University of Southampton
Faculty of Engineering and Physical Sciences
Electronics and Computer Science

Neural network based audio effect emulation

Brijee Rana

September 2022

Supervisor: Dr Jagmohan Chauhan
Second Examiner: Dr Nicholas M Gibbins

A dissertation submitted in partial fulfilment of
the degree of MSc Data Science

Abstract

In this dissertation, I will investigate whether we can replicate audio effects using neural networks. I will discuss the state-of-the-art in terms of what is being researched as well as implement recurrent neural network architectures for the emulation of a non-linear analog sound effect called **fuzz** and a modulation/time-varying sound effect called **chorus**. This types of audio effects are used by a variety of professionals and musicians to achieve a desired sound. Digital models are not as good as the analog devices and thus a demand for better models have always been around. I will attempt to use end-to-end black box modelling as an approach that is currently being investigated by digital audio researchers. There are different types of recurrent neural networks, I will mostly be using LSTMs and GRUs to train an architecture and then apply that model to a new input such as a guitar recording. In my research this approach turns out to be effective in modelling non-linear audio effects like fuzz but perform poorly for time-varying effects like chorus. LSTMs perform slightly better than GRUs but take longer to train the model. I believe that this approach of neural network based audio effect emulation can successful even more in the future as it shows promising results now.

0.1 Acknowledgements

My father has supported me throughout my whole tempestuous tenure in this masters degree. His encouragement and positively has allowed me to ”ride the storm”.

I would like to thank my supervisor Dr Jagmohan Chauhan for his patience and advice, the Data Science department, and the University itself.

Statement of Originality

- I have read and understood the ECS Academic Integrity information and the University's Academic Integrity Guidance for Students.
- I am aware that failure to act in accordance with the Regulations Governing Academic Integrity may lead to the imposition of penalties which, for the most serious cases, may include termination of programme.
- I consent to the University copying and distributing any or all of my work in any form and using third parties (who may be based outside the EU/EEA) to verify whether my work contains plagiarised material, and for quality assurance purposes.

You must change the statements in the boxes if you do not agree with them.

We expect you to acknowledge all sources of information (e.g. ideas, algorithms, data) using citations. You must also put quotation marks around any sections of text that you have copied without paraphrasing. If any figures or tables have been taken or modified from another source, you must explain this in the caption and cite the original source.

I have acknowledged all sources, and identified any content taken from elsewhere.

If you have used any code (e.g. open-source code), reference designs, or similar resources that have been produced by anyone else, you must list them in the box below. In the report, you must explain what was used and how it relates to the work you have done.

I have not used any resources produced by anyone else.

You can consult with module teaching staff/demonstrators, but you should not show anyone else your work (this includes uploading your work to publicly-accessible repositories e.g. Github, unless expressly permitted by the module leader), or help them to do theirs. For individual assignments, we expect you to work on your own. For group assignments, we expect that you work only with your allocated group. You must get permission in writing from the module teaching staff before you seek outside assistance, e.g. a proofreading service, and declare it here.

I did all the work myself, or with my allocated group, and have not helped anyone else.

We expect that you have not fabricated, modified or distorted any data, evidence, references, experimental results, or other material used or presented in the report. You must clearly describe your experiments and how the results were obtained, and include all data, source code and/or designs (either in the report, or submitted as a separate file) so that your results could be reproduced.

The material in the report is genuine, and I have included all my data/code/designs.

We expect that you have not previously submitted any part of this work for another assessment. You must get permission in writing from the module teaching staff before re-using any of your previously submitted work for this assessment.

I have not submitted any part of this work for another assessment.

If your work involved research/studies (including surveys) on human participants, their cells or data, or on animals, you must have been granted ethical approval before the work was carried out, and any experiments must have followed these requirements. You must give details of this in the report, and list the ethical approval reference number(s) in the box below.

My work did not involve human participants, their cells or data, or animals.

ECS Statement of Originality Template, updated August 2018, Alex Weddell aiofficer@ecs.soton.ac.uk

Contents

0.1 Acknowledgements 1

1 Introduction **5**
1.1 Background and motivation 6
1.2 Problem statement 7
1.3 Research aim 8
1.4 Report structure 8

2 Literature review **9**
2.1 Guitar Capture Technology 9
2.2 Audio effects 10
2.2.1 Fuzz 10
2.2.2 Chorus 10
2.3 Neural networks 10
2.3.1 Convolutional neural networks 11
2.3.2 Autoencoders 11
2.4 Recurrent neural networks 11
2.4.1 LSTM 12
2.4.2 GRU 12

3 Methodology **13**
3.1 Dataset 13
3.2 Data pre-processing 13
3.3 Network architecture 14
3.3.1 Models and parameters 15
3.4 Implementation 15

