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.

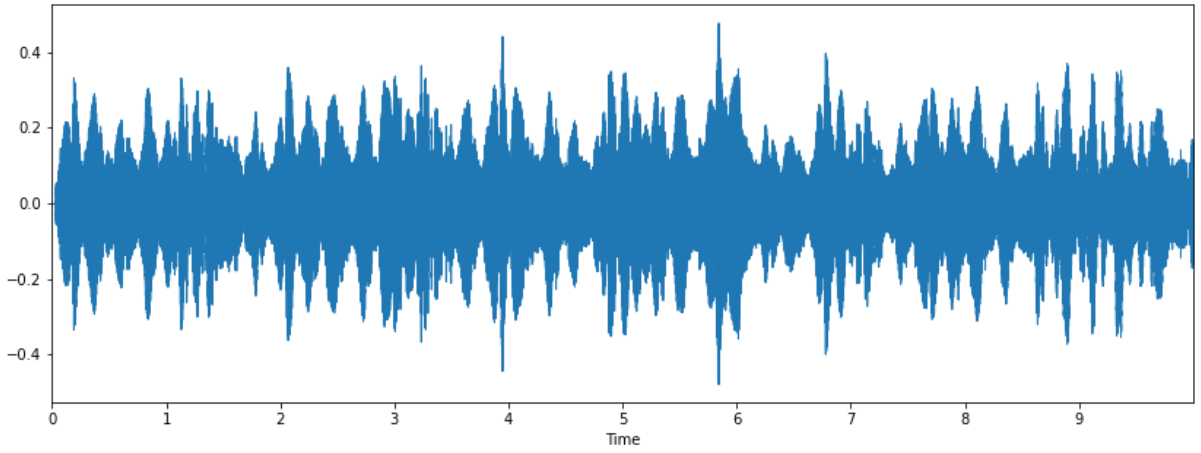
There was also a small matter of dealing with corrupted data. There were 164 tracks that didn't contain any audio data, so they had to be passed over. The necessary information

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

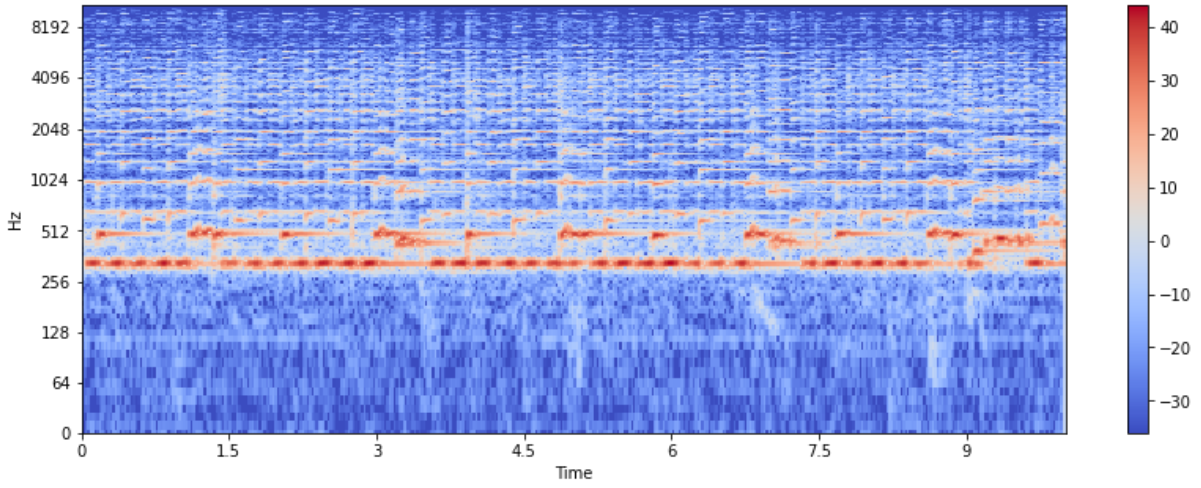
track_id	split	subset	filepath	genres_all	genre_top
2	training	small	data/fma_large/000/000002.mp3	[21]	Hip-Hop
3	training	medium	data/fma_large/000/000003.mp3	[21]	Hip-Hop
5	training	small	data/fma_large/000/000005.mp3	[21]	Hip-Hop
10	training	small	data/fma_large/000/000010.mp3	[10]	Pop

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.

