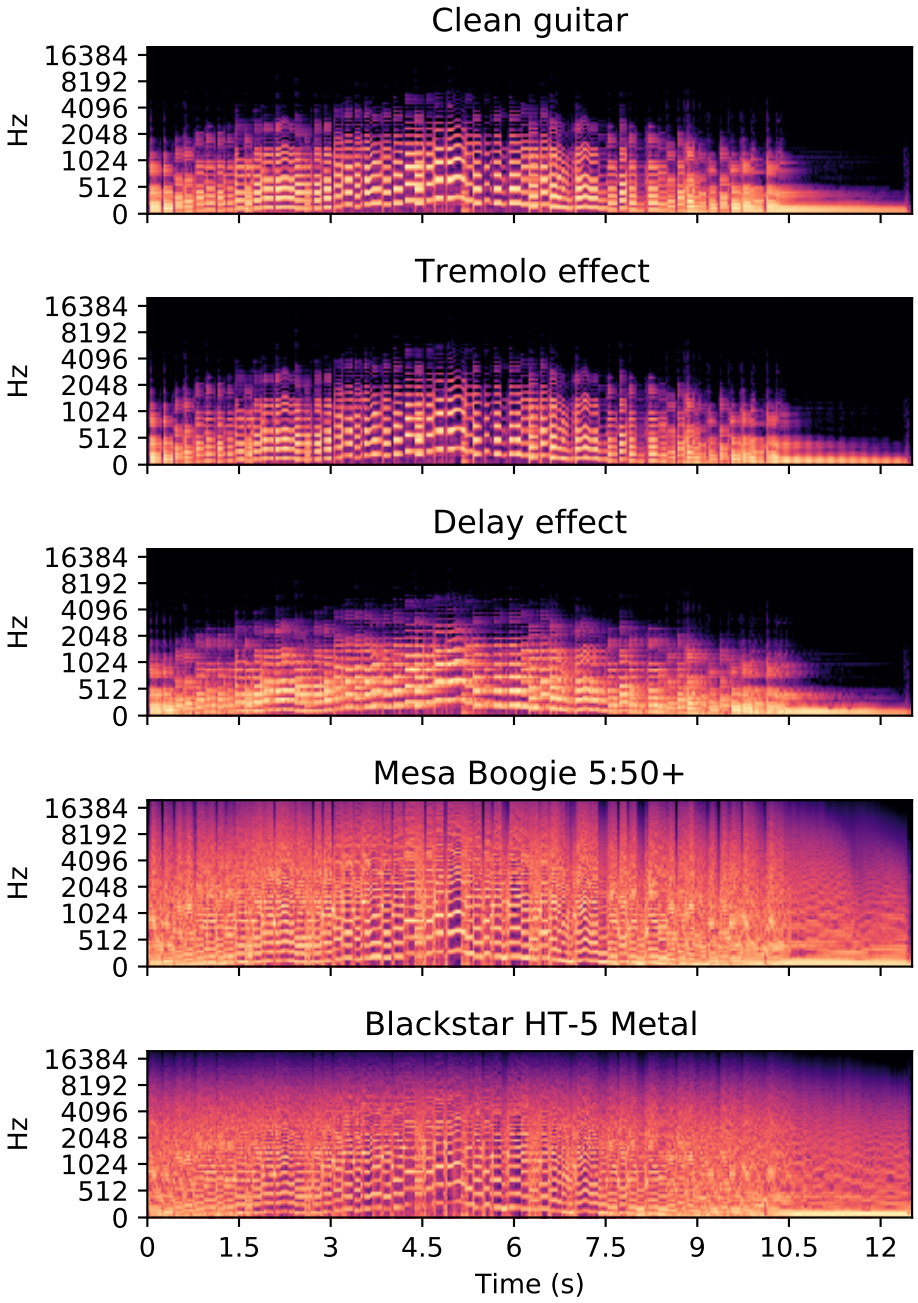

Figure 3.1: Guitar sounds with and without audio effects, visualised with a Mel spectrogram. Strong distortion in the high frequencies can be seen in the last two overdrive effects.

3.3 Performance Measurement Techniques

3.3.1 Loss Functions

A loss function is a certain metric which tells the neural network how well it performs. It is also used in the backpropagation step, when the network's parameters are updated. Many loss functions exist, each of them might be suited better for a different type of data. Four were chosen for evaluating the accuracy of predicted audio.

The Mean Absolute Error (MAE), also known as the L1 loss, is a classic loss function commonly used in machine learning. It is the arithmetic mean distance between predictions and targets, or as an equation

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|, \tag{3.2}$$

where n is the length of the data, y is the expected value, $\hat{y}$ is the predicted value.

Another common regression loss is the Mean Squared Error (MSE) or the L2 loss. It is similar to the L1 loss, but the squared difference penalises larger errors and outliers more significantly. A square root of the MSE is called the Root Mean Squared Error (RMSE), its advantage is that such error is represented in the same units as the data.

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|^2} \tag{3.3}$$

The Error-to-Signal Ratio (ESR) loss was proposed in [8] to measure audio signals. Written as

$$ESR = \frac{\sum_{i=1}^{n}|y_i - \hat{y}_i|^2}{\sum_{i=1}^{n}|y_i|^2}, \tag{3.4}$$

it can described as the squared error divided by the target signal energy.

Finally, in [7] it was proposed to also measure the DC offset in addition to the ESR loss. This leads to the combined $ESR + DC$ loss, where

$$DC = \frac{|\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)|^2}{\frac{1}{n}\sum_{i=1}^{n}|y_i|^2}. \tag{3.5}$$

3.3.2 Pre-Emphasis

In Figure 3.1, it can be seen that most of the sound energy is concentrated in the lower frequencies of the spectrum. On the other hand, the overdrive effect has a significant impact on the frequencies above 10 kHz,

which should be correctly accounted for in the evaluation. To compensate for this when calculating a loss or accuracy, high-pass filters are used in audio processing to amplify the higher frequencies. Such process is called pre-emphasis filtering. First, the filter is applied to both the expected and predicted values, then the loss is computed from pre-emphasised data. An exception is made in case of the ESR+DC loss function, where pre-emphasis is used only for the ESR part.

In [11], three perceptual filters were proposed and compared for training a guitar amplifier model, along with the case of not using pre-emphasis at all. Practically, all the filters were ready to use thanks to the auraloss library [24], they can be easily plugged into the training or validation loop as simple PyTorch 1D convolutions. Because of that, the use of pre-emphasis filters does not have a negative impact on processing time.

First, the first-order high-pass filter (HP) has been the usual choice for speech processing for a long time [25]. Its operation can be described with an equation

$$y(n) = x(n) - \alpha x(n-1), \tag{3.6}$$

where $y(n)$ is the current output of the filter, $x(n)$ is the current input sample, and $x(n-1)$ is the previous input sample. The coefficient α usually lies between $0.8 < \alpha < 1$; values 0.85 and 0.95 were chosen for the RNN and WaveNet, respectively, in previous research [10, 11]. At 44.1 kHz sampling

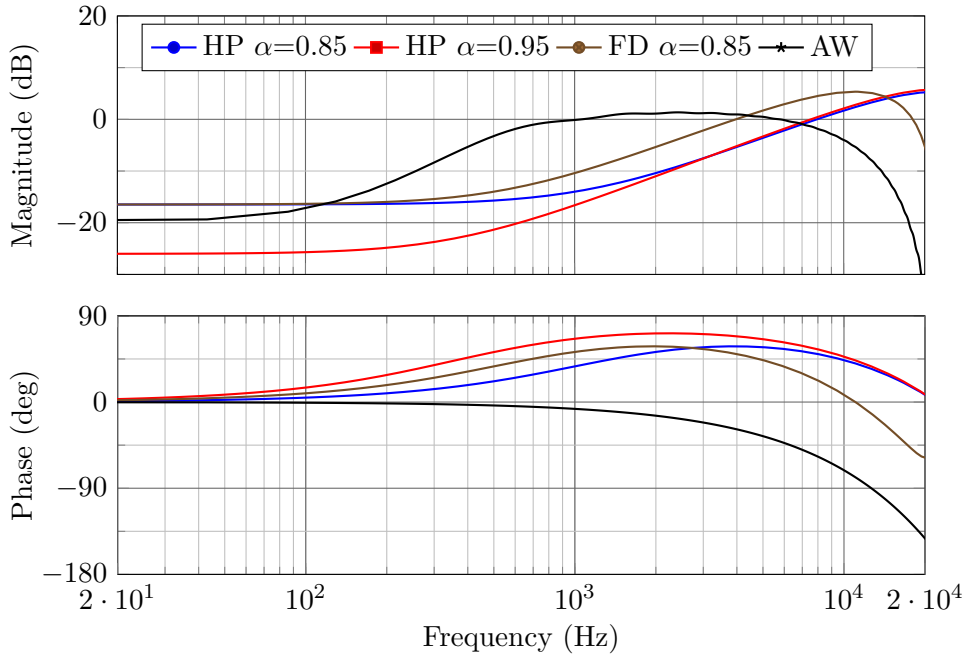

Figure 3.2: Bode plot showing the frequency response of tested pre-emphasis filters at 44.1 kHz sampling rate.

rate, this filter attenuates frequencies below cca 8 kHz and the frequencies above that are slightly amplified.