4 Results and analysis **17**
4.1 Audio effect evaluation 17
4.2 Model evaluation 20
4.3 Discussion 21

5 Conclusion **23**
5.1 Future work 23

List of Figures

1.1 Jimi hendrix playing at woodstock, source: www.guitargearfinder.com 6
1.2 My personal multi-fx pedal 7

2.1 Guitar amplifier plug-in, source:www.musicianonamission.com . . 9
2.2 Standard RNN layer, source:http://colah.github.io/ 11
2.3 LSTM layer, source:http://colah.github.io/ 12

3.1 Network architecture 14

4.1 Graph plots of Model 3 on Fuzz 18
4.2 Graph plots of Model 1 on Chorus 19

Chapter 1

Introduction

Ever since the electric guitar was invented in 1936, amplifiers and audio effect pedals have been created and used by countless people to make music. As the development of technology increases so has the number of ways we can now use such devices, especially on the computer as digital audio processing allows for these physical musical items to be modelled. However, contemporary amp modeling and audio effect plug in software has limitations in copying its real-life counter parts as the analog hardware devices have complex nuances including some being non-linear, time-varying, dynamic which prove very complicated and tedious. Most techniques involve modelling the circuitry or applying complex algorithms for the desired audio emulation to be achieved. Due to the recent advances in machine learning, recent papers have explored signal processing applications using neural networks [1]. My initial aim for this project was to emulate the sound of guitar tube amplifiers yet whilst researching, found that it has been investigated by a number of researchers recently [2] [3] since when someone first owns an electric guitar, they must get an amplifier to plug into and play. Therefore, I turned my attention to replicating audio effects to explore slightly uncharted areas. I really enjoy chorus effects which mimic the sound of a choir by doubling the original sound and putting it slightly out of tune as well as a highly distorted fuzz audio effect thanks to Jimi Hendrix.

1.1 Background and motivation

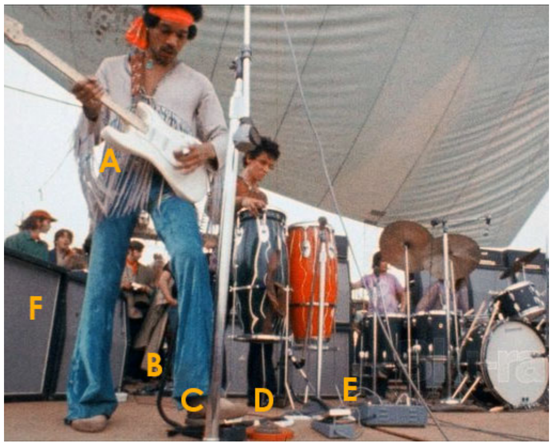


Figure 1.1: Jimi hendrix playing at woodstock, source: www.guitargearfinder.com

The above image show Jimi Hendrix using analog audio effects pedals which are analog boxes with circuitry inside in which you can plug your cable through your guitar signal and then into the amplifier to provide the desired tones and sounds. Nowadays, thanks to the advancements of technology and computers, many music producers and guitar players are moving towards digital audio effect software called plug-ins which ”plug” into you digital audio workstation (DAW). Electric guitars can be connected to an audio interface which transfers the guitar signal into a digital signal to the computer. This can then be manipulated by the DAW which, with plug-ins can emulate amplifiers, audio effects and essentially be a virtual recording studio. However, some of the current digital audio effects are lackluster compared to analog audio effects as they are often poorly modelled or optimised. Due to recent developments in deep learning, there has been some research in the digital audio field but mostly within audio classification and generation. Papers from interspeech 2021/22 about removing acoustic echo [4], improving voice separation [5] and reverb conversion [6] as my initial reading into what topic I wanted to get into has pointed me towards the field of audio, signal processing, deep learning and thanks to my passion in guitars, music. Applying an end-to-end black box neural network model would be interesting as I do not need to use algorithms explicitly, replicate circuitry or have deep knowledge in signal processing. I believe machine learning can be utilized for these audio tasks and it will be the future going forward in music production.

Figure 1.2: My personal multi-fx pedal

1.2 Problem statement

Analog audio devices can be steeply expensive, difficult to procure and fragile. An example are vacuum tubes for high-end guitar amplifiers which are expensive, made of glass, require frequent up-keep and changes and are now scarce in quality as the conflict in Ukraine has halted the production of the vacuum tubes as they are produced from only 1 factory in this country. Digital emulation is a cheaper alternative which also provides a solution to the other problems of analog audio devices. Contemporary digital methods can emulate these sounds to a certain degree of quality, yet, they are tedious to model, require bespoke modelling for each sound and fail to capture the idiosyncrasies and non-linearity of certain sounds as various parameters such as volume are altered. Thus investigating the state-of-the-art in terms of audio signal processing and neural networks can further improve on these modern digital audio software.

