

Machine learning approaches to sound effect synthesis

Yangjiangdai Pan

School of Computing Science

Sir Alwyn Williams Building

University of Glasgow

G12 8RZ

A dissertation presented in part fulfillment of the requirements of the Degree of Master of Science at the University of Glasgow

September 2018

**Abstract**

From gramophone record to digital audio, the aspect of saving sound has changed during the last century. Digital technology not only makes music easier to be accessed, but also provides the musicians and the scientists a precise and reliable method to process sound. In the following report I will describe how I process audio files and train a neural network to emulate a specific guitar sound effect.

Education Use Consent

I hereby give my permission for this project to be shown to other University of Glasgow students and to be distributed in an electronic form.

<**Please note that you are under no obligation to sign this declaration, but doing so would help future students.>**

Name: Yangjiangdai Pan Signature: Yangjiangdai Pan

Acknowledgements

<Acknowledgements go here>

Contents

<Update the table of contents by right-clicking on it and selecting Update Field… and then select page numbers only.>

Chapter 1 Introduction 1

1.1 Research objectives 1

1.2 Report Structure 1

1.1.1 A subsection <This is style Heading 3> 1

Chapter 2 Analysis & Requirements 2

Chapter 3 Design & Implementation 2

Chapter 4 Testing & Evaluation details 3

Chapter 5 Conclusion 4

Chapter 6 References 5

Appendix A <Name of appendix> 1

Appendix B <Another appendix> 2

# Introduction

Though a small number of guitarists claim that the tone of digital effects sound “plastic”, the fact is that the popularity of digital devices is growing rapidly. More professional musicians choose to use digital devices such as Fractal Audio Axe FX and Kemper Profiling Amplifier to create their guitar tones. The reason why digital effects are replacing tradition electronic devices is that digital devices weigh less and take less spaces. Moreover, the sound processed by digital methods is close enough to traditional methods due to the increasement of precision.

Since all digital guitar effect devices are designed as emulations of physical electronic devices, using machine learning technology to emulate the effects of electronic devices can be the method to produce sound effects in the future.

## Research objectives

The project aims to train a neural network that can transfer clean guitar tracks to tracks that can be used in a music genre called Djent. In view of the uniqueness of sound effects, this project can be divided into two parts in general: audio processing and machine learning.

Audio processing:

Create a guitar sound effect for Djent music.

Extract the sound effects feature from audio and transfer it to a format that can be used for machine learning.

Design a method to transfer this digital feature back to audio.

Machine learning:

Build a neural network model which is suitable for time series.

Find out a training pattern that provides higher accuracy.

## Report structure

The following chapters of this report are Analysis & Requirements, Design & Implementation, Testing & Evaluation and Conclusion.

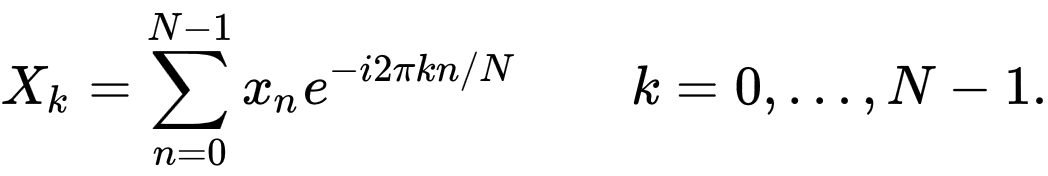
# Analysis & Requirements

## Background

Current machine learning projects related to audio focus more on classification. The solution of these kinds of projects is to extract a few features that can tell the difference between audios. They do not need to keep all the information of the audio during their processes. The aim of this project is to emulate the sound effect of a specific effect setting for guitar, which means all information of the input audio files should be kept during data processing. Unlike other kinds of data, digital audio is a time series (discrete data samples) data. The training model should be suitable for dealing with time series data.

### Fast Fourier Transform

The fast Fourier transform (FFT) is an algorithm for efficiently computing the discrete Fourier transform (DFT) of a time series.[1] This method gives a faster solution to deal with large amount of data than DFT. The formula of the DFT is:



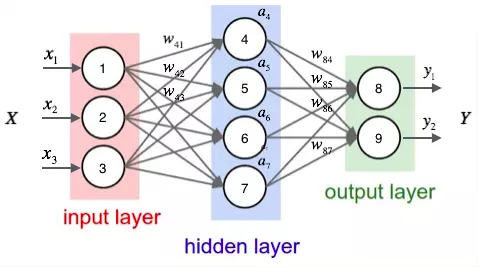
The complexity of computing the DFT is O(), where N is the data size. The FFT can give the same output and reduce the complexity to O(*Nl*og*N*). [2]

### Neural Networks

Neural networks are computing systems inspired by the [biological neural networks](https://en.wikipedia.org/wiki/Biological_neural_network) that constitute animal [brains](https://en.wikipedia.org/wiki/Brain).[3] It consists of a large number of connected processors called neurons like biological neurons. These systems can be trained with relevant data and perform tasks without given specific rules. The connection structure of a neural network can be changed due to the problem.[4]

### Recurrent Neural Network

Recurrent neural networks (RNNs) are neural networks that can be useful when training with sequences in domains, such as music. RNNs process real data sequences one step at a time and predict the next.[5] The structure of RNNS:



### Long Short Term Memory Recurrent Network

Long Short-term Memory (LSTM) is an RNNs architecture. It is better at storing and accessing information than standard RNNs architecture.[5] A typical LSTM consists of a cell, an input gate, an output gate and a forget gate. It suits processing and predicting time series data. The advantage of LSTM compares to standard RNNs is that an LSTM relatively insensitive to duration between time series data. [6]

### Keras

Keras is a relatively user-friendly high-level neural networks API. It aims at fast experimentation. It is easy to use for people who do not have experience in machine learning. [7]

## Requirements Gathering

The relevant technologies and information of this project is provided by Dr John Williamson, the supervisor of this project, in a series of meetings. The function of the program developed for this project is decided during these meetings.