The second pre-emphasis filter is the folded differentiator (FD), proposed in [11]. An output of this filter is computed as

$$y(n) = x(n) - \alpha x(n-2), \tag{3.7}$$

which means that the current and the 2nd-to-last input values are used for filtering the output. This works as a band-pass filter with the peak at around 10 kHz. The intuition behind this is to attenuate the importance of frequencies above 15 kHz where the human hearing gets limited.

Finally, the A-weighting curve filter (AW), defined in the international standard IEC 61672:2003, represents the human-perceived loudness across the frequency range. When a device reports noise levels in dBA units, it means that the measurement was done after passing the signal through the A-weighting filter. The curve is defined for continuous signal in quite a complicated way, hence it was approximated as a 101-tap Finite Impulse Response (FIR) filter using least-squares error minimisation.

Magnitude and phase shift of the frequency response of these filters is visualised with a Bode plot in Figure 3.2.

3.4 Neural Networks Implementation

All the neural networks were implemented in Python with use of the PyTorch [26] machine learning library. Adam optimiser [27] with the default decay coefficients $\beta_1 = 0.9, \beta_2 = 0.999$ was used for gradient descent. Learning rates were different for each neural network.0 Models were trained on a single NVIDIA GeForce GTX 1080Ti graphics card provided by the Department of Cybernetics computing server.

3.4.1 Long Short-Term Memory

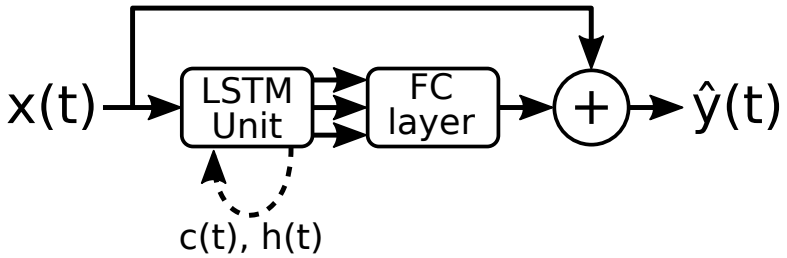

Figure 3.3: Schematic of the LSTM recurrent neural network with a skip connection. x is the sequence of inputs, $\hat{y}$ are the predictions, and c, h are the cell state and hidden state respectively.

The Long Short-Term Memory (LSTM) network was developed independently in PyTorch. The model, shown in Figure 3.3, consists of a single LSTM cell, followed by a fully connected layer, and a skip connection that should mitigate the vanishing gradients problem. Because RNNs are also vulnerable to the exploding gradients problem, gradient clipping is performed before each optimiser step.

The *hidden size* hyperparameter affects shapes of the hidden state and cell state vectors, as well as the predicted value. Therefore, the fully connected layer shrinks it down to a single dimensional time series prediction.

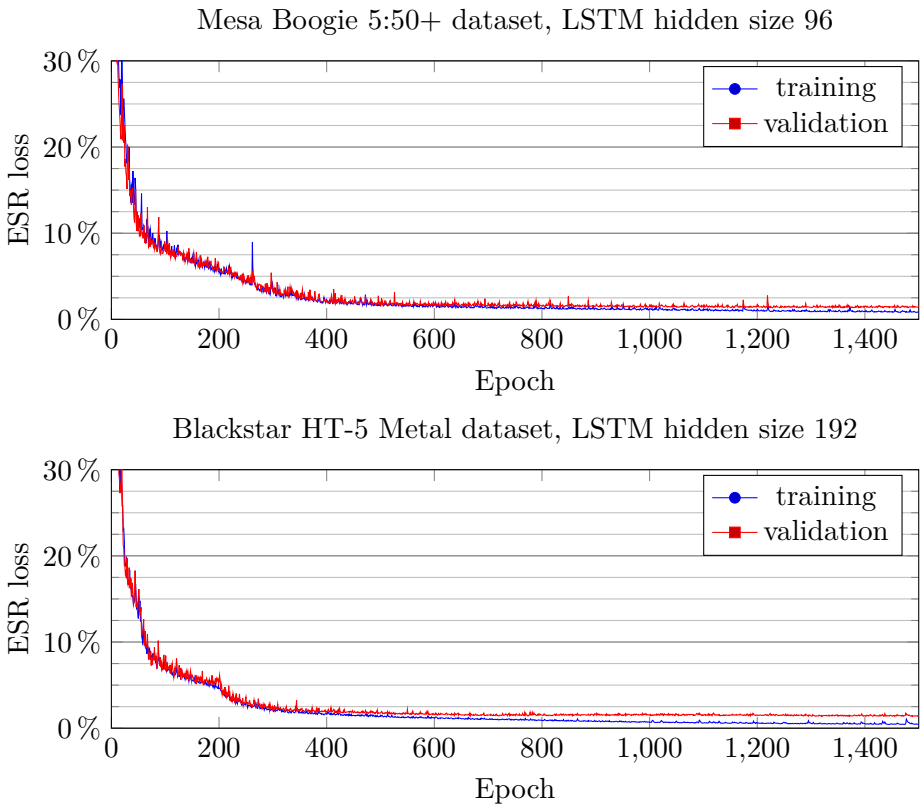

Figure 3.4: Evolution of training and validation loss of the RNN during training.

By performing a grid search, optimal values for the batch size and learning rate hyperparameters were found to be 56 and 0.001, respectively. The rate at which the training and validation losses decrease during network training can be seen in Figure 3.4. For next experiments, training was limited to 1000 epochs to save some time as the validation loss did not improve further.

The LSTM network's performance was then evaluated with hidden sizes ranging from 16 to 192 on the Mesa Boogie 5:50+ dataset. For each of the hidden sizes, the network was trained ten times and the median validation loss was chosen as the performance index. The results are shown in Figure 3.5.

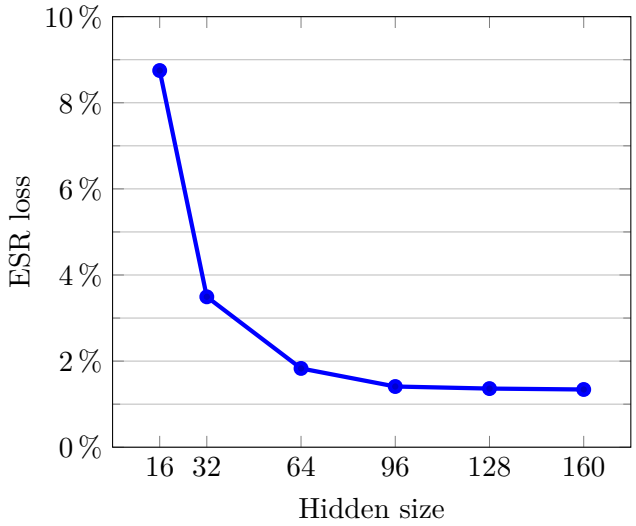

Figure 3.5: ESR validation loss of the LSTM network with different hidden sizes. Almost no improvement can be seen beyond hidden size 96.