1.3 Research aim

The aim of this project it to apply a neural network based audio effect emulator.

Objectives:

1. Familiarisation with digital signal processing and the current techniques used.
2. Understanding how amp modelling and digital audio effects work.
3. Investigating AI field within audio signal processing.
4. Finding suitable neural network architecture.
5. Implement and optimise the model.
6. Evaluate and compare with modern techniques used now.

1.4 Report structure

This first chapter is just the introduction to the project. The next chapter will explain the current research in my area of research such as what deep learning models are being explored. The following chapter will explain the methodology of the project including what dataset I am using, architecture implemented. Then finally I will discuss and evaluate my findings and conclude.

Chapter 2

Literature review

This chapter discusses the current models used as well as the research happening in the field of audio emulation using neural networks.

2.1 Guitar Capture Technology

Technology to try and capture the sound profiling of analog devices have been attempted before but not with deep learning. Guitar amplifiers which are typically first to replicate, has been modelled by digital signal processing hardware. Tube amplifiers contain a plethora of non linear components which make it difficult to model the circuitry of. Some devices which are popular among guitarists are the Kemper Profiler and Mooer Tone Capture GTR. In fact, some people make money buy selling their pro-filings of their desired amplifiers especially rare and out of production amps. These hardware devices have their information about how they profile amplifiers heavily guarded and hardware filled with security features.

Figure 2.1: Guitar amplifier plug-in, source:www.musicianonamission.com

Most of these methods try to find a work-around with the non-linearity of audio transformations. These methods are called white box models in which inner workings of the model are transparent and interpretive, these models are often used in modern audio effect software, an example of one is shown in figure 2.1. Some papers and methods that I have read into are based on control theory, Wiener-Hammerstein models [7], linear time-invariant (LTI) systems [8] and Volterra series [9].

My project will be based on end-to-end black box modelling which is the opposite as its based on deep learning to capture the non-linear transformations but is hard to gain the comprehensive inner working of the model.

2.2 Audio effects

Audio effects are essential to musicians and producers, as audio signals are manipulated to achieve desired sounds. It is used countless times in almost every piece of music played. Even sounds from an elevator has probably been transformed, in fact almost every piece of audio produced digitally generally involve some manipulation of signals. There are a plethora of audio effects which offer a wide variety of changes to the sound wave. Here I shall discuss the 2 audio effects that I will model using neural networks.

2.2.1 Fuzz

Fuzz is an aggressive distortion sound. Distorts audio waves non-linearly into square waves (actually more irregular quadrilateral), adding complex overtones as the volume is increased and waves are clipped. This effect is time invariant which means the output of the samples do not depend on the time.

2.2.2 Chorus

Chorus is a modulation or time-varying effect. It mimics a choir by duplicating the audio wave, playing it alongside the original audio wave at different pitches and speeds. Time-varying audio effects typically have a low frequency oscillator (LFO) that periodically changes parameters of the effect over time. Other notable effects with an LFO are flangers, phaser and wah wah.

2.3 Neural networks

Neural networks are branch of machine learning and AI. It mimics the way a human brain operates as it contains layers of artificial neurons to solve AI problems based on data. These networks can approximate complex functions without explicitly involving difficult maths. This is ideal when attempting to deal with the complexity of audio effects.

2.3.1 Convolutional neural networks

Convolutions neural networks are a type of neural networks in which a input has convolutions or mathematical operations applied to it. A modified version of Wavenet [10], with dilated convolutional layers can encapsulate the dynamic response of guitar amplifiers [11]. As I am trying to emulate time-varying effect in chorus, I do not think convolutional networks would be as effective as recurrent neural networks in remembering long term dependencies.

2.3.2 Autoencoders

During my research, this was a novel way in which I have seen audio effects being replicated [12]. Auto-encoders are a type of neural network that is trained to copy its input to its output. The proposed model that I have seen online contains an adaptive front end, synthesis back-end and latent-space DNN where the DNN is wavenet or bi-LSTM structure. These models are very large and complicated and any code examples or source code has been removed for the public due to a new license. I did try to replicate the paper but sadly failed.

2.4 Recurrent neural networks

Recurrent neural networks (RNNs) are similar to feedforward neural networks but contain feedback connections. This is useful for sequential data such discrete time series data like audio waves.

Traditional RNNs however, they are susceptible to vanishing or exploding gradient when doing backpropagation through time so it cannot manage long term dependencies as well as expected.

