Springboard--DSC Program
Capstone Project 2 Milestone Report

Music Genre Classification from Audio Samples
By Morgan Fry
June 2020

The Problem

The classification of music by genre is a common problem with current applications, for instance recommendation engines used by companies like Spotify and Pandora. The inherently subjective nature of music makes systematic description and classification difficult. In the past machine classifications were made according to song metadata (Artist, label, etc.), but it is increasingly possible to achieve this using the audio data themselves.

The Data

The Free Music Archive was compiled for purposes of evaluating various tasks in MIR (Music Information Retrieval). It contains over 100,000 songs in both 30 second clips and entire tracks. It also contains extensive metadata about each track such as artist, genre, etc., as well as extracted audio data such as MFCCs.

The genre data is multilabel, many songs are classified as for instance rock and blues.

Data Wrangling

The dataset used is from the Free Music Archive. It is a dataset of roughly 100,000 30-second clips of music along with track data such as artist, genre, etc. and also some extracted features.

Much of the dataset is already well organized. There are .csv files containing track and feature information as well as 100,000 30-second musical clips in .mp3 format. To ease comparisons between different models, there is a train/validation/test split already annotated, as well as small, medium, and large sets of data. To use these clips in a keras model it was necessary to build a keras data generator so that model training can load the data in batches. .mp3 is a compressed audio format that also contains metadata about the file. So the first part of the preprocessing pipeline uses the common utility FFmpeg to decode the files and load them into numpy arrays to feed into the model training.

There was also a small matter of dealing with corrupted data. There were 164 tracks that didn't contain any audio data, so they had to be passed over. The necessary information

for this project -- the track id, set info, filepath, and classes -- and was saved in a dataframe for later use:

	split	subset	filepath	genres_all	genre_top
track_id					
2	training	small	data/fma_large/000/000002.mp3	[21]	Hip-Hop
3	training	medium	data/fma_large/000/000003.mp3	[21]	Hip-Hop
5	training	small	data/fma_large/000/000005.mp3	[21]	Hip-Hop
10	training	small	data/fma_large/000/000010.mp3	[10]	Pop