Large changes in validation loss can be seen at small hidden sizes, but the differences get gradually smaller. Interestingly, the increase in hidden size did not make the training process significantly longer, 1000 epochs took cca 1 h 20 min for the hidden size 16 and 1 h 35 min for the hidden size 192.

Only the networks with hidden sizes 64 and 96 (LSTM-64 and LSTM-96) were used in later experiments.

3.4.2 WaveNet

Since the dilated convolutions might be tricky to understand and correctly implement, it was decided to use an existing open-source WaveNet implementation. The PedalNet [28] project, which uses the PyTorch Lightning [29] machine learning library, is specifically designed for the task of guitar effects emulation. For this thesis, minor edits to the code needed to be done to make it work with the latest version of PyTorch Lightning and a different dataset format. After some experimenting, learning rate 0.0001 and batch size 48 were chosen as hyperparameters.

Experiments were done with three dilation patterns, inspired by [10]:

$$d_k = \{1, 2, 4, \ldots, 512\} \text{ (10 } \times \text{ 1 layers)},$$
$$d_k = \{1, 2, 4, \ldots, 256, 1, 2, \ldots, 256\} \text{ (9 } \times \text{ 2 layers), and}$$
$$d_k = \{1, 2, 4, \ldots, 128, 1, 2, \ldots, 128, 1, 2, \ldots, 128\} \text{ (8 } \times \text{ 3 layers).}$$

For each dilation pattern, an ablation study was done on the number of convolution channels.

Performance of each of the WaveNet sizes is shown in Figure 3.6. With too little convolution channels, the network cannot learn the effect well, but

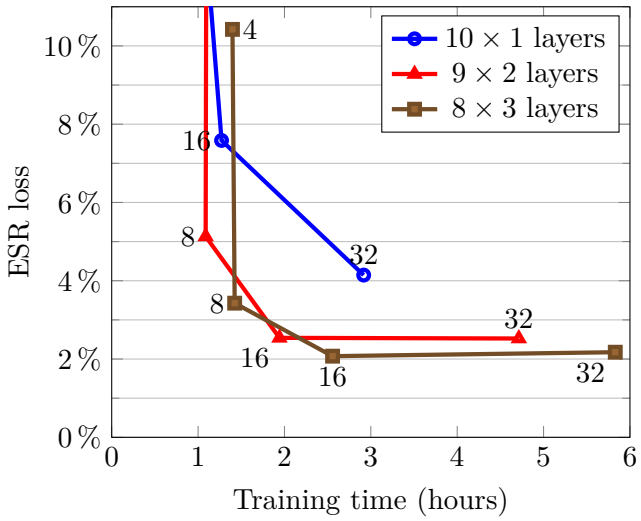

Figure 3.6: ESR validation loss of various WaveNet model sizes on the Mesa 5:50+ validation dataset. The numbers at nodes indicate number of convolution channels.

increasing the number of learnable parameters significantly increases the time needed for training. When choosing the optimal model complexity, both the accuracy and training time were considered. Based on the presented results, the 18-layer and 24-layer WaveNets, both with 16 convolution channels, were chosen for further experiments. Relation between the training and validation loss during training is shown in Figure 3.7.

3.5 Experiments

First, a grid search experiment was conducted to find the best combination of a loss function and a pre-emphasis filter for learning. The study was done on LSTM networks with hidden size 64 because they were faster to train than WaveNets. With 4 loss functions (MAE, MSE, ESR, ESR+DC) and 4 pre-emphasis filter options (no filter, A-weighting, first-order high-pass, folded differentiator), and two guitar amplifier models (Mesa Boogie 5:50+, Blackstar HT-5 Metal), this meant the total of $4 \times 4 \times 2 = 32$ training configurations. As the learning process is non-deterministic, it was repeated 5 times with each configuration and the one with the lowest achieved validation loss was included in the results. Comparison of all configurations was done on the validation set with 16 metrics (all combinations of loss functions and pre-emphasis filters).

Next, the RNN and WaveNet architectures were directly compared The selection criteria for this experiment were as follows:

- Two WaveNets which have shown the best performance in Figure 3.6

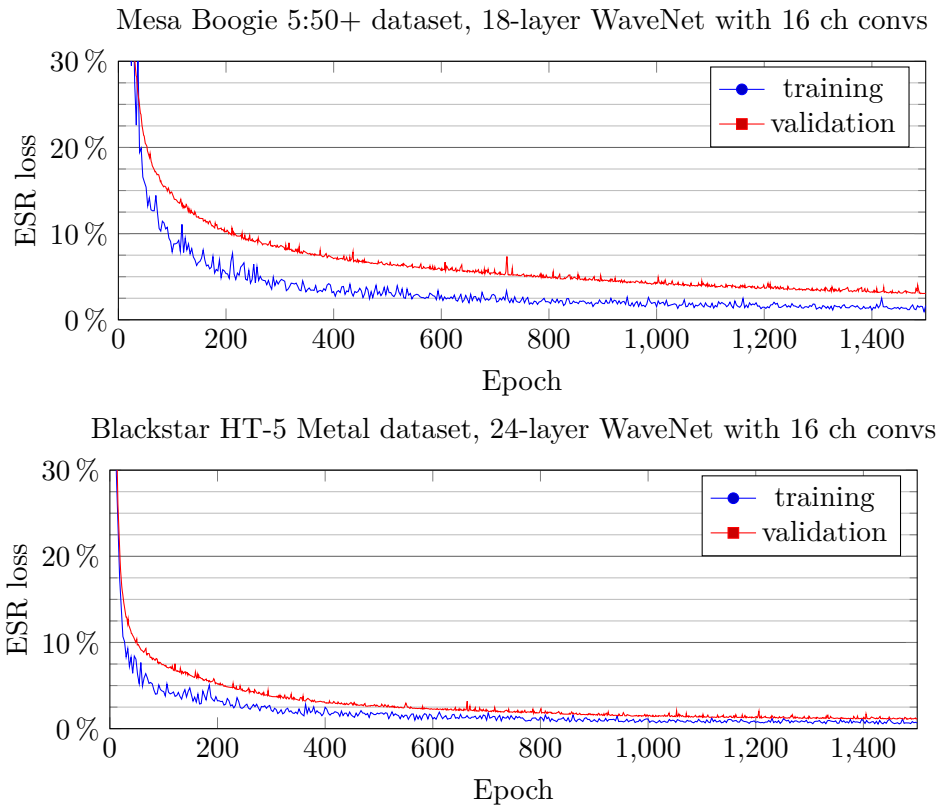

Figure 3.7: Evolution of training and validation loss of the WaveNet during training.

- An LSTM network with hidden size 64 and the best combination of loss function and pre-emphasis

- An LSTM network with hidden size 96 and the worst combination of loss function and pre-emphasis

- An LSTM network with hidden size 96 and the best combination of loss function and pre-emphasis

Performance of all models was measured on the test set using the RMSE and ESR metrics without pre-emphasis and for both overdrive effects.

Finally, the models mentioned in the previous experiment were also trained on the Tremolo and Delay effect datasets. Performance evaluation was done in the same way as in the previous experiment, i.e. RMSE and ESR test losses without pre-emphasis.

3.6 Listening Tests

Loss functions give an objective measurement of how close a predicted waveform is to the expected one, however, in the end it is more important how big of a difference can be heard by listeners. For example, a vast number of predictions can have the exact same loss or accuracy, but each of them would sound differently. That is why listening tests and the subjective results from them were added to the model evaluation.

The Multi Stimulus test with Hidden Reference and Anchor (MUSHRA) method, defined in the ITU-R BS.1534-3 Recommendation [30], was used to conduct the listening tests. In each experiment, participants are instructed to grade test samples with a score between 0 (Bad) and 100 (Excellent), depending on how close they sounds to the provided reference. Among the test signals, the reference and two low-passed anchored signals are hidden with the purpose of having the best and worst signals well defined. In theory, participants should rate the hidden reference with a score of 100, while the anchors should get the worst scores relative to other samples.