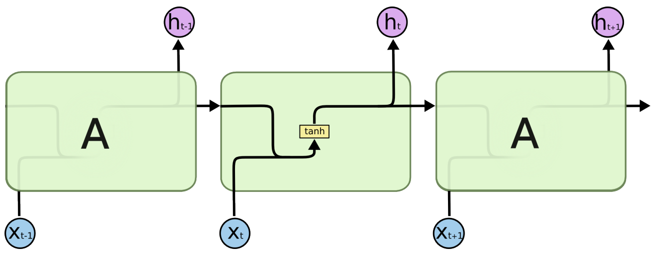


Figure 2.2: Standard RNN layer, source:http://colah.github.io/

2.4.1 LSTM

Long short term memory (LSTM) are introduced to solve this vanishing/exploding gradient issue of RNNs with its unique gated structure and ability to store memory in its cells, thus being able to learn long-term dependencies.

As you can see by the diagrams of figures 2.2 and 2.3, RNNs has a simple structure. LSTMs on the other hand, contain 4 layers instead of 1. These gates ”protect and control the cell state”.

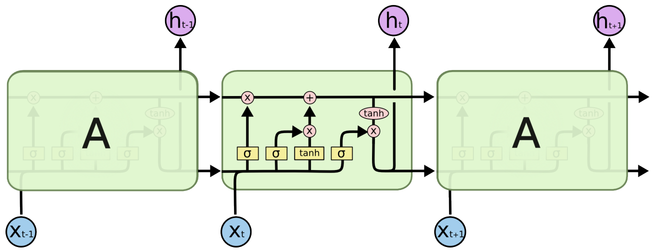


Figure 2.3: LSTM layer, source:http://colah.github.io/

2.4.2 GRU

Gated Recurrent Units are a simpler version of an LSTM, its without an output gate, so it fully writes the contents from its memory cell to the larger network at each time step. This also means that there is 1 less parameter to handle so it is quicker to learn or faster to train than LSTMs.

Chapter 3

Methodology

In this chapter, I will explain how my project was conducted.

3.1 Dataset

With machine learning, the performance of the model depends on the quality of the dataset. I am using the Fraunhofer Institute for Digital Media Technoloy, IDMT-SMT-Audio-Effects dataset which contains approximately 30 hours of audio. The dataset contains 55044 wav files with monophonic and polyphonic guitar notes and sounds meaning there are recordings of just 1 note being played and also several notes being played at the same time. 11 different audio effects are used such as: feedback, delay, slapback delay, reverb, chorus, flanger, phaser, tremolo, vibrato, distortion, overdrive no effect (unprocessed notes/sounds). Available online:

This dataset is neatly organised and labelled, however, for my specific task, it had issues that I had to solve which will be discussed in the next section. The audio files are recorded in mono (1 channel) which is beneficial as stereo would add another layer of complexity such that the network would have to process 2 channels of audio simultaneously (left and right ear). It is also 16 bit so processing and training times are faster and less computationally demanding yet maintains a respectable sample rate of 44.1 kHz which is of the quality of a CD.

3.2 Data pre-processing

This dataset contained 2 sec wav files of audio being played. As my network architecture would be end-to-end, I decided to concatenate them and then split them into test, train and validation files. Unfortunately, the audio effect files corresponding to the unprocessed audio files were disorganised and had 3 different versions. Therefore, I decided to apply the audio effects myself following the same procedure as they did. This was also beneficial as they did not have an audio effect for fuzz. I went on Ableton 10 (DAW) and applied warm fuzz effect and chorus, noting down their parameters for reproducibility. Before applying these effects however, I made sure to apply amplitude normalization meaning that each sound was being played at the same volume so that differences in sound does not affect the very sensitive model. This also makes it easier to plot graphs and compare audio files from target output with the predicted output of the model. The amplitude is between normalised between -1 and 1. Finally, I separate my audio files into train (15 minutes), test (5 minutes) and validate(4 minutes).

Audio effect parameters:

Warm Fuzz: Gain=50%, Output=-5.4 dB, Bass= 44%, Mid = -14%, Treble= -24%. Dry/Wet=67%.

Chorus: Delay 1=20Hz, 9.99ms, Delay 2=Mod, 14.6ms, Modulation= (Amount= 1ms, Rate= 1Hz), Polarity= negative, Feedback= 0.8%, Dry/Wet= 60%.

As you can see by these parameters, chorus is much more complicated than fuzz.

3.3 Network architecture

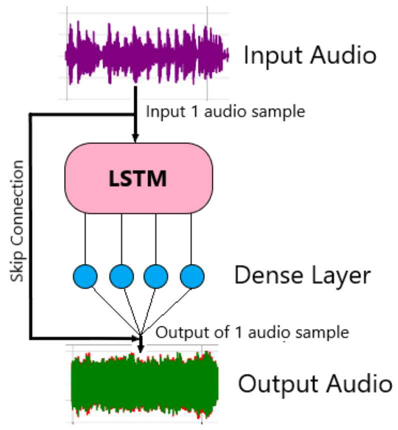


Figure 3.1: Network architecture

The recurrent neural network architecture I will be using is based on this network, as shown in figure 3.1, from a paper by the Aalto University Acoustics Lab in Findland which explored tube amplifier emulation.

