Deep Learning in Music Genre Classification

Brian Fletcher, Claire Holton, Alexandre Yiou Li,

# Introduction

There are many successful approaches taken to the problem of music classification. These approaches have varying levels of success. This project attempts to apply deep learning techniques to music genre classification. The objective is to classify various music files into 10 different genres.

To do this, we use the dataset published by Marsyas [<http://marsyas.info/>]. It contains 1,000 audio files. The files are provided in au file format. Each audio file (~1 MB) is a single track (i.e. mono) of approximately 30 seconds long and a sample rate of 22050Hz. [<http://marsyas.info/downloads/datasets.html>] The dataset is made up of 10 genres of 100 files as shown in figure 1. We tried two approaches and documented their results.

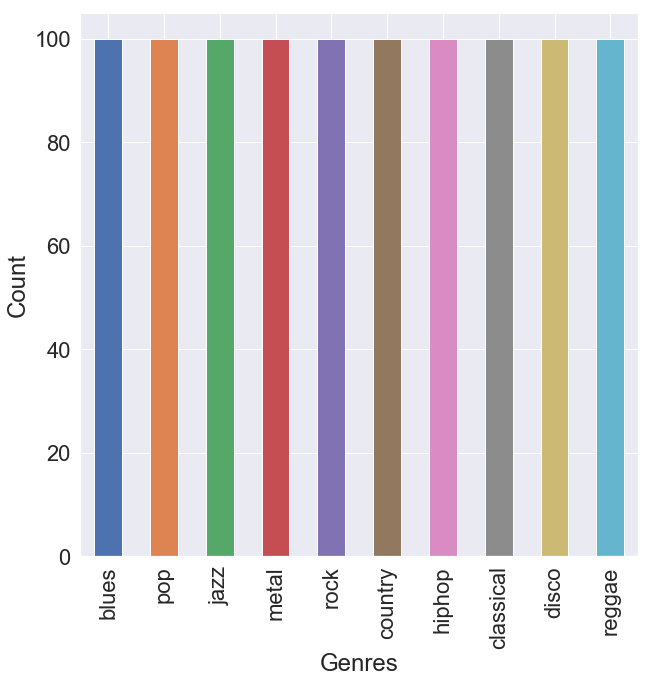


Fig 1. Bar chart showing distribution of audio samples

# 2. Requirements

We made extensive use of Python3, Jupyter Notebooks, Google Cloud Engine and Tensorflow 1.13.1 throughout this project. We used the Python libraries Librosa and Audioread for reading the audio files. We made use of the Python libraries Mathplotlib and Seaborn for data visualization. The Python library SK Learn was used for pre-processing the data.

As well as reading the raw audio files, we used the Python library Librosa extensively for audio file feature extraction and analysis of the files.

We used the genres dataset published by Marsyas [<http://opihi.cs.uvic.ca/sound/genres.tar.gz>], untarred using `tar -xvf genres.tar.gz`.

# 

# Description

## Data Preparation

**Short Term Fourier Transform**

Fourier transform is a ucs. It takes an audio segment and decomposes it into its constituent frequencies. Short term Fourier transform is the process by which a longer segment of audio is split into evenly sized time segments and a Fourier transform is applied on each.

The short term Fourier transform of an audio file is a two dimensional array containing values that represent the intensity of a frequency at a specific time period. The following diagram shows a Mel spectrogram of an audio signal decomposed using short term Fourier transform.

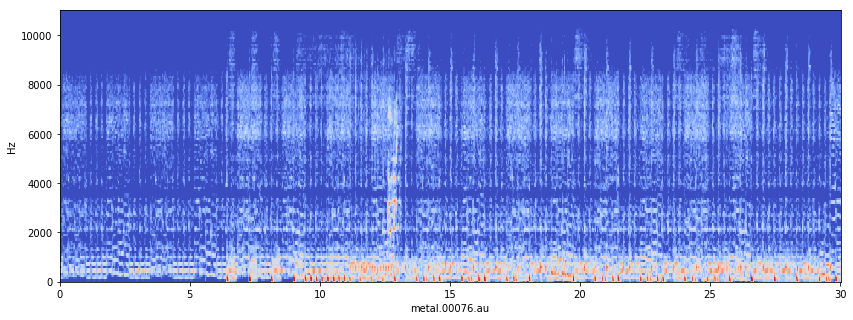


Fig 2. Mel spectrogram

A spectrogram is a representation of sound showing the volume of the frequencies that make up the sound in brightness. Normalized, this data can be used as input to a deep learning network.