Test sessions were performed remotely by using the webMUSHRA [31] web-based testing interface, which can be easily configured with a single YAML file. The ITU Recommendation mandates the same equipment and test conditions for all participants, but in this case participants were instructed to be in a quiet room and to use their headphones. Schedule of each test session is outlined in Appendix A.

Chapter 4

Results

4.1 Loss Function and Pre-Emphasis Training Combinations

The purpose of this experiment was to find the best combination of a loss function and pre-emphasis filter for training the LSTM-64 model. There were 32 test configuration and their performance was measured with 16 metrics. The raw results are presented in a table in Appendix B due to its size. However, losses computed with different loss functions cannot be directly compared, so it was needed to come up with a way to compare them relative to each other.

A simple scoring system was therefore used, where for each type of measurement, the configuration achieving the lowest validation loss received 15 points and the worst one got 0 points. These relative scores are shown in

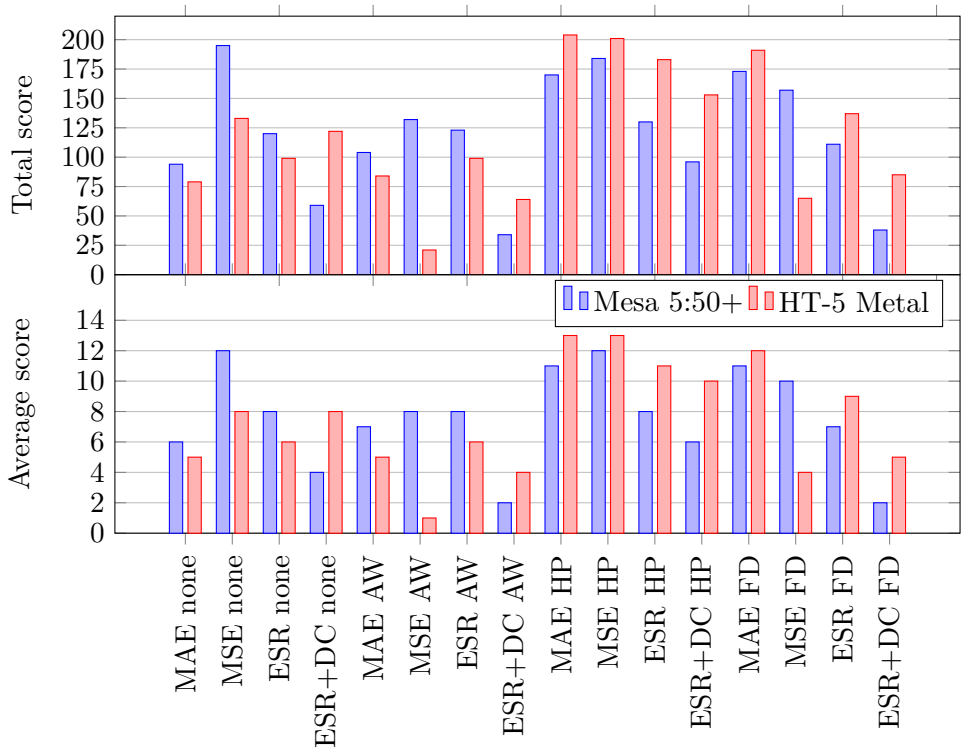

Figure 4.1: Total and average scores (from Table 4.1) achieved by the LSTM-64 trained with 16 combinations of loss functions and pre-emphasis filters

Table 4.1. To get the final score of a training configuration, the sum or the arithmetic mean of all sub-scores was calculated. Both of these are plotted in Figure 4.1, where the training configurations can be easily compared.

It appears that the best training loss function is the Mean Squared Error (MSE), which achieved the lowest validation loss in 75 % of all measurements regardless of the overdrive effect. When looking for the best training pre-emphasis function, the first-order high-pass filter got the highest score

Training		Validation Loss + Pre-Emphasis															
Preemp.		none				AW				HP				FD			
	Loss	a	b	c	d	a	b	c	d	a	b	c	d	a	b	c	d
Mesa Boogie 5:50+																	
none	a	14	12	12	12	10	2	2	2	7	4	4	3	4	2	2	2
	b	**15**	**15**	**15**	**15**	11	13	13	13	10	10	10	10	11	11	11	12
	c	13	14	14	14	8	5	5	5	9	7	7	7	3	3	3	3
	d	10	10	10	10	2	1	1	1	4	2	2	2	1	1	1	1
AW	a	9	6	6	6	**15**	11	11	11	5	1	1	1	9	4	4	4
	b	4	7	7	7	13	**15**	**15**	14	1	5	5	5	7	9	9	9
	c	5	8	8	8	14	14	14	**15**	2	3	3	4	5	7	7	6
	d	3	1	1	1	7	7	7	7	0	0	0	0	0	0	0	0
HP	a	12	13	13	13	9	6	6	6	**15**	13	13	13	14	8	8	8
	b	8	9	9	9	5	10	10	10	14	**15**	**15**	**15**	13	14	14	14
	c	6	5	5	5	3	4	4	4	13	14	14	14	8	10	10	11
	d	7	3	3	3	1	3	3	3	12	11	11	11	6	6	6	7
FD	a	11	11	11	11	12	9	9	9	11	9	9	9	**15**	12	12	13
	b	2	4	4	4	6	12	12	12	8	12	12	12	12	**15**	**15**	**15**
	c	1	2	2	2	4	8	8	8	6	8	8	8	10	13	13	10
	d	0	0	0	0	0	0	0	0	3	6	6	6	2	5	5	5
Blackstar HT-5 Metal																	
none	a	**15**	10	10	10	12	0	0	0	10	1	1	1	9	0	0	0
	b	11	**15**	**15**	**15**	9	5	5	6	8	7	7	7	5	6	6	6
	c	10	11	11	11	6	3	3	3	6	6	6	6	4	4	4	5
	d	13	14	14	14	10	4	4	4	9	5	5	5	7	5	5	4
AW	a	6	5	5	5	14	8	8	9	4	2	2	2	8	2	2	2
	b	0	0	0	0	1	6	6	5	0	0	0	0	0	1	1	1
	c	3	3	3	3	11	13	13	14	1	4	4	4	2	7	7	7
	d	2	2	2	2	5	10	10	10	2	3	3	3	1	3	3	3
HP	a	14	12	12	12	13	14	14	13	**15**	12	12	12	14	12	12	11
	b	8	9	9	9	7	**15**	**15**	**15**	12	**15**	**15**	**15**	12	**15**	**15**	**15**
	c	9	8	8	8	8	11	11	11	13	14	14	14	13	14	14	13
	d	7	7	7	7	4	9	9	8	11	13	13	13	11	11	11	12
FD	a	12	13	13	13	**15**	12	12	12	14	10	10	10	**15**	10	10	10
	b	1	1	1	1	0	2	2	1	3	8	8	8	3	9	9	8
	c	5	6	6	6	3	7	7	7	7	11	11	11	10	13	13	14
	d	4	4	4	4	2	1	1	2	5	9	9	9	6	8	8	9

Abbrev. a = MAE loss, b = MSE loss, c = ESR loss, d = ESR+DC loss
15 is the best, 0 is the worst achieved validation loss in each column

Table 4.1: Relative validation error scores of the LSTM-64 models trained with different loss function and pre-emphasis combinations.

in 10 out of 16 metrics on the Blackstar HT-5 Metal dataset, whereas it correlated with the validation pre-emphasis metrics in the case of Mesa Boogie 5:50+.

The plots of total and average scores in Figure 4.1 further confirm that the combination of MSE loss function and first-order high-pass filter should be the best choice for learning the LSTM network. Surprisingly, the model trained with ESR+DC loss function and A-weighting pre-emphasis filtering, which were the choices of Wright and others in [10], got the lowest score of all combinations in these experiments.

4.2 Objective Evaluation of the Models

Two WaveNet and three LSTM neural networks were chosen for the final comparison according to the criteria outlined in section 3.5. To measure the models' accuracy, ESR losses were calculated on the test sets without pre-emphasis. Results of the Mesa Boogie 5:50+ and Blackstar HT-5 Metal models are shown in Table 4.2.