Paper: Real-Time Guitar Amplifier Emulation with Deep Learning[11].

The neural network architecture contains an LSTM layer followed by dense layer, and uses skip connected so that network can learn the difference between input audio and target audio. I used this network to train my data, experiment with RNN structures and performance as well as plot graphs. The paper has code of the network architecture which I have used and is linked here . I have used the code only to train the network and borrow the network architecture structure.

3.3.1 Models and parameters

I will be mostly using and talking about these 3 models as shown:

Table 3.1: Models structure

	Model 1	Model 2	Model 3
hidden size	16	16	32
unit type	LSTM	GRU	LSTM
loss fcns	ESR:0.75,DC:0.25	ESR:0.75, DC:0.25	ESR:0.75, DC:0.25
pre filt	None	None	None

As you can see from the table above, model 1 and 3 are alike with LSTM units but model 3 has double the size of the hidden layer. Model 2 uses GRU units so that we can compare the difference with LSTM units in model 1. I have experimented with more variations but I chose these models to discuss the difference in parameters.

3.4 Implementation

I used google colab pro to run the code as my personal laptop would be too slow in training the network. The benefits of having colab pro is that you can access a higher end GPU and more RAM. I used the dataset that I have separated into train/test/val and used it to train the neural network architecture given from the paper. The dataset had 2 separate recording each of both unprocessed input audio and processed target audio. The network is trained along with an optimiser (Adam) and loss function (Error to Signal Ratio [ESR]). The network will try to minimise the loss between target and predicted audio/signal, finding the most optimum weights for the network to model the audio effects via back-propogation trough time. Error to signal ratio behaves similar to mean squared error loss function but calculates the difference between signals. The training dataset was split into half-second segments. This is so that training time is reduced as sequences can be processed in parallel and improves convergence rate by shuffling the dataset at each epoch. Early stopping is also included when training the network, stopping after 20 epochs if there was no improvements. Once training has finished, the network weights that had the lowest validation loss is saved as a .json file. This can then be loaded to apply the desired audio effect onto a new audio file.

That same network that produces the best validation loss, once training is finished, produces the best predicted validation prediction from it. I can then use this audio file to compare with the target audio based on the error to signal ratio. This gives me an opportunity to provide some quantitative data about the performance of the networks as well as a way to visualise the audio files. This is because it can be difficult judge the audio just by ear. Thus, I did not do any qualitative testing in terms of using humans to rate the performance of each model as using human data would also require a lot of more steps and ethical steps to go through.