Exploratory Data Analysis

The most common visualization of sound is a plot of the amplitude envelope of the soundwave. This is equivalent to plotting the 1-d array containing the sample as a time series. The shape of the array is sample rate * sample length in seconds.
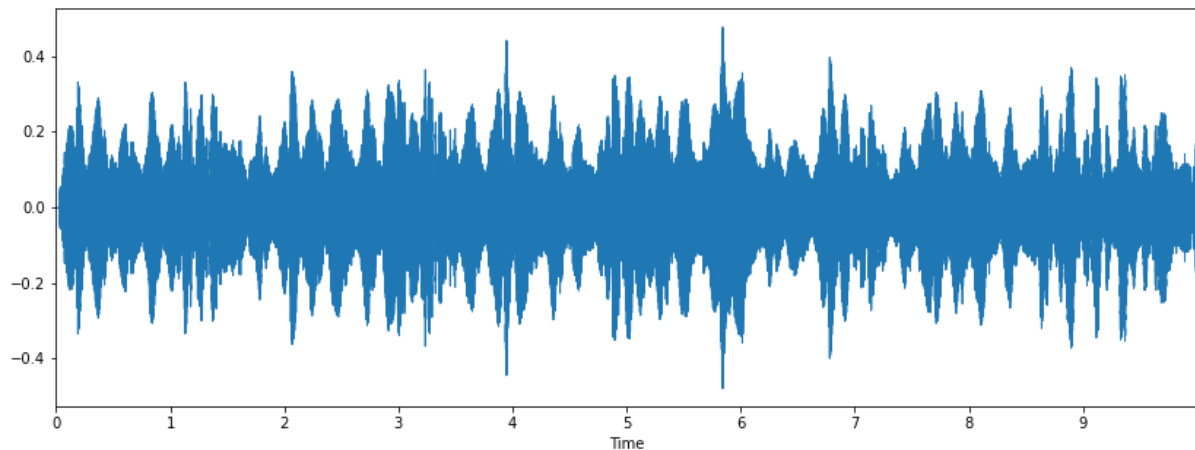
It is more useful for our purposes to look at the frequency spectrum. Not just how loud the entire sample is at any time but what frequencies are present. Musical instruments have most of their sound in the 50-2000 Hz range, and as we can see there is a lot more information in the lower frequencies, so looking at the log of the frequencies will be more instructive.
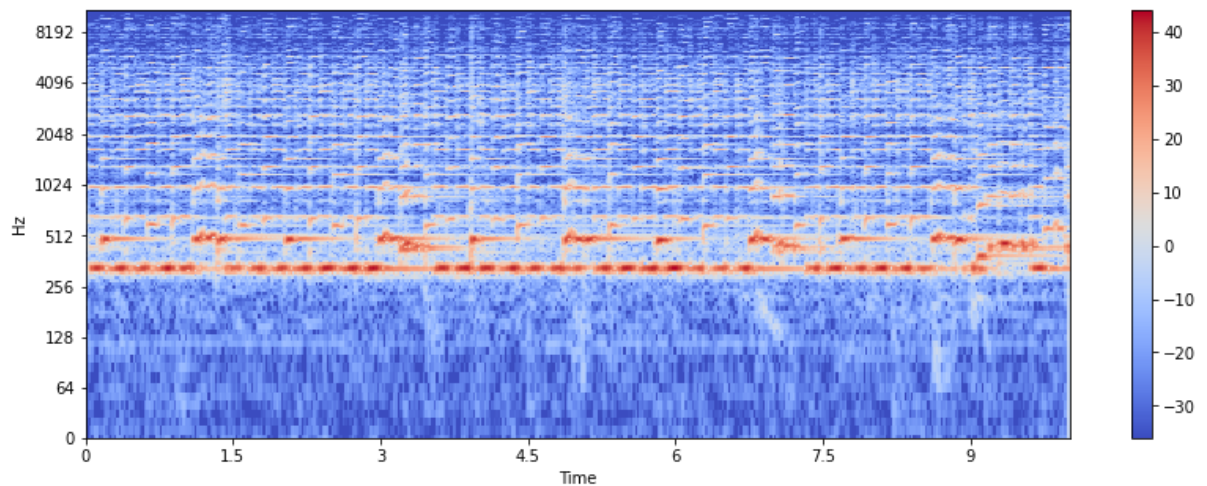
In this case it is a solo violin, so the open strings are from about 200Hz to about 700Hz, and we can see that most of the sound is in the 300Hz to 2000Hz range.

Even more useful is the use of Mel-frequency cepestral coefficients (MFCC). An MFCC is a representation of the power spectrum of a sound, it uses the following transformations of the signal:

- Fourier transformation of the signal
- Mapping that onto the mel scale
- Take the log of the powers at each mel frequency
- Discrete cosine transform of those log powers

The mel scale noted above is a scale of pitches that we perceive to be equidistant from each other. For instance every multiple of 440Hz is the note 'A', the difference between 220 and 440 is perceived as the same as the difference between 440 and 880.
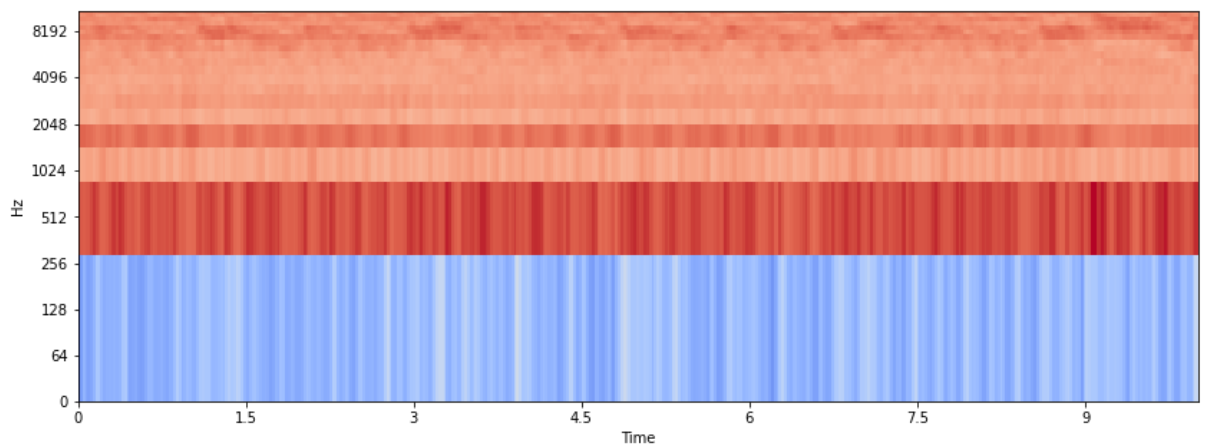
Now that we have established some of what the data looks like, let's look at the distribution of the labels. We are going to attempt to classify these clips by genre.