The data is further used to extract other metadata as shown in Fig 3. It was extracted from the 1000 audio files and posted to Kaggle.

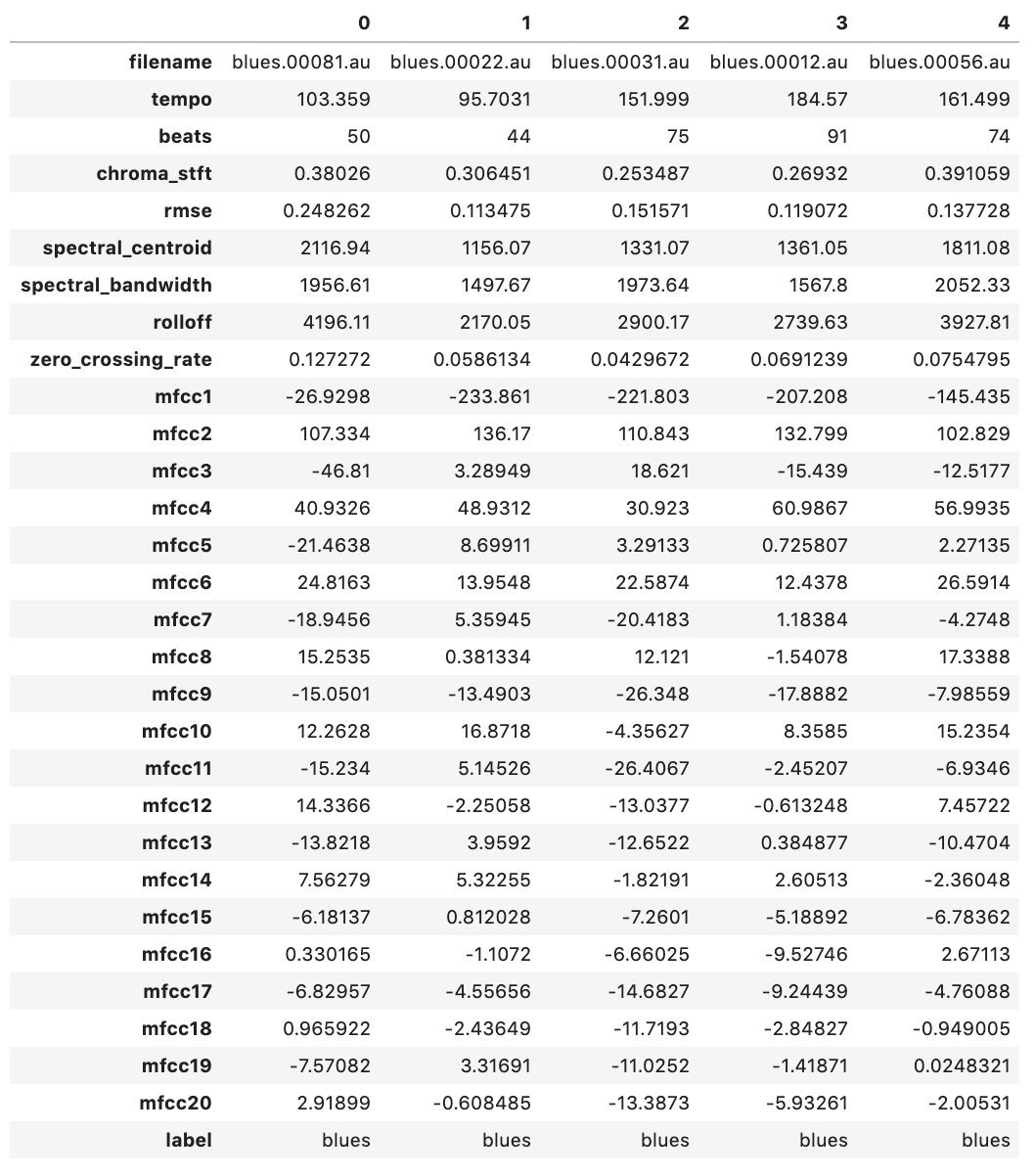


Fig 3. Examples of feature set from [<https://www.kaggle.com/insiyeah/musicfeatures#data.csv>]

We looked at each feature in the dataset. As can be seen from the histograms in figure 4, the majority of features are normally distributed. The feature ‘mfcc1’ appears to be skewed.