In terms of accuracy, there is no clear winner when comparing the LSTM and WaveNet architectures; the Mesa amplifier model was better learned by the LSTM network, the WaveNet excelled in learning the HT-5 Metal amplifier. However, the LSTM network took only around 1 h 20 min to 1 h 30 min to learn, whereas the training time of a sufficiently large WaveNet was 2 or more hours. It was also once again confirmed that it is preferred to use the MSE loss and first-order high-pass filter pre-emphasis for training the LSTM network.

Network Type	Training Configuration	ESR Test Loss (%)	
		Mesa 5:50+	HT-5 Metal
WaveNet-18	ESR loss, HP filter	4.44	1.72
WaveNet-24	ESR loss, HP filter	3.04	**1.23**
LSTM-64	MSE loss, HP filter	2.54	2.95
LSTM-96	MSE loss, HP filter	**2.19**	2.46
LSTM-96	ESR+DC loss, AW filter	2.47	2.52
WaveNet-18	ESR loss, HP filter	0.29*	0.32*
LSTM-96	ESR+DC loss, AW filter	0.20*	1.80*

* Results achieved by Wright et al. (2020) [10] on a different test set

Table 4.2: ESR loss achieved by the compared models on the test sets of Mesa 5:50+ and Blackstar HT-5 Metal overdrive effects.

Additionally, the networks have tried to learn the Tremolo and Delay effects, which did not end up being successful. These results are presented in Table 4.3. The wanted effects were inaudible in produced recordings,

the models only introduced 'clicks' and other sound impairments. It was meaningless to have these recordings rated in the advanced listening tests.

A probable explanation for this failure is that these effects are very time-dependent – the Tremolo effect modulates volume periodically, the Delay effect always repeats sound after a set time. When pre-processing was performed, the recordings were split into 100 ms samples and shuffled in each training epoch, which completely broke the time context. This is not an issue with overdrive effects where the effect response time is almost negligible.

Network Type	Training Configuration	ESR Test Loss (%)	
		Delay	Tremolo
WaveNet-18	ESR loss, HP filter	37.17	12.44
WaveNet-24	ESR loss, HP filter	36.83	12.13
LSTM-96	MSE loss, HP filter	**25.06**	**9.21**
LSTM-96	ESR+DC loss, AW filter	25.63	9.98

Table 4.3: ESR loss achieved by the compared models on the test sets of Tremolo and Delay effects.

4.3 Subjective Results

Some of the trained models were compared in 6 sets of listening tests, 3 for each of the overdrive effects, by 9 assessors aged between 16 and 29 years. All assessors declared that they consider their hearing to be normal without any impairments. Five of them (56 %) stated that they have experience with playing a musical instrument, which could be a precondition to being more sensitive to imperfect sounds. An average MUSHRA score of the hidden references given by musicians was 98.9, whereas non-musicians rated them on average with 87 points, but the the population sample is too small and the test environment was not controlled, so the causation cannot be proved.

The MUSHRA Recommendation [30] states that assessors should be excluded from the responses if they rate the hidden reference for more than 15 % of the test items with a score lower that 90, or if they give the low quality anchor more than 90 points for more that 15 % of the test items. This would, however, exclude most of the participants of this experiment – one of the assessors even graded one mid anchor sample with a score of 100, unfortunately it was not possible to analyse whether this happened due to a simple error or as a deliberate judgement. Thus, the post-screening step was skipped and no submissions were rejected, but the results should not be taken as conclusive.

The grades given by assessors are presented in box plots, with outliers shown as circles and the labelled average scores as diamonds. Results of

the first experiment in Figure 4.2 compare the quality of LSTM-64 networks trained with four different training loss functions. In Figure 4.3, a comparison is made between models trained with different pre-emphasis filters. Finally, the sound quality of WaveNet and LSTM models is compared in Figure 4.4. Almost all mean scores appear in the 'Excellent' range above 80 points, but the data sample is not convincing enough to be able to declare a decisive winner.

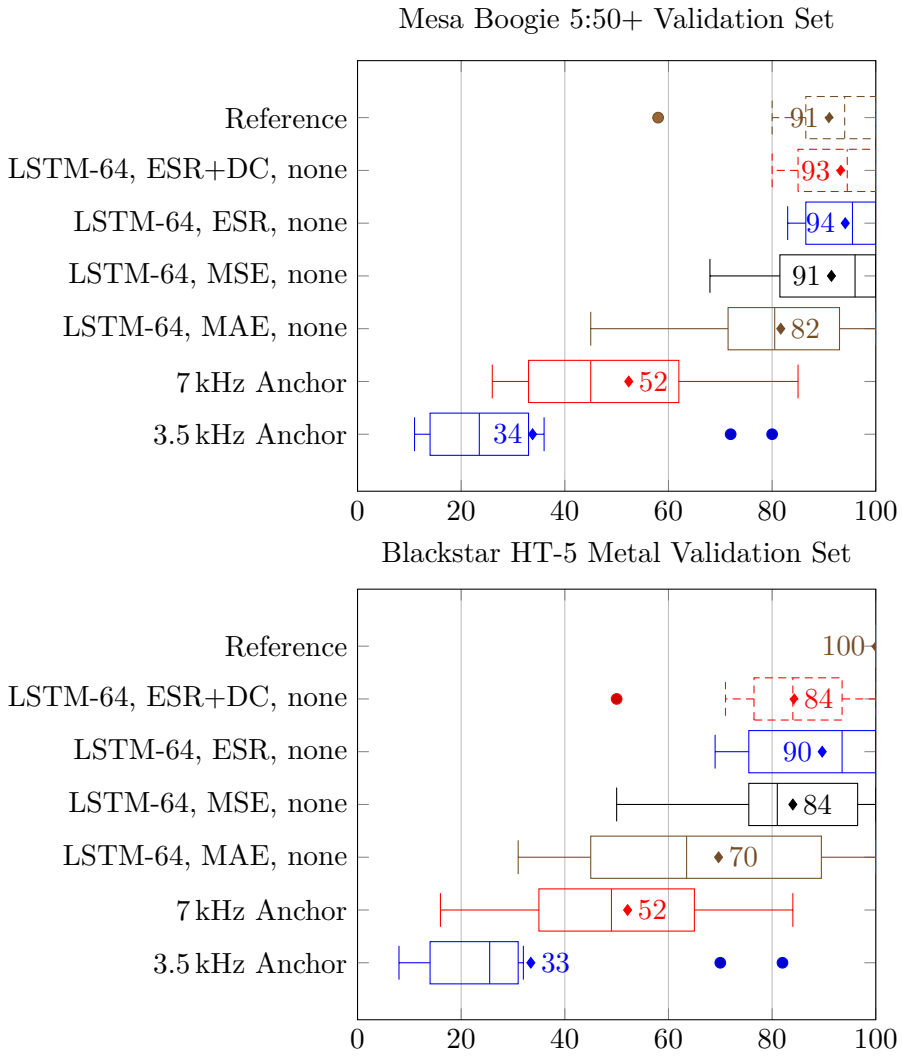

Figure 4.2: MUSHRA scores given in the listening test where training loss functions were compared.