Frequency of genres
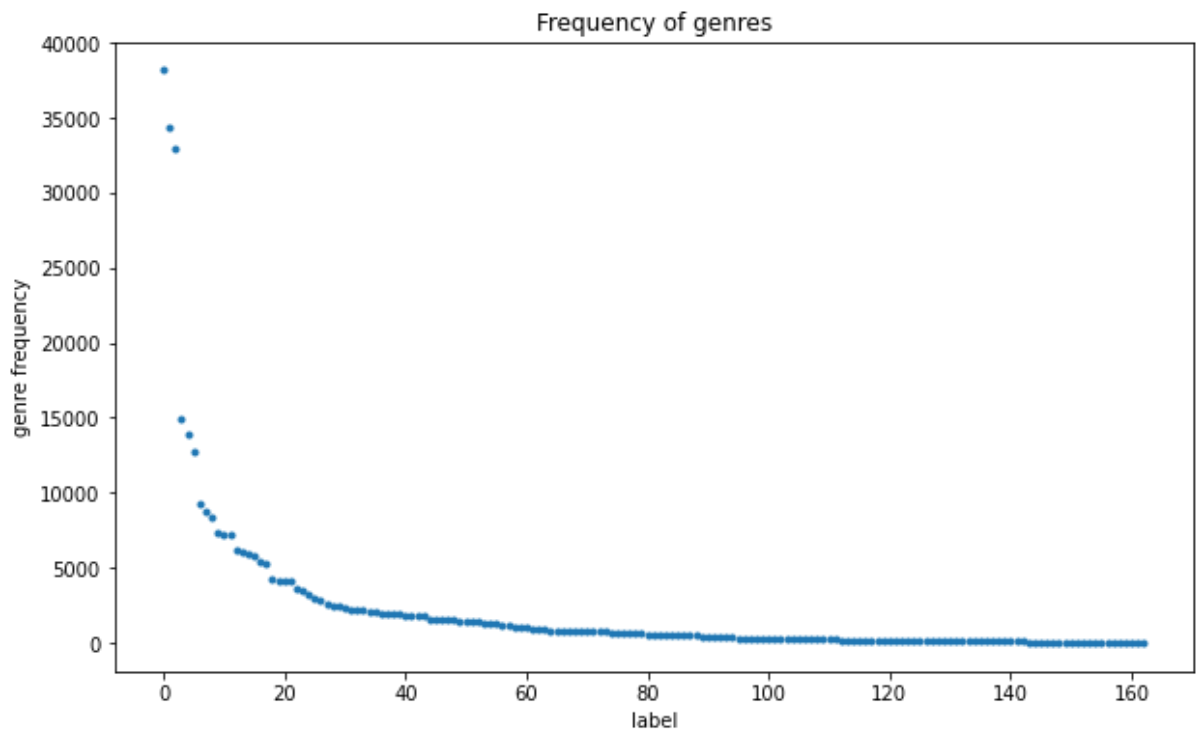
We can see that out of 163 genres, only about 25 have more than 3000 occurrences. Keep in mind that many tracks have multiple labels, both hierarchical and flat. For instance, [1, 12, 38, 90, 250] indicates:

- 1 = Avant-Garde
- 12 = Rock
- 38 = Experimental (top-level and parent of both [1]Avant-Garde and [250]Improv)
- 90 = Sludge (child of [12]Rock)
- 250 = Improv

Examining the features for correlations

In order to get an idea of how well correlated the features were, I plotted a correlation matrix between the features that are most important to how we interpret sounds. First the Chroma CENS.A chroma vector is a typically a 12-element feature vector indicating how much energy of each pitch class, {C, C#, D, D#, E, ..., B}, is present in the signal. Chroma energy normalized statistics (CENS) take statistics over large windows, smoothing local deviations in tempo, articulation, and musical ornaments such as trills and arpeggiated chords. This makes CENS useful for tasks such as audio matching and similarity.