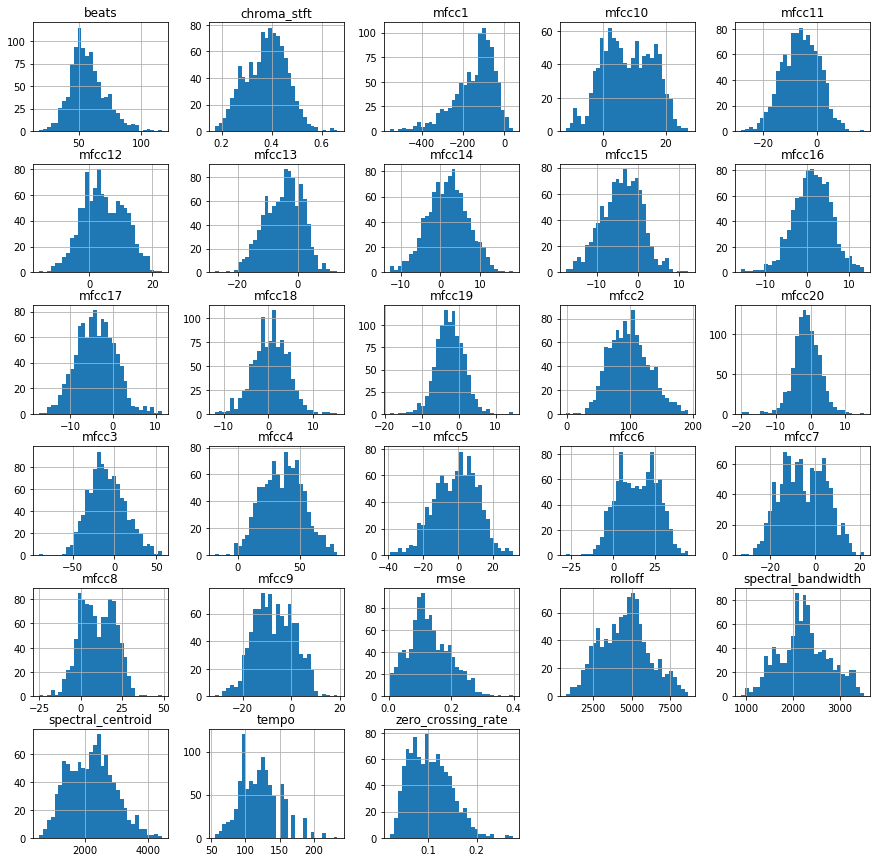


Fig 4. Histogram for each numeric input variable

We then looked at the correlation between each feature to get an indication of which variables have a high correlation with each other.