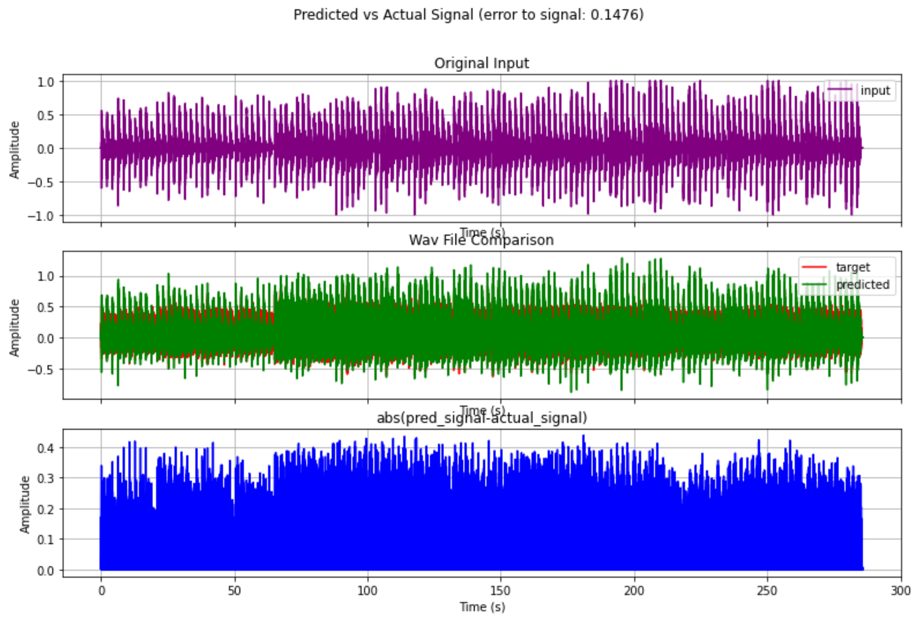
Chapter 4

Results and analysis

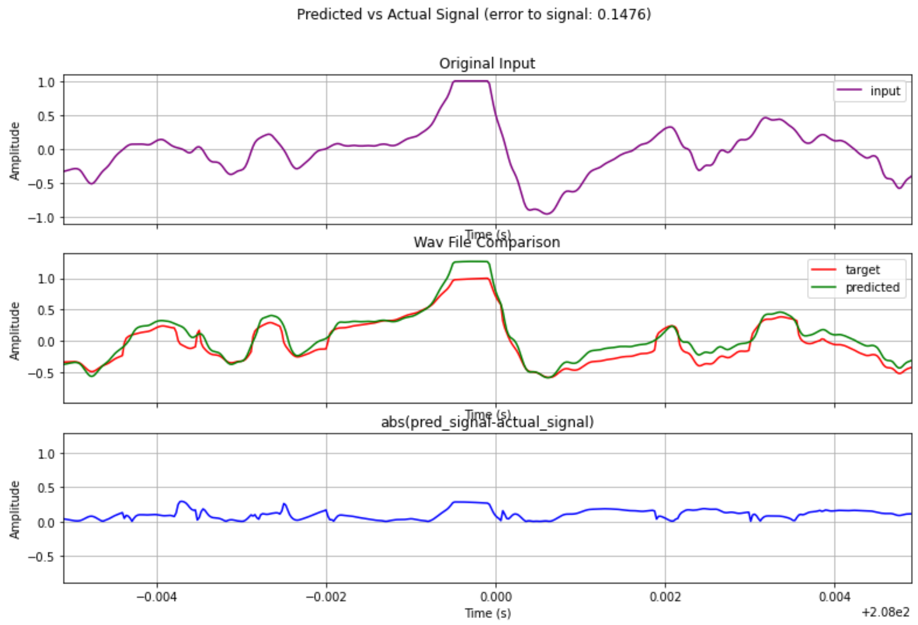
In this chapter, graph plots and the performance of the models are analysed and discussed.

4.1 Audio effect evaluation

Below are the graph plots of the model that best emulated the audio effects the best. For fuzz it was model 3 and for chorus model 1. I had some initial trouble with the graphs because I did not normalise the audio waves so it was difficult to compare them. This was also causing the network to poorly train as well so I had re-normalise all the data. As you can see by the figures below, the amplitudes are between -1 and 1. The first set of graphs(3) show a portion of the audio file where amplitude peaks can be easily seen and compared between target and predicted. The difference of the amplitude as well, in the bottom graph of the set. The top graph shows the input data. The second bundle of graphs show a finer audio file in which it is easier to interpret how close predicted is to target audio.