Chroma CENS correlations
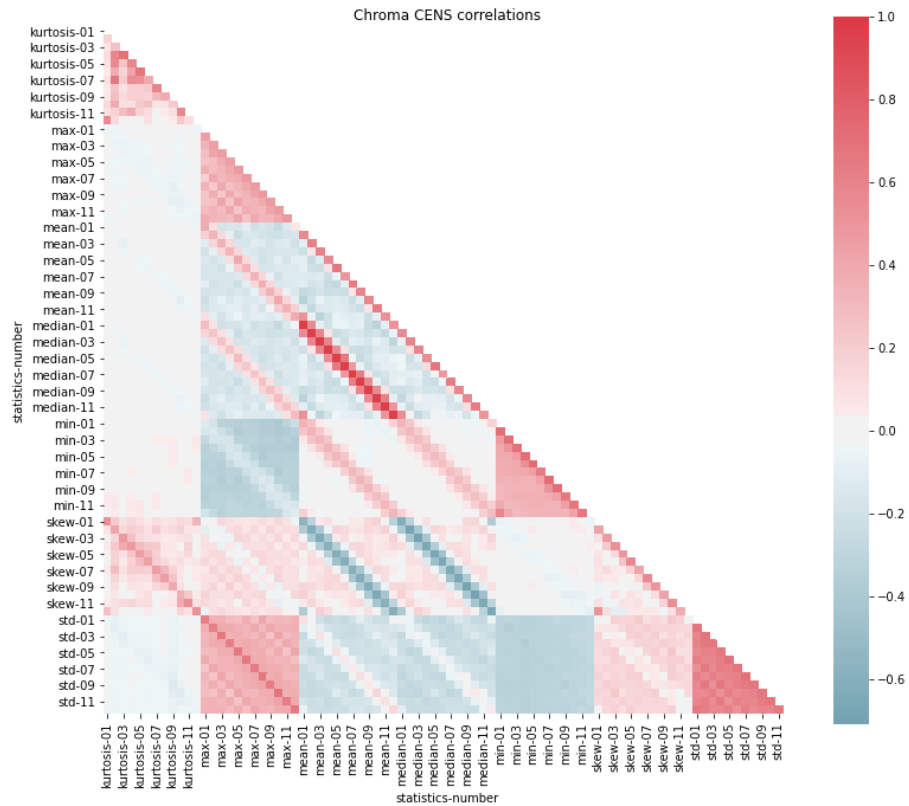
We can see some patterns of regularity. This indicates that the features of neighboring time slices tend to be highly correlated. This makes sense when you consider that a song will typically sound the same from one 23ms sample to the next. When we isolate just the highest positive and negative correlations:


Chroma CENS correlations >.7
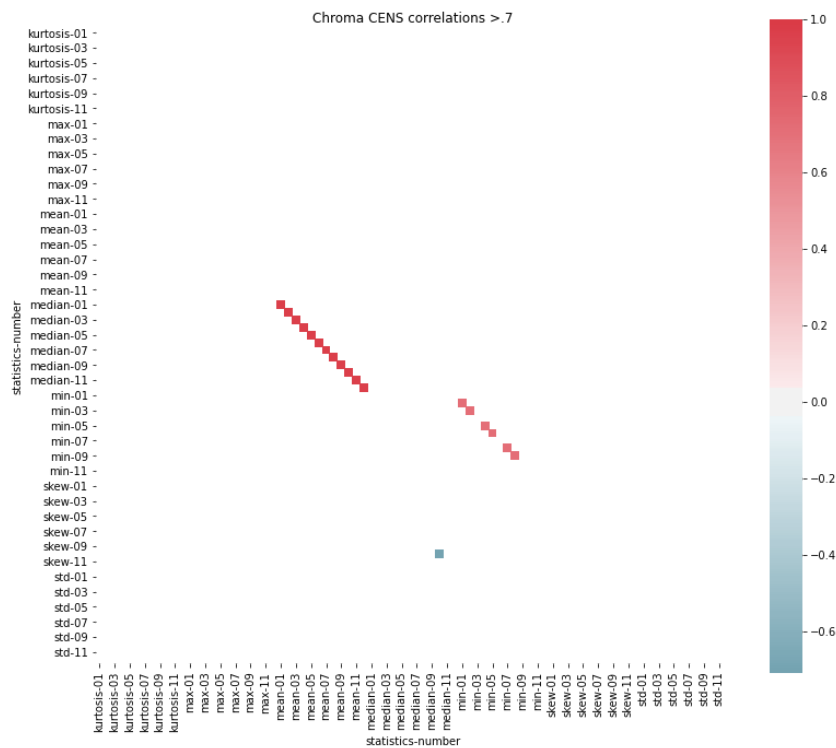
There are only a few pairs with a pearson r value of over .7 or under -.7. The one large negative value is the median/skew pair from sample 11 which has an r-value of -.867 and a p-value below .001.

As calculating the Mel-scale Frequency Cepstral Coefficients (MFCC) is a common transformation used for music classification, these were also examined for correlation. The correlation matrix for the highest absolute value pairs:
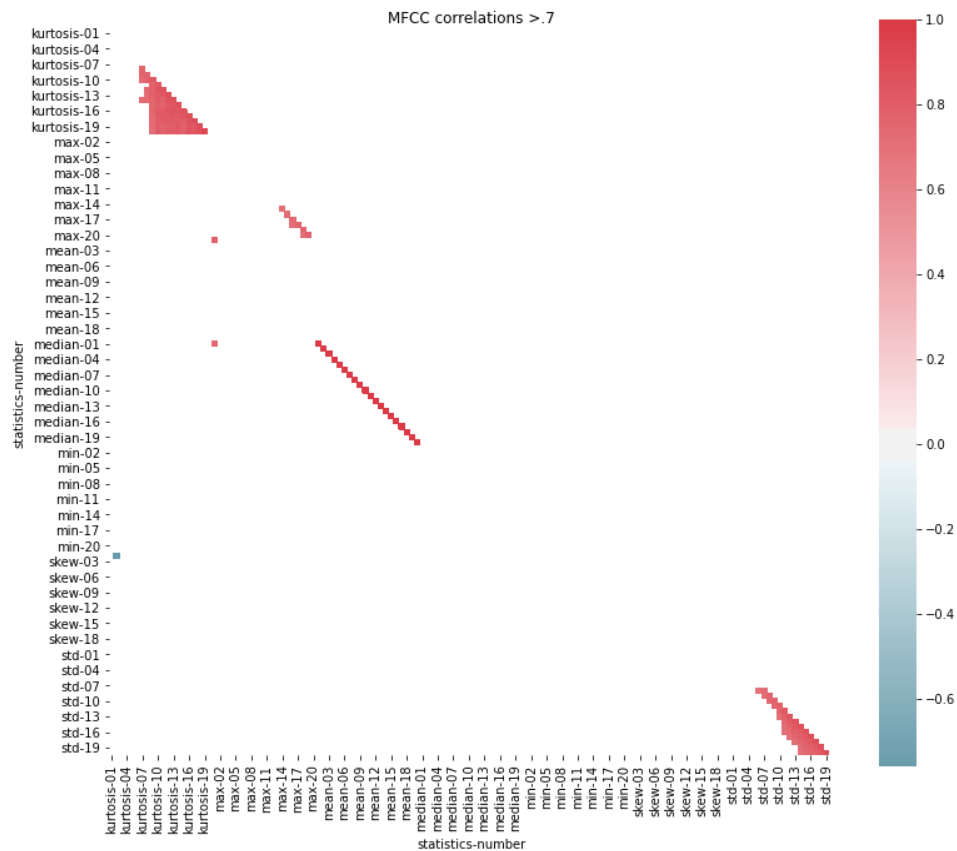
Looking at statistical features of the mfccs, such as max, min, median, skew, kurtosis, etc. From the documentation of the librosa functions that constructed the mfccs, we can see that they are ~23ms long, with 50% overlapping the previous and next samples. We can notice a few things from the matrix of pairwise correlations above:

- The columns that have strong correlation are generally close to each other. e.g. median 10 and median 13 have correlation of .706
- Most of the correlations can be attributed to nearness in time to each other and the fact that they share some underlying data due to the sampling overlap
- Skew 03 and Kurtosis 01 are the only outliers to this pattern with a -.867 pearson coefficient and p-value of less than .001

Pearson Coefficients between MFCC features
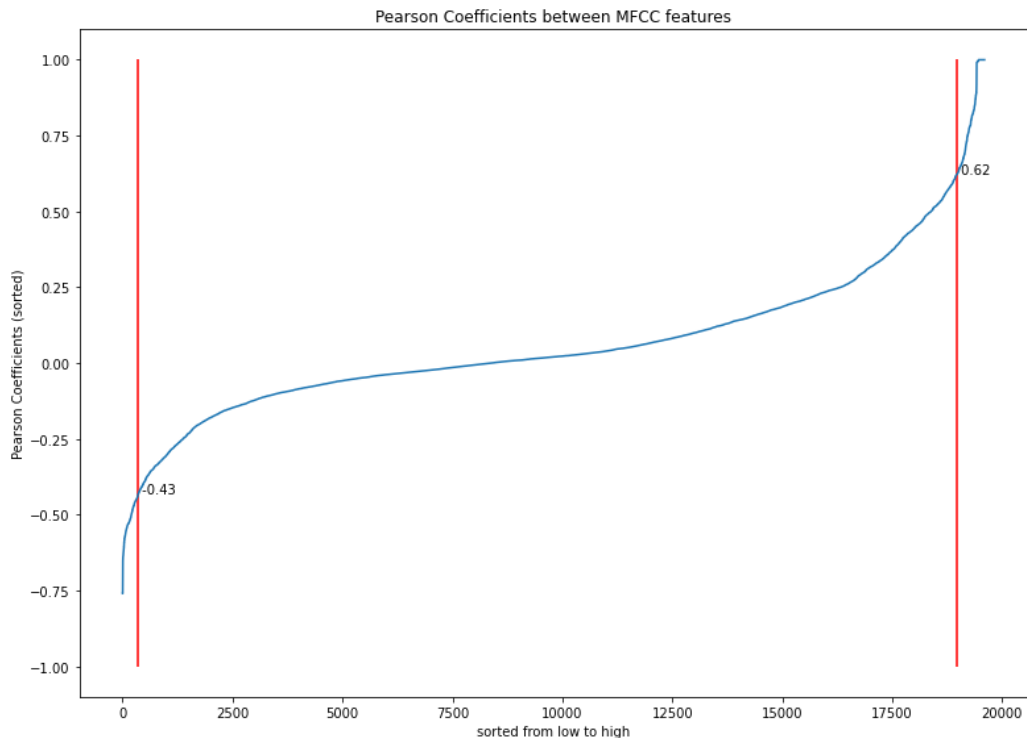
Plotting the distribution of the correlation values, We see that 95% of the pairwise correlations are between -.43 and .62. With only a few instances of high collinearity we may drop the highly correlated columns if it can improve performance of the model.

Baseline Model

For baseline modeling I used FMA's reduced set of 8000 tracks in 8 balanced classes. I compared the performance of a logistic regression in Scikit-Learn to the performance of a fully connected neural net with 3 hidden layers built in Keras. The FMA dataset has already been separated into train, test, and validation sets, so that models from different sources can be compared to one another if desired. I used this split in all modeling.

Logistic Regression

First, a logistic regression with default parameters achieved an overall accuracy of .22.