## Requirements Analysis

This project aims at using machine learning approaches to achieve sound effect synthesis. The Program should be able to use clean guitar tracks and expected guitar tracks to train neural networks, and transfer clean guitar tracks to guitar tracks with expected tone.

The program of this project should be able to read digital audio files from given directory to digital numbers. Then these digital numbers can be divided and processed with STFT. The segments numbers in one STFT process should be able to be adjusted in the program. The real and imaginary parts of the outputs of the STFT process need to be read as separated real numbers, and reshape into single arrays due to the limitation of the calculation ability of the program. To increase the efficiency and the endurance of dirty data, the output arrays after STFT processes of separate audio files should be reshaped into one array and shuffled.

An LSTM neural network model needs to be built. The input and output size of the model should be the same as the training arrays. The training process should be easy to change. The number of epochs and the size of batches should be easy to change so the training method can be improved during training. The included files in each process can be selected to make sure each audio file be trained for equal times. This is aiming to increase efficiency. Currently best weights of the neural network should be recorded to avoid over-saturation. After a round of training process, the neural network model should be saved for later use.

The saved weights and model of the neural network should be available at any time. The program should be able to read test audio files from given directory and predict the output with existing models and weights. The output should be audio files that saved in an assigned location and easy to be listened to.

# Design & Implementation

This

## System Architecture

The architecture of the program can be described as Model-View-Controller design pattern. The View is output of audio files without a typical GUI since this project is for research. The only user is also the developer of the program.

## Design Details

The program has 3 major stages:

1. Audio processing
2. Training
3. Prediction

### Audio processing

The audio data training is recorded with Reaper. The frame rate and sample depth of all the audio files (including the input/output of training/testing data) is 44100HZ and 16bit separately. Screenshot of the UI of Reaper:(图)

All tracks are recorded in mono channel so that they can be easy to be used in later processes. After recording, the clean guitar tracks and Djent guitar tracks will be put in separate folders.

The program uses wavfile model from Python package SciPy to read the audio files. It gives the frame rate of the files and represents the data of each frame with a 16bit number. Codes of this process: (截图)

Then, the digital data is processed with STFT. The programs uses signal model from SciPy package to do STFT. Codes of this process: (截图)

The output is numbers consists of real and imaginary numbers. The real and imaginary part of the output is separated and reshaped to a single array. (All array modification is done with NumPy package.) The arrays of multiple audio files are combined for increasing training efficiency. The combined array is shuffled before being trained with the neural network. The codes are as follow: (截图)

### Training

The deep learning model of the neural network in this program is built with Keras. It is a 3 layers sequential LSTM model. The codes are as follow: (截图)

The training audio files, including training input and output, are transferred to arrays with shape that suits the model after above audio processing stage. The codes of training process are as follow: (截图)

### Prediction

The function of prediction stage is transfer clean guitar track to Djent guitar track. Like the training stage, raw audio input file is processed by functions of audio processing stage before being used as input data to create prediction data by the models and the weights saved in training stage. The output data is then modified with the inverse operation of STFT by using signal model from SciPy package. The output of STFT inverse operation are digital numbers that represent frame information of audio. These numbers are then write into audio files. The codes are as follow: (截图)

# Conclusion

Show how you plan to organise your work, identifying intermediate deliverables and dates.

# References

1. Cochran W T, Cooley J W, Favin D L, et al. *What is the fast Fourier transform?*[J]. Proceedings of the IEEE, 1967, 55(10): 1664-1674
2. *Fast Fourier transform* url: <https://en.wikipedia.org/wiki/Fast_Fourier_transform>
3. *Artificial neural network* url: <https://en.wikipedia.org/wiki/Artificial_neural_network>
4. Schmidhuber J. *Deep learning in neural networks: An overview*[J]. Neural networks, 2015, 61: 85-117.
5. Graves A. *Generating sequences with recurrent neural networks*[J]. arXiv preprint arXiv:1308.0850, 2013.
6. *Long short-term memory* url: https://en.wikipedia.org/wiki/Long\_short-term\_memory
7. *Keras Documentation* url: https://keras.io/
8. Eck D, Schmidhuber J. *A first look at music composition using lstm recurrent neural networks*[J]. Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale, 2002, 103
9. Längkvist M, Karlsson L, Loutfi A. *A review of unsupervised feature learning and deep learning for time-series modeling*[J]. Pattern Recognition Letters, 2014, 42: 11-24
10. Bretan M, Weinberg G, Heck L. *A Unit Selection Methodology for Music Generation Using Deep Neural Networks*[J]. arXiv preprint arXiv:1612.03789, 2016.
11. Ngiam J, Khosla A, Kim M, et al. *Multimodal deep learning*[C]//Proceedings of the 28th international conference on machine learning (ICML-11). 2011: 689-696.
12. Hamel P, Eck D. *Learning Features from Music Audio with Deep Belief Networks*[C]//ISMIR. 2010, 10: 339-344.
13. LeCun Y, Bengio Y, Hinton G. *Deep learning*[J]. nature, 2015, 521(7553): 436.

###### <Name of appendix>

<Use Heading 6 for the Appendix heading>

###### <Another appendix>