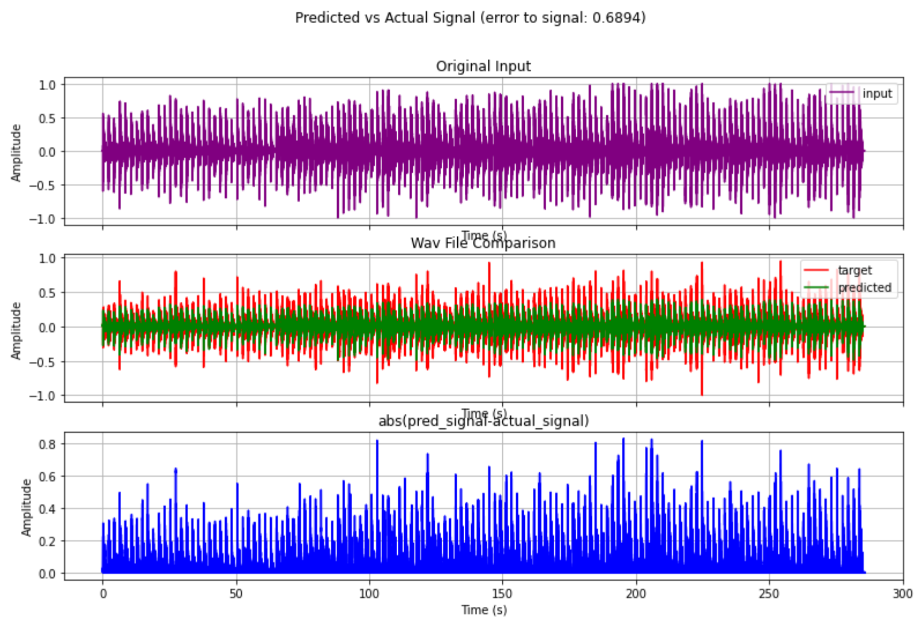


(a) Signal comparison

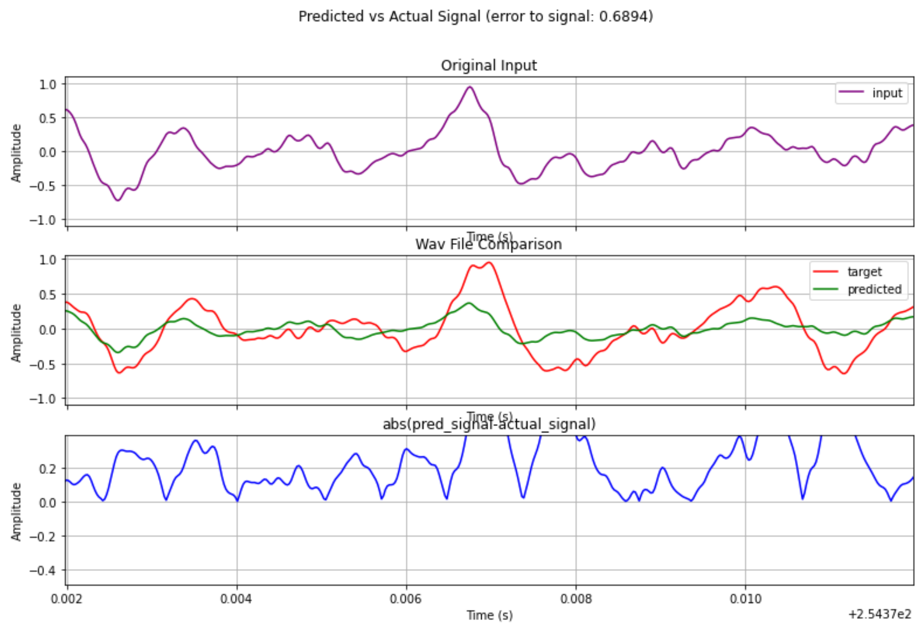


(b) Detailed signal comparison

Figure 4.1: Graph plots of Model 3 on Fuzz



(a) Signal comparison



(b) Detailed signal comparison

Figure 4.2: Graph plots of Model 1 on Chorus

Fuzz
As you can see by figure 4.1, in the previous section, model 3 does relatively well. It is also sonically convincing. Models 1 and 2 only performed slightly worse, but the bigger size of the hidden layer in model 3 allows it to generalize better. Model 3 took 23 minutes to train so it took the longest out of every model as well as early stopping at 130 epochs.

Chorus
Unfortunately, as you can see by figure 4.2, in the previous section, the best model out of the rest still had very poor performance and it was difficult to hear the audio effect applied. It was not sonically convincing. Model 1 did the best which is surprising as it has a small hidden layer. Also, Model 2 which has GRU, perhaps should have done better as the structure of a GRU allows for better learning of long term dependencies thus model time-varying effect more accurately. Model 1 took 10 minutes to learn and early stopped at 54 epochs.

For both graphs, it seems that the output and predicted audio waves are quieter than the input audio as the peaks of those waves are shorter than in the input. Both have differences in amplitude throughout the whole signal yet the predicted chorus signal has a bigger difference. As you can clearly see by the detailed signal comparison, the network can predict the audio waves closely to the target with the fuzz effect but is very poor with the chorus effect.

4.2 Model evaluation

Here are the performances of every model on both audio effects.

Table 4.1: Model performances on Fuzz

	Model 1	Model 2	Model 3
Error to signal	0.178	0.157	0.148
Best test loss	0.132	0	0.149
Best validation loss	0.135	0.131	0.110
Early stop epoch	112	68	130
Model training time(minutes)	20	12	23

Table 4.2: Model performances on Chorus

	Model 1	Model 2	Model 3
Error to signal	0.689	0.695	0.693
Best test loss	0.302	0.292	0.321
Best validation loss	0.512	0.521	0.519
Early stop epoch	54	74	62
Model training time(minutes)	10	13	11

As you see from table 4.1, model 3 performs the best (also shown in figure 4.1). Takes the longest to run at 23 minutes but has the best validation loss and error to signal scores. Model 2 preformed only a little poorer. Interestingly, it had a significantly quicker training time due to it having GRU units. Furthermore, it seems to over-fit as the test loss significantly smaller than the validation loss. Model 3 performed better than model 1 as it has a wider hidden layer, yet consequently took longer to train.

In table 4.2, performance in all models are poor and very similar to each other. Interestingly model 1 had the better performance while having a small hidden later. All models seems to over-fit as test loss is lower than validation loss.