Classification Report (test set, logistic regression):

	precision	recall	f1-score	support
0	0.28	0.34	0.31	100
1	0.18	0.25	0.21	100
2	0.15	0.13	0.14	100
3	0.28	0.19	0.23	100
4	0.22	0.23	0.23	100
5	0.29	0.19	0.23	100

6	0.11	0.10	0.10	100
7	0.30	0.36	0.33	100
accuracy			0.22	800
macro avg	0.23	0.22	0.22	800
weighted avg	0.23	0.22	0.22	800

Next, I used a fully connected neural net with 3 hidden layers over the same dataset. The architecture is as follows:

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 512)	11895296
dense_2 (Dense)	(None, 256)	131328
dense_3 (Dense)	(None, 128)	32896
dense_4 (Dense)	(None, 64)	8256
dense_5 (Dense)	(None, 8)	520

Total params: 12,068,296
Trainable params: 12,068,296
Non-trainable params: 0

The model was trained with a batch size of 256 and 10 epochs. The accuracy and loss over the training set:
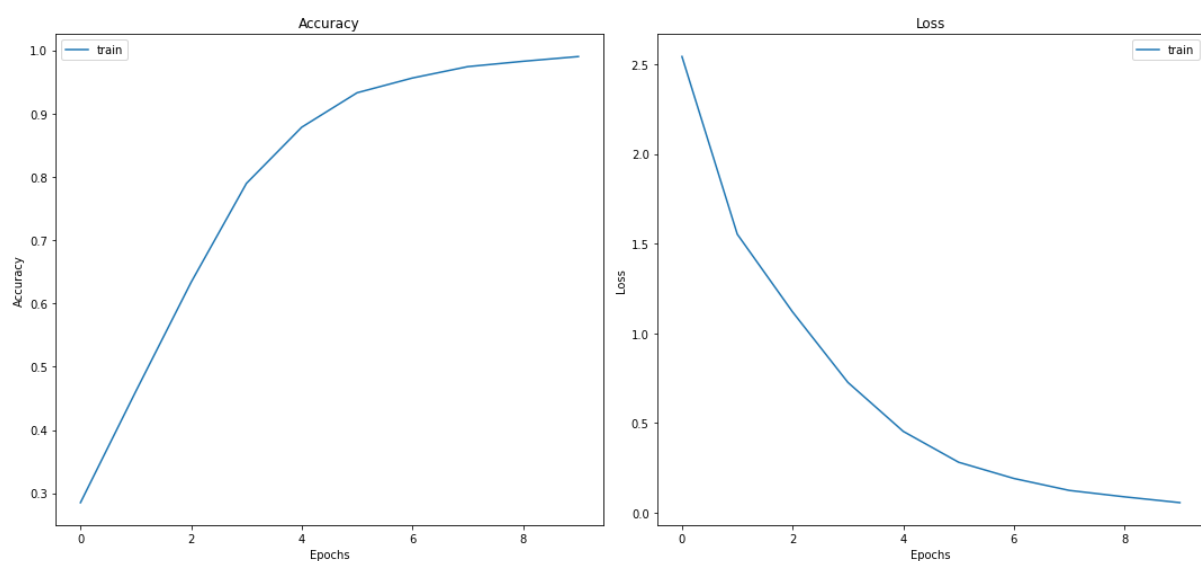
The overall performance over the test set was better than the logistic regression.

Classification Report (Neural Net, test set)

	precision	recall	f1-score	support
0	0.34	0.35	0.34	100
1	0.12	0.12	0.12	100
2	0.20	0.20	0.20	100
3	0.49	0.49	0.49	100
4	0.35	0.41	0.38	100
5	0.35	0.33	0.34	100
6	0.15	0.14	0.15	100
7	0.40	0.40	0.40	100
accuracy			0.30	800
macro avg	0.30	0.30	0.30	800
weighted avg	0.30	0.30	0.30	800

Over the set of:

- 8 classes
- 1000 samples each class

A baseline model of a fully connected neural net achieves better accuracy over the test set than a Logistic Regression:

Model	Accuracy(all classes)	Train Time
Logistic Regression	.22	2m 36s
Neural Net	.30	8.34s

I will extend this model using more complicated neural net architectures as follows:

- Convolutional Neural Net (CNN)
- Long Short Term Memory (LSTM)
- Combination architecture with both CNN and LSTM layers