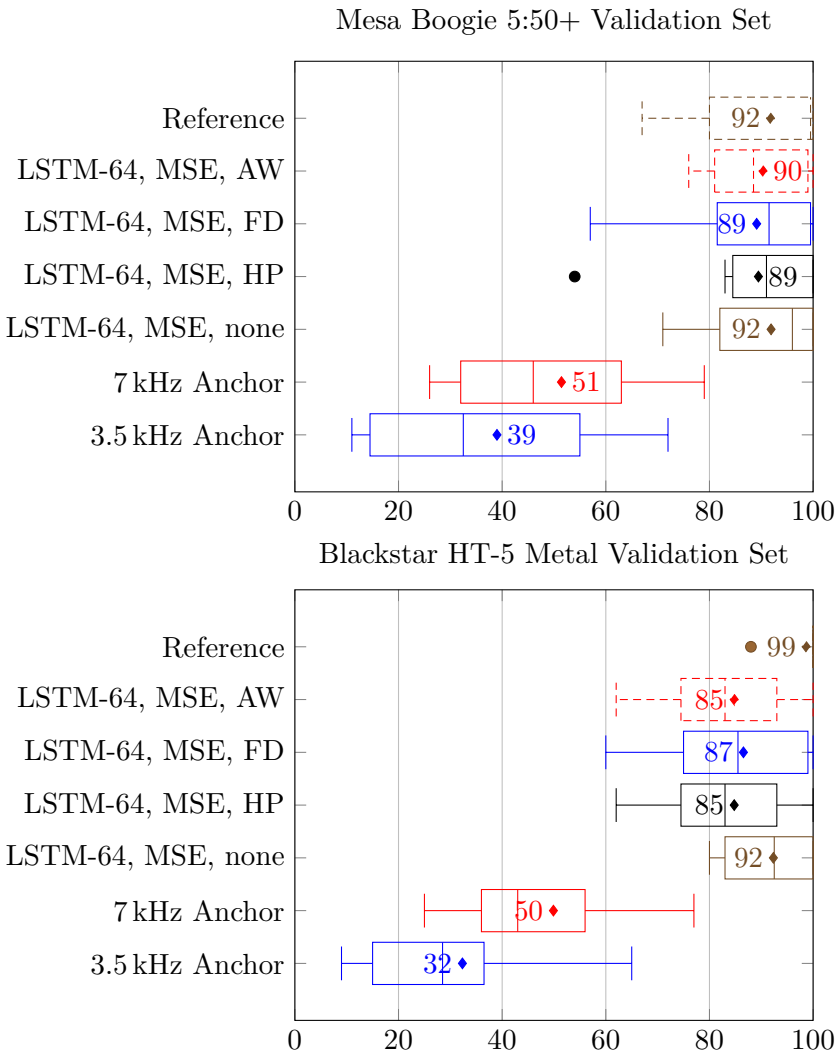

Figure 4.3: MUSHRA scores given in the listening test where training pre-emphasis functions were compared.

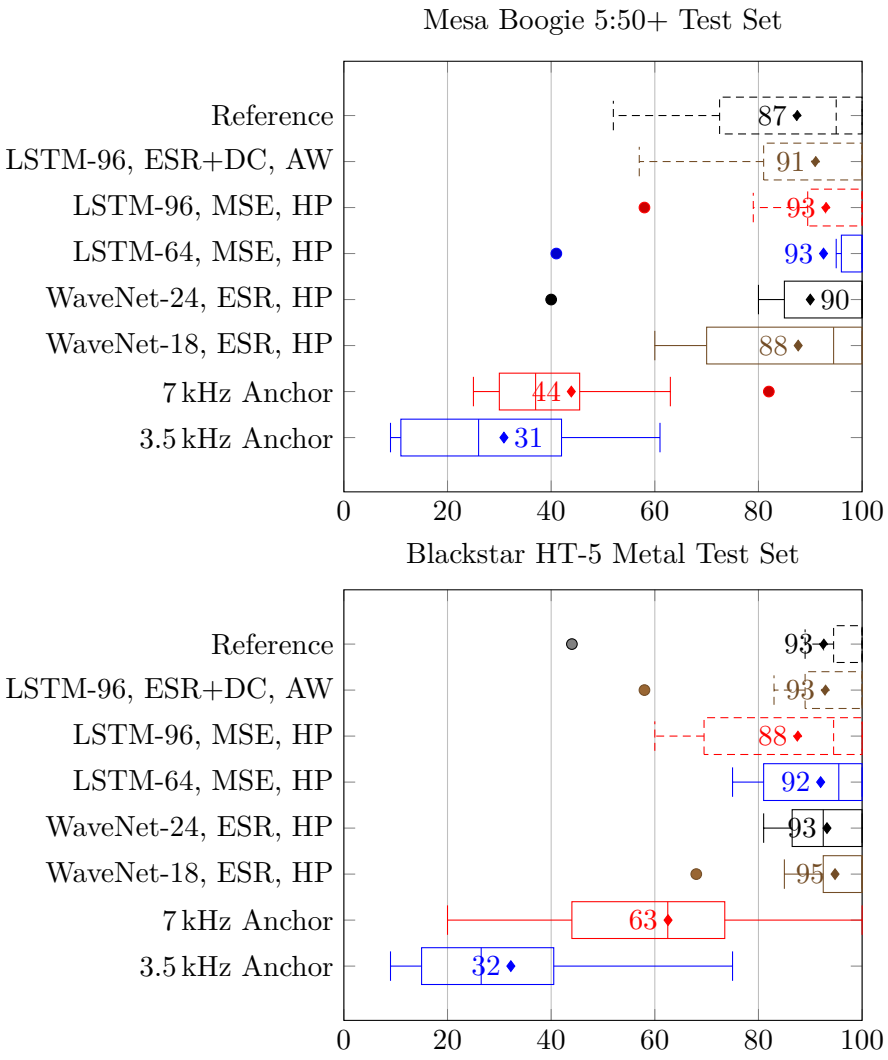

Figure 4.4: MUSHRA scores given in the listening test in the final model compare test.

Chapter 5

Conclusion

In conclusion, two neural network architectures were presented, which were able to learn two valve guitar amplifier models with high accuracy from short audio recordings. Several experiments were conducted to find the best learning hyperparameters, where it was shown that the correct combination of training loss function and pre-emphasis filtering ultimately lead to lower validation error. This was followed by a direct comparison between the WaveNet and the recurrent neural network.

WaveNet achieved lower test errors on the Blackstar HT-5 Metal amplifier model, whereas the Mesa 5:50+ amplifier was better learned by the LSTM network. In listening tests, assessors have stated that most of the presented models have an 'Excellent' audio quality when compared to the reference. There might not be a clean winner, but the LSTM network has the bonus of significantly faster training time. It was also tested whether the networks could learn other than overdrive effects. Unfortunately, this was not successful with the Tremolo and Delay effects, probably because the time context was lost during pre-processing when the sound sequences are split into batches and shuffled.

This thesis did not aim to create a real-time implementation, but it would undoubtedly be a major improvement. Both the WaveNet and the RNN already have existing real-time versions created by other researchers. In the future, it would be interesting to explore some other approaches to the effects emulation problem, for example with the Generative Adversial Networks (GANs). Finally, the produced audio still contains small imperfections, which might be fixed with some smart post-processing.

Appendix A

Listening Test Schedule

1. **Introduction**: Participant is shortly briefed with the goal of their participation and the rules.

2. **Volume settings:** In this step, participant can adjust the volume of test samples inside the web interface. It cannot be changed later.

3. **Training:** Participant gets familiar with the MUSHRA grading interface in a practical manner, no ratings are recorded from this step.

4. **Mesa 5:50+:** The following experiments follow in random order:

   a. MUSHRA – compare Mesa 5:50+ validation samples learned with:

Network architecture	Training loss	Training pre-emphasis
LSTM-64	MAE	none
LSTM-64	MSE	none
LSTM-64	ESR	none
LSTM-64	ESR+DC	none

   b. MUSHRA – compare Mesa 5:50+ validation samples learned with:

Network architecture	Training loss	Training pre-emphasis
LSTM-64	MSE	none
LSTM-64	MSE	A-weighting
LSTM-64	MSE	HP (α=0.85)
LSTM-64	MSE	FD (α=0.85)

   c. MUSHRA – compare Mesa 5:50+ test samples learned with:

Network architecture	Training loss	Training pre-emphasis
WaveNet-18	ESR	HP (α=0.95)
WaveNet-24	ESR	HP (α=0.95)
LSTM-64	MSE	HP (α=0.85)
LSTM-96	MSE	HP (α=0.85)
LSTM-96	ESR+DC	A-weighting

5. **HT-5 Metal:** The following experiments follow in random order:

   a. MUSHRA – compare HT-5 Metal validation samples learned with:

Network architecture	Training loss	Training pre-emphasis
LSTM-64	MAE	none
LSTM-64	MSE	none
LSTM-64	ESR	none
LSTM-64	ESR+DC	none

b. MUSHRA – compare HT-5 Metal validation samples learned with:

Network architecture	Training loss	Training pre-emphasis
LSTM-64	MSE	none
LSTM-64	MSE	A-weighting
LSTM-64	MSE	HP (α=0.85)
LSTM-64	MSE	FD (α=0.85)

c. MUSHRA – compare HT-5 Metal test samples learned with:

Network architecture	Training loss	Training pre-emphasis
WaveNet-18	ESR	HP (α=0.95)
WaveNet-24	ESR	HP (α=0.95)
LSTM-64	MSE	HP (α=0.85)
LSTM-96	MSE	HP (α=0.85)
LSTM-96	ESR+DC	A-weighting

6. **Questionnaire:** Participants answer the following questions:
   - Age
   - Do you consider your hearing to be normal (i.e. no hearing loss, no tinnitus)? *(Yes/No)*
   - Can you play any musical instrument? *(Yes/No)*
7. *End of test session*

Appendix B

Validation Data Tables of Training Combination Experiments

Training Loss		Validation Loss															
		none pre-emphasis				AW pre-emphasis				HP pre-emphasis				FD pre-emphasis			
		MAE	RMSE	ESR	ESR+DC	MAE	RMSE	ESR	ESR+DC	MAE	RMSE	ESR	ESR+DC	MAE	RMSE	ESR	ESR+DC
none	MAE	0.0217	0.0433	0.0184	0.0185	0.0145	0.0348	0.0157	0.0160	0.0112	0.0260	0.0423	0.0429	0.0147	0.0375	0.0335	0.0341
	MSE	**0.0205**	**0.0381**	**0.0143**	**0.0144**	0.0137	0.0301	0.0117	0.0119	0.0105	0.0225	0.0318	0.0319	0.0135	0.0321	0.0246	0.0248
	ESR	0.0227	0.0412	0.0167	0.0167	0.0152	0.0331	0.0142	0.0142	0.0108	0.0243	0.0370	0.0370	0.0149	0.0355	0.0301	0.0302
	ESR+DC	0.0245	0.0435	0.0186	0.0187	0.0170	0.0351	0.0159	0.0160	0.0127	0.0269	0.0451	0.0456	0.0167	0.0383	0.0350	0.0350
AW	MAE	0.0256	0.0445	0.0195	0.0195	**0.0121**	0.0315	0.0129	0.0129	0.0126	0.0272	0.0461	0.0465	0.0137	0.0350	0.0292	0.0293
	MSE	0.0274	0.0442	0.0192	0.0192	0.0133	**0.0298**	**0.0115**	0.0117	0.0130	0.0256	0.0409	0.0410	0.0144	0.0328	0.0257	0.0259
	ESR	0.0271	0.0438	0.0189	0.0190	0.0129	0.0298	0.0115	**0.0116**	0.0129	0.0260	0.0424	0.0426	0.0146	0.0333	0.0265	0.0269
	ESR+DC	0.0281	0.0470	0.0217	0.0218	0.0154	0.0329	0.0140	0.0140	0.0154	0.0308	0.0593	0.0593	0.0172	0.0383	0.0350	0.0353
HP	MAE	0.0235	0.0424	0.0177	0.0179	0.0145	0.0329	0.0141	0.0142	**0.0087**	0.0217	0.0295	0.0297	0.0126	0.0328	0.0257	0.0259
	MSE	0.0266	0.0436	0.0187	0.0187	0.0157	0.0316	0.0130	0.0130	0.0092	**0.0201**	**0.0253**	**0.0253**	0.0130	0.0304	0.0220	0.0221
	ESR	0.0270	0.0450	0.0199	0.0199	0.0167	0.0332	0.0143	0.0144	0.0099	0.0215	0.0289	0.0292	0.0140	0.0322	0.0248	0.0248
	ESR+DC	0.0269	0.0456	0.0205	0.0205	0.0171	0.0344	0.0153	0.0153	0.0101	0.0223	0.0311	0.0312	0.0144	0.0335	0.0268	0.0268
FD	MAE	0.0240	0.0434	0.0185	0.0185	0.0135	0.0320	0.0133	0.0134	0.0102	0.0226	0.0319	0.0320	**0.0118**	0.0319	0.0242	0.0242
	MSE	0.0283	0.0453	0.0202	0.0203	0.0156	0.0311	0.0125	0.0125	0.0112	0.0220	0.0302	0.0304	0.0132	**0.0302**	**0.0217**	**0.0219**
	ESR	0.0287	0.0457	0.0206	0.0206	0.0162	0.0324	0.0136	0.0139	0.0117	0.0231	0.0334	0.0344	0.0137	0.0317	0.0240	0.0249
	ESR+DC	0.0302	0.0485	0.0231	0.0233	0.0184	0.0356	0.0164	0.0165	0.0128	0.0250	0.0391	0.0392	0.0153	0.0346	0.0286	0.0287

Table B.1: Validation errors of all LSTM-64 Mesa 5:50+ training combinations

Training Loss		Validation loss															
		none pre-emphasis				AW pre-emphasis				HP pre-emphasis				FD pre-emphasis			
		MAE	RMSE	ESR	ESR+DC	MAE	RMSE	ESR	ESR+DC	MAE	RMSE	ESR	ESR+DC	MAE	RMSE	SER	ESD+DC
none	MAE	**0.0200**	0.0433	0.0162	0.0163	0.0121	0.0299	0.0117	0.0119	0.0073	0.0191	0.0355	0.0357	0.0110	0.0290	0.0283	0.0286
	MSE	0.0210	**0.0419**	**0.0152**	**0.0152**	0.0127	0.0283	0.0105	0.0106	0.0076	0.0166	0.0267	0.0267	0.0115	0.0261	0.0230	0.0231
	ESR	0.0217	0.0428	0.0159	0.0160	0.0130	0.0289	0.0110	0.0110	0.0077	0.0167	0.0270	0.0271	0.0117	0.0264	0.0235	0.0235
	ESR+DC	0.0210	0.0423	0.0155	0.0155	0.0126	0.0286	0.0107	0.0107	0.0073	0.0168	0.0273	0.0274	0.0112	0.0263	0.0233	0.0235
AW	MAE	0.0290	0.0493	0.0211	0.0211	0.0112	0.0280	0.0103	0.0103	0.0085	0.0189	0.0345	0.0346	0.0112	0.0267	0.0241	0.0243
	MSE	0.0462	0.0663	0.0381	0.0382	0.0149	0.0283	0.0105	0.0107	0.0114	0.0200	0.0387	0.0389	0.0150	0.0275	0.0256	0.0258
	ESR	0.0333	0.0525	0.0239	0.0239	0.0125	0.0272	0.0097	0.0097	0.0096	0.0184	0.0329	0.0331	0.0124	0.0257	0.0223	0.0223
	ESR+DC	0.0333	0.0530	0.0244	0.0245	0.0131	0.0277	0.0101	0.0102	0.0093	0.0187	0.0338	0.0339	0.0126	0.0264	0.0236	0.0237
HP	MAE	0.0207	0.0424	0.0156	0.0156	0.0116	0.0271	0.0097	0.0098	**0.0062**	0.0151	0.0219	0.0220	0.0092	0.0235	0.0186	0.0187
	MSE	0.0244	0.0440	0.0168	0.0170	0.0129	**0.0267**	**0.0093**	**0.0095**	0.0067	**0.0140**	**0.0189**	**0.0190**	0.0098	**0.0220**	**0.0163**	**0.0163**
	ESR	0.0240	0.0443	0.0170	0.0171	0.0128	0.0275	0.0099	0.0101	0.0066	0.0144	0.0202	0.0208	0.0096	0.0227	0.0174	0.0181
	ESR+DC	0.0261	0.0457	0.0181	0.0182	0.0138	0.0280	0.0103	0.0103	0.0072	0.0149	0.0215	0.0216	0.0105	0.0235	0.0186	0.0186
FD	MAE	0.0210	0.0423	0.0155	0.0155	**0.0111**	0.0273	0.0098	0.0098	0.0062	0.0154	0.0230	0.0230	**0.0089**	0.0236	0.0188	0.0189
	MSE	0.0401	0.0594	0.0306	0.0306	0.0154	0.0289	0.0110	0.0113	0.0089	0.0160	0.0249	0.0254	0.0118	0.0239	0.0192	0.0198
	ESR	0.0302	0.0490	0.0208	0.0209	0.0141	0.0281	0.0103	0.0105	0.0076	0.0151	0.0220	0.0222	0.0107	0.0230	0.0178	0.0181
	ESR+DC	0.0312	0.0507	0.0223	0.0223	0.0149	0.0290	0.0111	0.0113	0.0080	0.0158	0.0241	0.0244	0.0112	0.0240	0.0194	0.0196