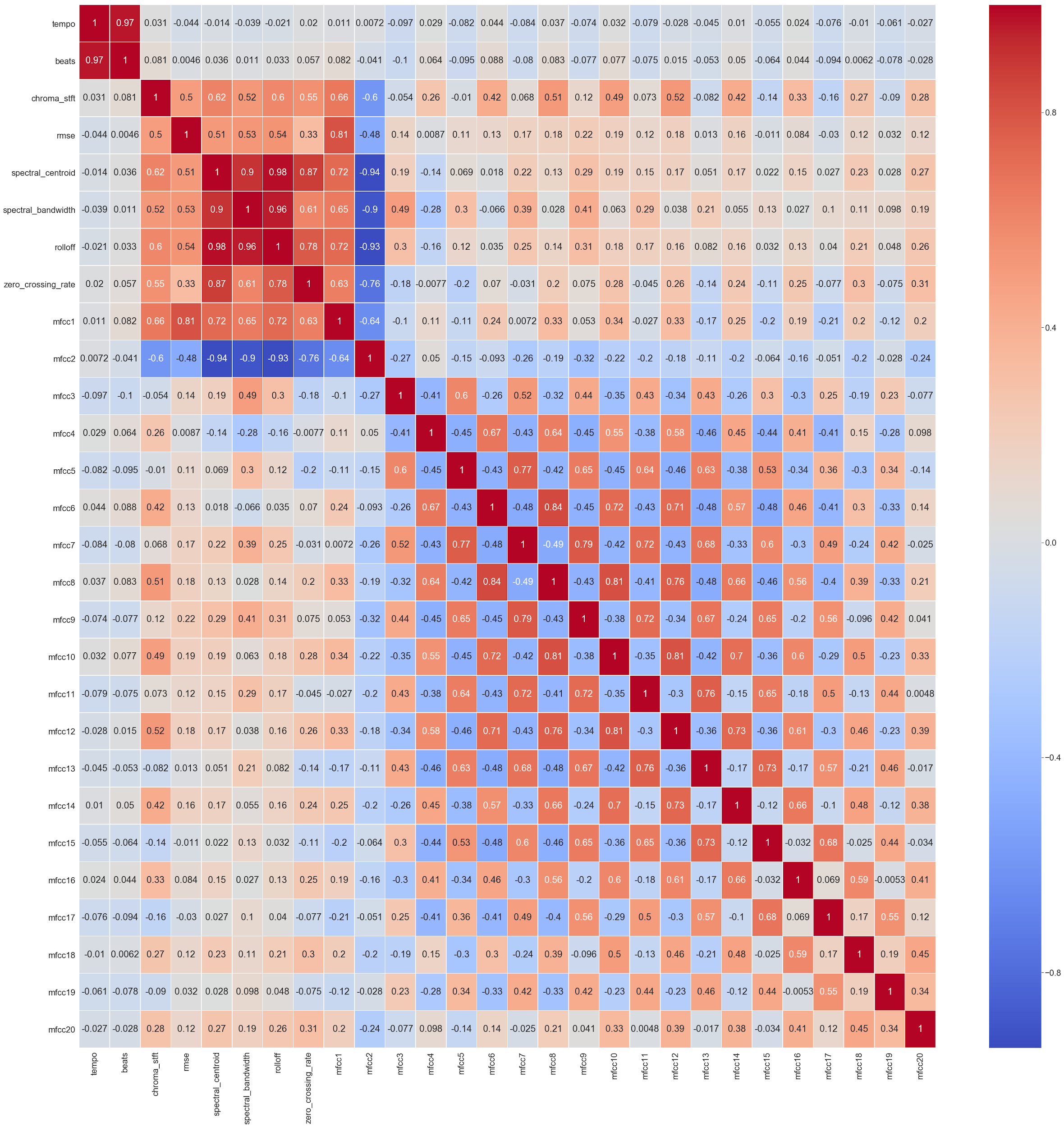


Fig 5. Correlation Matrix

We can see that the correlation matrix is symmetrical. From looking along the diagonal from left to right, each variable is perfectly positively correlated with each other.

**Distribution of Classes**

As can be seen in figure 1, all 10 classes are uniformly distributed. This is important as it shows there is no bias in the dataset.

**Normalization**

Data provided as inputs to the network were normalized to values between 0 and 1. Also in the case of the multilayer perceptron architecture, we used PCA with 16 components to reduce the dimensions of the data to input.

## **Architecture**

We took a number of different approaches. We attempted to use CNN and RNN with the raw frequency samples from the audio file. We were unable to make RNN work due to system memory constraints. Although we faced memory constraints with CNN, by reducing the size of the input data (truncation), we got test accuracy around 20%. We attempted CNN using the Mel spectrogram as an input image. A number of changes to the network configuration gave a test accuracy to a maximum of 37%.

We finally settled on using the dataset provided on Kaggle as an input to a multilayer perceptron.

**CNN**

The input data was short term Fourier transformation of 5 second segments of the audio tracks. These features are normalized between 0 and 1. The input shape is 1025 rows and 276 dimensions. This is fed into a CNN network as shown below in figure 6.



Fig 6. CNN Network

The image goes through two rounds of a 2D convolution, max pooling and drop out before being fed into a perceptron network. The output layer of the perceptron contains 10 nodes. Each node represents a single genre using one hot encoding.