4.3 Discussion

Although the model performance shown in the previous are not to a very high level, there is definitely potential for better performance by changing network structure and applying for data. I had experimented with a 128 hidden size network which was reaching 0.067 validation loss (for non-linear fuzz effect) before after an hour the training crashed due to computational bottlenecks.. I cannot play the audio files through a pdf document so it would be difficult to explain how the model does better than the numbers in terms of modelling the fuzz audio effect. The opposite is to be said with chorus as it can be difficult to discern if the audio effect is being applied or not. Having done some testing by recording my own guitar and applying it through the model I am pleased with the fuzz tones I am getting. Hopefully, the analysis in chapter 4 justifies this. To further continue the topic about performance, CPU and GPU usage was moderate and I am sure most modern computers with a graphics card could have similar performance to mine. I used google collab pro to access a better gpu and extra ram on the cloud. Compared to what I have read about wavenet, the models computational power requirements are tame. Since fuzz could be successfully modelled, I am sure by extension other non-linear audio effects would not be of concern. We cannot say the same about chorus. When testing the network on some guitar audio that I have made from playing and recording through a usb interface and ableton 10 DAW. The network was less sonically convincing when playing bends and other guitar sounds that were not included in the dataset. Perhaps more data, especially in playing certain techniques or generating certain sounds should be included in the future. The limitations of this approach is that only 44.1 kHz sample rate can be used and not higher, which improves the performance and time taken training the model. Also the quality in audio is just better which more of the sound wave sampled. Furthermore, this approach does not include audio effect settings or parameters found normally in analog devices or digital plug-ins. This approach only captures the audio effect in one instance, with its parameters set. In addition, the performance of this network depends on the quality of the recorded samples.

Chapter 5

Conclusion

Recurrent neural networks can definitely be used to model non-linear audio effects to a believable degree but not complex time-varying audio effects, further experiments and approaches would have to be explored. LSTMs and GRUs have performed well in emulating the nuances of a fuzz pedal but were not enough in learning the complex and long time-varying dependencies that a chorus alters on discrete time series data. The audio produced from these models are disappointing for chorus, different approaches must be explored such as a grey-box modelling approach which incorporates low-frequency oscillations which are used in time-varying effects. Compared to what is available right now for audio effect software, these models are not as good as them but not by a significant margin. I believe this method of modelling audio effects can be further developed and used in the future as I am sure I have not found the best end-to-end black box model. The neural network architecture I used was simple in terms of being a single LSTM layer and dense layer, more experimentation in structure can be investigated. In general LSTM's perform better than GRU's but are slower, the larger the layers the more computation needs but better performance too. I am excited to see what the researchers are working on next to build upon the state-of-the-art in audio modelling.

5.1 Future work

In terms of what I can do to build upon this project is to definitely apply this model in real time, perhaps making a neural based plug-in. Furthermore, adding more control parameters and knobs to alter the tone of the output like volume and gain. I would have liked to explore other models such as auto-encoders, TCN and wavenet architectures. For evaluation I would have also liked to add human listening tests to judge the performance of models.

Bibliography

[1] T. Vanhatalo, P. Legrand, M. Desainte-Catherine, P. Hanna, A. Brusco, G. Pille, and Y. Bayle, “A review of neural network-based emulation of guitar amplifiers,” *Applied Sciences*, vol. 12, no. 12, 2022. [Online]. Available: https://www.mdpi.com/2076-3417/12/12/5894

[2] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, “Wavenet: A generative model for raw audio,” 2016. [Online]. Available: https://arxiv.org/abs/1609.03499

[3] E.-P. Damskägg, L. Juvela, E. Thuillier, and V. Välimäki, “Deep learning for tube amplifier emulation,” in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 471–475.

[4] M. M. Halimeh, T. Haubner, A. Briegleb, A. Schmidt, and W. Kellermann, “Combining adaptive filtering and complex-valued deep postfiltering for acoustic echo cancellation,” in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 121–125.

[5] N. Takahashi, M. K. Singh, S. Basak, P. Sudarsanam, S. Ganapathy, and Y. Mitsufuji, “Improving voice separation by incorporating end-to-end speech recognition,” in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 41–45.

[6] J. Koo, S. Paik, and K. Lee, “Reverb conversion of mixed vocal tracks using an end-to-end convolutional deep neural network,” in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 81–85.

[7] F. Eichas and U. Zölzer, “Virtual analog modeling of guitar amplifiers with wiener-hammerstein models,” 03 2018.

[8] F. Eichas, “System identification of nonlinear audio circuits,” Ph.D. dissertation, 10 2019.

[9] Y. Wan, T. J. Dodd, and R. F. Harrison, “Modeling of power amplifier nonlinearities using volterra series,” *IFAC Proceedings Volumes*, vol. 38, no. 1, pp. 785–790, 2005, 16th IFAC World Congress. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1474667016361444

[10] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, “Wavenet: A generative model for raw audio,” in *Arxiv*, 2016. [Online]. Available: https://arxiv.org/abs/1609.03499

[11] A. Wright, E.-P. Damskägg, L. Juvela, and V. Välimäki, “Real-time guitar amplifier emulation with deep learning,” *Applied Sciences*, vol. 10, no. 3, 2020. [Online]. Available: https://www.mdpi.com/2076-3417/10/3/766

[12] M. A. Martínez Ramírez, E. Benetos, and J. D. Reiss, “Deep learning for black-box modeling of audio effects,” *Applied Sciences*, vol. 10, no. 2, 2020. [Online]. Available: https://www.mdpi.com/2076-3417/10/2/638