Table B.2: Validation errors of all LSTM-64 HT-5 Metal training combinations

Bibliography

1. GUARNIERI, Massimo. The Age of Vacuum Tubes: The Conquest of Analog Communications [Historical]. *IEEE Industrial Electronics Magazine*. 2012, **6**(2), 52–54. Available from DOI: 10.1109/MIE.2012.2193274.
2. BARBOUR, Eric. The cool sound of tubes [vacuum tube musical applications]. *IEEE Spectrum*. 1998, **35**(8), 24–35. Available from DOI: 10.1109/6.708439.
3. Profiler Overview. *Kemper Amps* [online]. Kemper GmbH, ©2021 [visited on 2021-05-13]. Available from: https://www.kemper-amps.com/profiler/overview.
4. YEH, David T. et al. Numerical Methods for Simulation of Guitar Distortion Circuits. *Computer Music Journal*. 2008, **32**(2), 23–42. Available from DOI: 10.1162/comj.2008.32.2.23.
5. ZHANG, Zhichen et al. A Vacuum-Tube Guitar Amplifier Model Using Long/Short-Term Memory Networks. In: *SoutheastCon 2018*. IEEE, 2018, pp. 534–538. Available from DOI: 10.1109/SECON.2018.8479039.
6. SCHMITZ, Thomas; EMBRECHTS, Jean-Jacques. Real Time Emulation of Parametric Guitar Tubes Amplifier With Long Short Term Memory Neural Network. In: *5th International Conference on Signal Processing (CSIP 2018)*. 2018. Available from DOI: 10.5121/csit.2018.80511.
7. WRIGHT, Alec; DAMSKÄGG, Eero-Pekka; VÄLIMÄKI, Vesa. Real-Time Black-Box Modelling with Recurrent Neural Networks. In: *Proceedings of the 22nd International Conference on Digital Audio Effects (DAFx-19)*. 2019, pp. 173–180. Available also from: https://dafx.de/paper-archive/2019/DAFx2019_paper_43.pdf.
8. DAMSKÄGG, Eero-Pekka et al. Deep Learning for Tube Amplifier Emulation. In: *Proceedings of the 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 471–475. Available from DOI: 10.1109/ICASSP.2019.8682805.
9. DAMSKÄGG, Eero-Pekka; JUVELA, Lauri; VÄLIMÄKI, Vesa. Real-Time Modeling of Audio Distortion Circuits with Deep Learning. In: *Proceedings of the 16th Sound & Music Computing Conference SMC*

2019. 2019, pp. 332–339. Proceedings of the Sound and Music Computing Conferences. Available also from: http://smc2019.uma.es/.

10. WRIGHT, Alec et al. Real-Time Guitar Amplifier Emulation with Deep Learning. *Applied Sciences*. 2020, **10**(3), 766. Available from DOI: 10.3390/app10030766.

11. WRIGHT, Alec; VÄLIMÄKI, Vesa. Perceptual Loss Function for Neural Modelling of Audio Systems. In: *Proceedings of the 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 251–255. Available from DOI: 10.1109/ICASSP40776.2020.9052944.

12. Plugins. *Neural DSP* [online]. [N.d.] [visited on 2021-05-09]. Available from: https://neuraldsp.com/plugins.

13. BLOEMER, Keith. *GuitarML* [online]. [2020] [visited on 2021-05-09]. Available from: https://guitarml.com.

14. HOCHREITER, Sepp; SCHMIDHUBER, Jürgen. Long short-term memory. *Neural computation*. 1997, **9**(8), 1735–1780. Available from DOI: 10.1162/neco.1997.9.8.1735.

15. OLAH, Christopher. Understanding LSTM Networks. *colah's blog* [online]. 2015 [visited on 2021-08-01]. Available from: https://colah.github.io/posts/2015-08-Understanding-LSTMs.

16. GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. *Deep Learning* [online]. MIT Press, 2016 [visited on 2021-04-19]. Available from: http://www.deeplearningbook.org.

17. Conv1d. *PyTorch 1.8.1 documentation* [online]. Torch Contributors, ©2019 [visited on 2021-04-20]. Available from: https://pytorch.org/docs/1.8.1/generated/torch.nn.Conv1d.

18. tf.nn.convolution. *TensorFlow Core v2.4.0 API Documentation* [online]. 2021 [visited on 2021-04-20]. Available from: https://www.tensorflow.org/versions/r2.4/api_docs/python/tf/nn/convolution.

19. VAN DEN OORD, Aäron et al. *WaveNet: A Generative Model for Raw Audio* [online]. 2016 [visited on 2021-05-09]. Available from arXiv: 1609.03499 [cs.SD].

20. SCHMITZ, Thomas; EMBRECHTS, Jean Jacques. Introducing a dataset of guitar amplifier sounds for nonlinear emulation benchmarking. In: *Proceedings of the AES 145th Convention* [online]. Audio Engineering Society, 2018 [visited on 2021-02-22]. Available from HDL: 2268/228910. Dataset available from: https://services.montefiore.uliege.be/acous/STSI/downloads.php.

21. STEIN, Michael et al. Automatic Detection of Audio Effects in Guitar and Bass Recordings. In: *Proceedings of the AES 128th Convention*. Audio Engineering Society, 2010. Dataset available from: https://www.idmt.fraunhofer.de/en/business_units/m2d/smt/audio_effects.html.

22. MCFEE, Brian et al. *librosa/librosa: 0.8.0* [online]. Zenodo, 2020 [visited on 2021-04-20]. Available from DOI: 10.5281/zenodo.3955228.

23. COMUNITÀ, Marco; STOWELL, Dan; REISS, Joshua D. *Guitar Effects Recognition and Parameter Estimation with Convolutional Neural Networks*. 2020. Available from arXiv: 2012.03216 [cs.SD].

24. STEINMETZ, Christian J.; REISS, Joshua D. auraloss: Audio focused loss functions in PyTorch. In: *Digital Music Research Network One-day Workshop (DMRN+15)* [online]. 2020 [visited on 2021-08-10]. Available from: https://www.qmul.ac.uk/dmrn/media/dmrn/DMRN-15_proceedings.pdf.

25. VERGIN, R.; O'SHAUGHNESSY, D. Pre-emphasis and speech recognition. In: *Proceedings 1995 Canadian Conference on Electrical and Computer Engineering*. 1995, vol. 2, pp. 1062–1065. Available from DOI: 10.1109/CCECE.1995.526613.

26. PASZKE, Adam; GROSS, Sam; MASSA, Francisco, et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: *Advances in Neural Information Processing Systems 32* [online]. Curran Associates, Inc., 2019, pp. 8024–8035 [visited on 2021-08-01]. Available from: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

27. KINGMA, Diederik P.; BA, Jimmy. *Adam: A Method for Stochastic Optimization*. 2017. Available from arXiv: 1412.6980 [cs.LG].

28. KOKER, Teddy et al. *PedalNet* [online]. GitHub, 2020 [visited on 2021-08-01]. Available from: https://github.com/teddykoker/pedalnet.

29. FALCON, William et al. *PyTorch Lightning* [online]. GitHub, 2019 [visited on 2021-05-09]. Available from: https://github.com/PyTorchLightning/pytorch-lightning.

30. ITU-R BS.1534-3. *Method for the subjective assessment of intermediate quality level of audio systems* [online]. 2015 [visited on 2021-08-11]. Recommendation. International Telecommunication Union. Available from: https://www.itu.int/rec/R-REC-BS.1534-3-201510-I/en.

31. SCHOEFFLER, Michael et al. webMUSHRA — A Comprehensive Framework for Web-based Listening Tests. *Journal of Open Research Software*. 2018, **6**(1), 8. Available from DOI: http://doi.org/10.5334/jors.187.