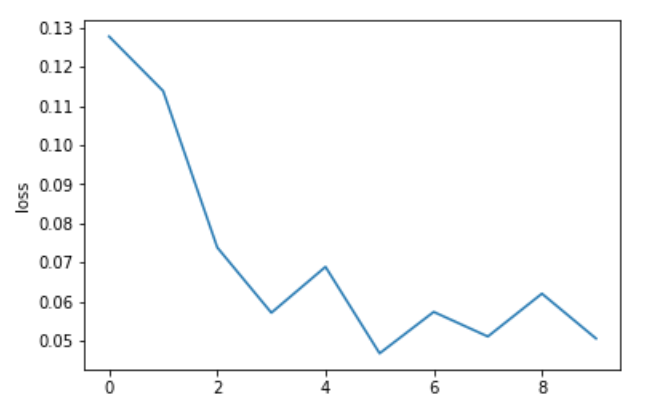


Fig 7. Loss curve for CNN

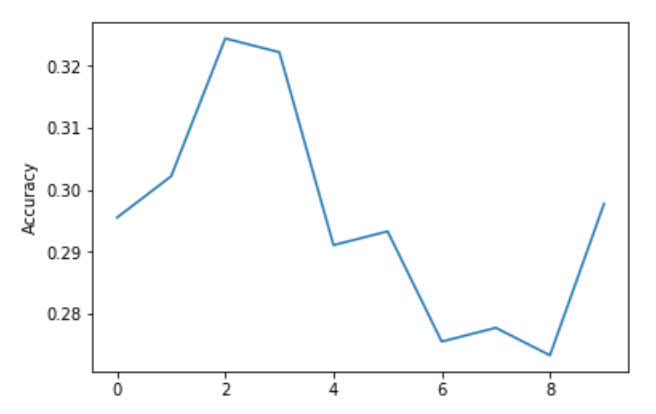


Fig 8. Accuracy curve for CNN

**Multilayer Perceptron**



Fig 9. Multilayer perceptron with 3 layers

The 18 raw feature data was preprocessed with PCA down to 16 features and then normalized between 0 and 1. The 16 features were fed into the input layer of an MLP. The labels are converted to one hot encoding for use in Softmax calculation in the output layer. We use 25% of the audio samples aside for testing. We used 10% of the training data for epoch accuracy validation and 400 epochs.

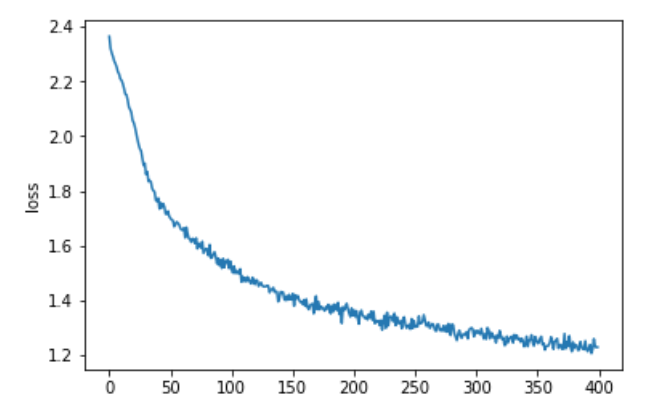


Fig 10. The loss curve for the multilayer perceptron

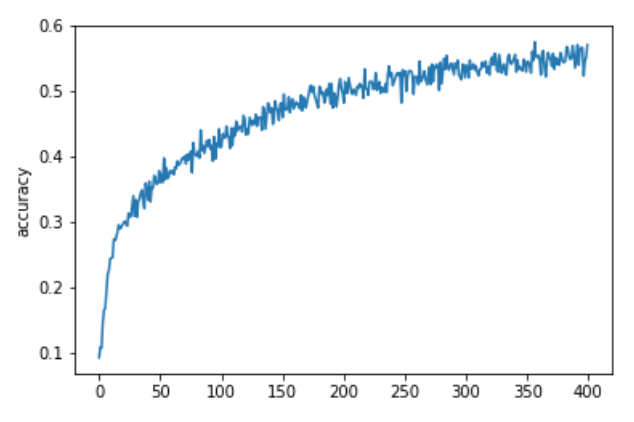


Fig 11. The accuracy curve for the multilayer perceptron

The CNN network achieved an accuracy of 40% and the MLP achieved an accuracy of 58%.

# Conclusion

We successfully extracted features from the dataset, using a few methods. We explored three different deep learning techniques: RNN, CNN and MLP in order to classify the music clips. We achieved the best accuracy using the multilayer perceptron.

CNN works well with images but trying to apply it to more complex data proved to be unreliable.

RNN might work better than CNN since the data at time t depends on the data at time t-1. This is something that could be explored given appropriate compute power.