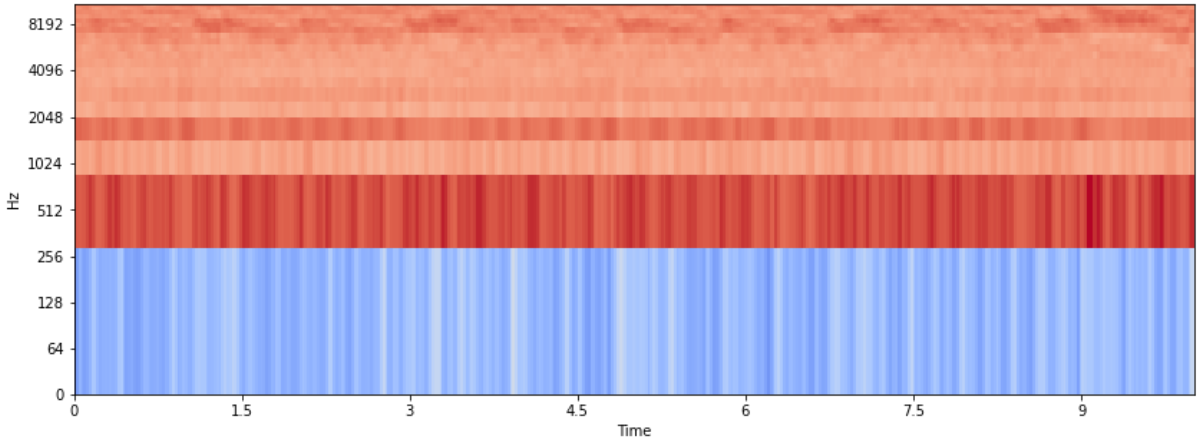


In this case it is a solo violin, so the open strings are from about 200Hz to about 700Hz, and we can see that most of the sound is in the 300Hz to 2000Hz range.

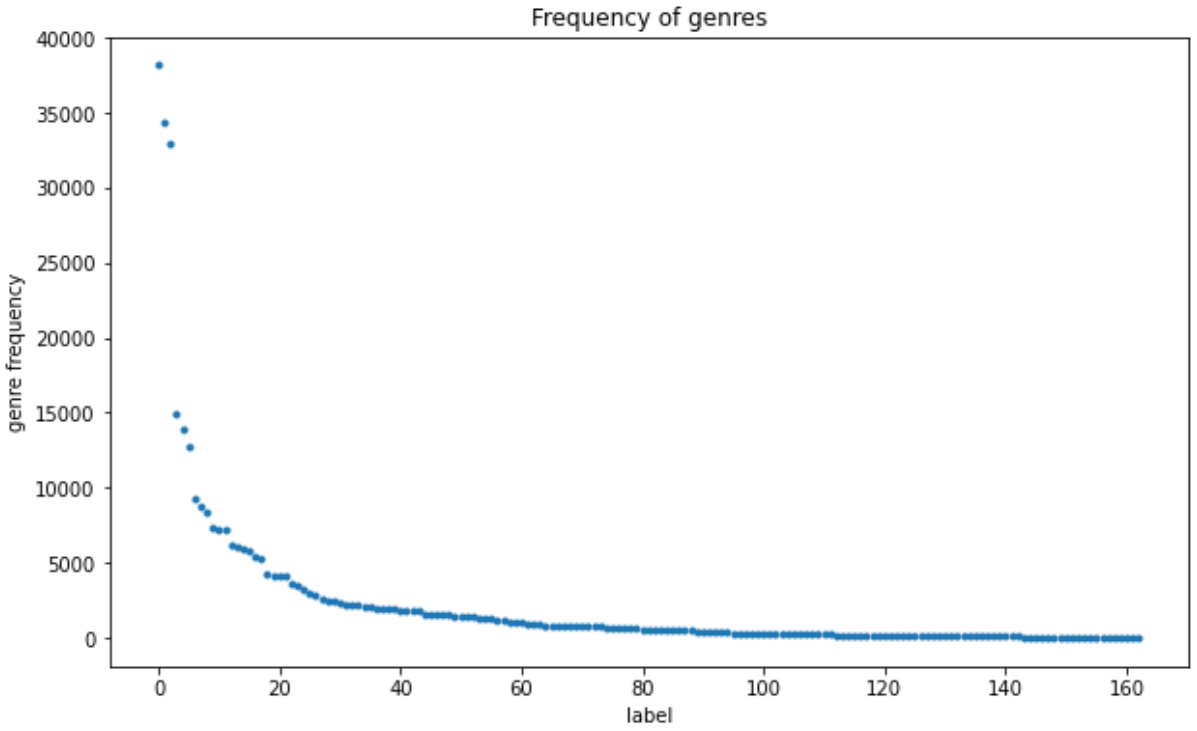
Even more useful is the use of Mel-frequency cepestral coefficients (MFCC). An MFCC is a representation of the power spectrum of a sound, it uses the following transformations of the signal:

- 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.

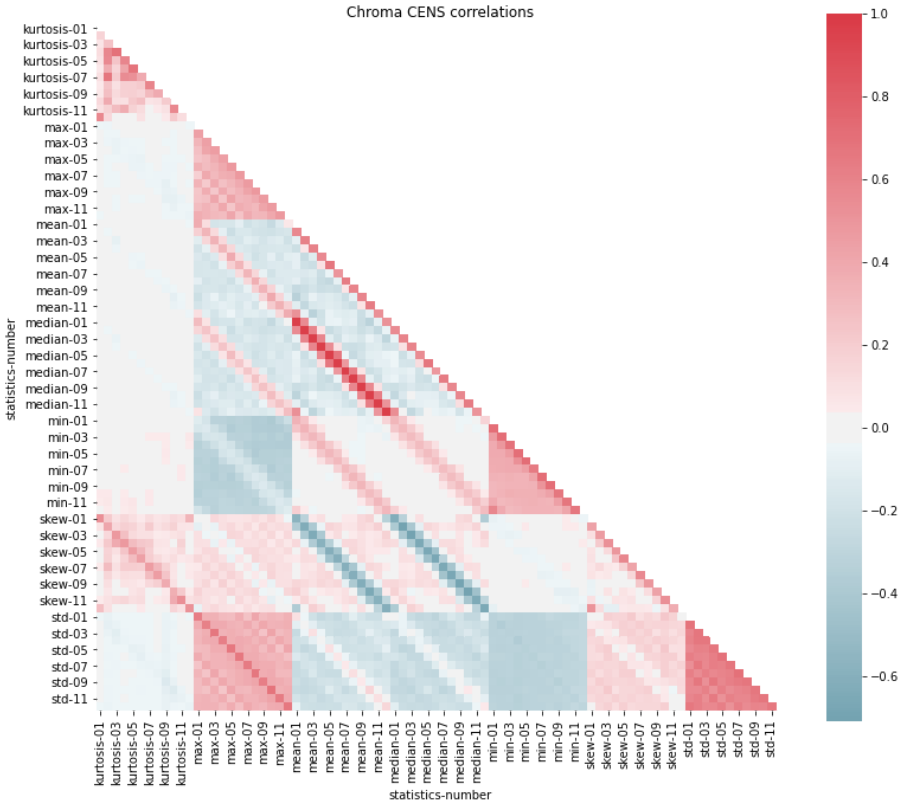


We can see that out of 163 genres, only about 25 have more than 3000 occurrences. Keep in mind that many tracks have multiple labels, both hierarchical and flat. For instance, [1, 12, 38, 90, 250] indicates:

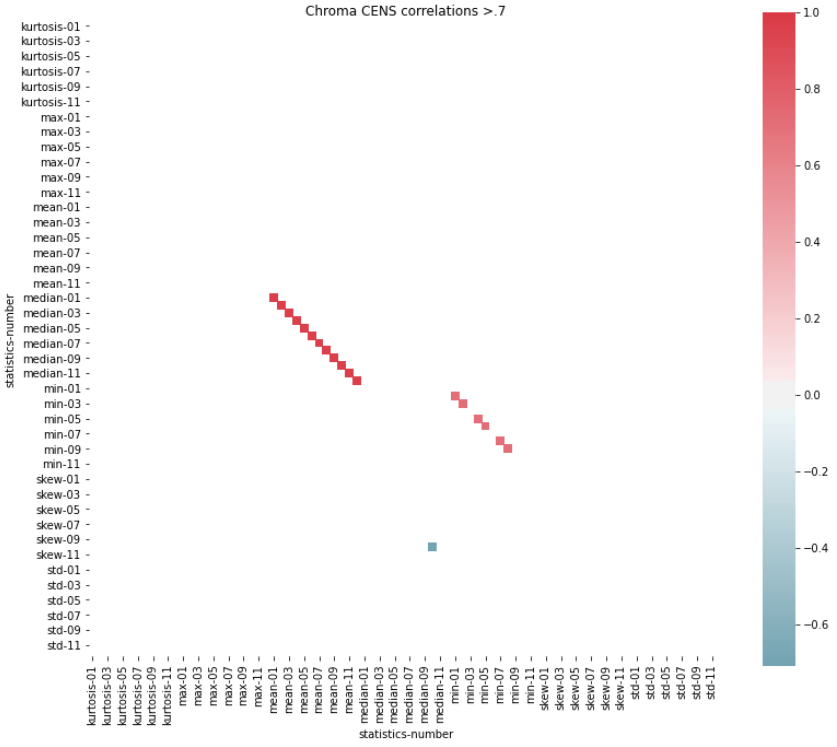
- 1 = Avant-Garde
- 12 = Rock
- 38 = Experimental (top-level and parent of both [1]Avant-Garde and [250]Improv)
- 90 = Sludge (child of [12]Rock)
- 250 = Improv

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.

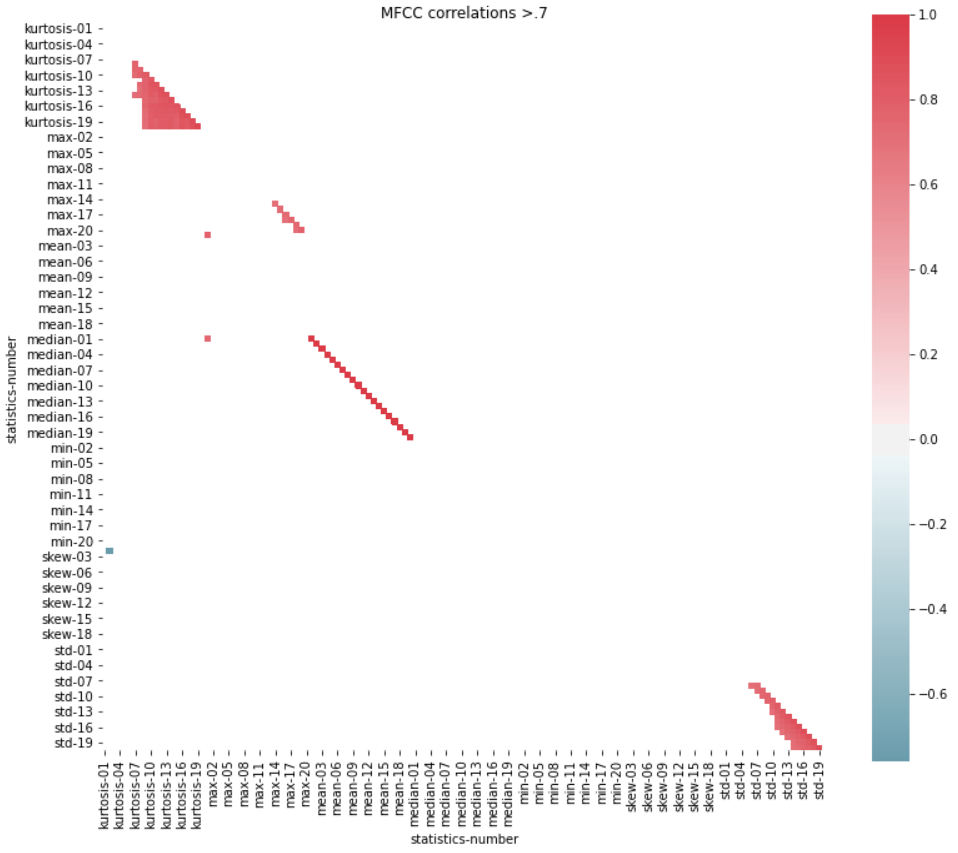


We can see some patterns of regularity. This indicates that the features of neighboring time slices tend to be highly correlated. This makes sense when you consider that a song will typically sound the same from one 23ms sample to the next. When we isolate just the highest positive and negative correlations:



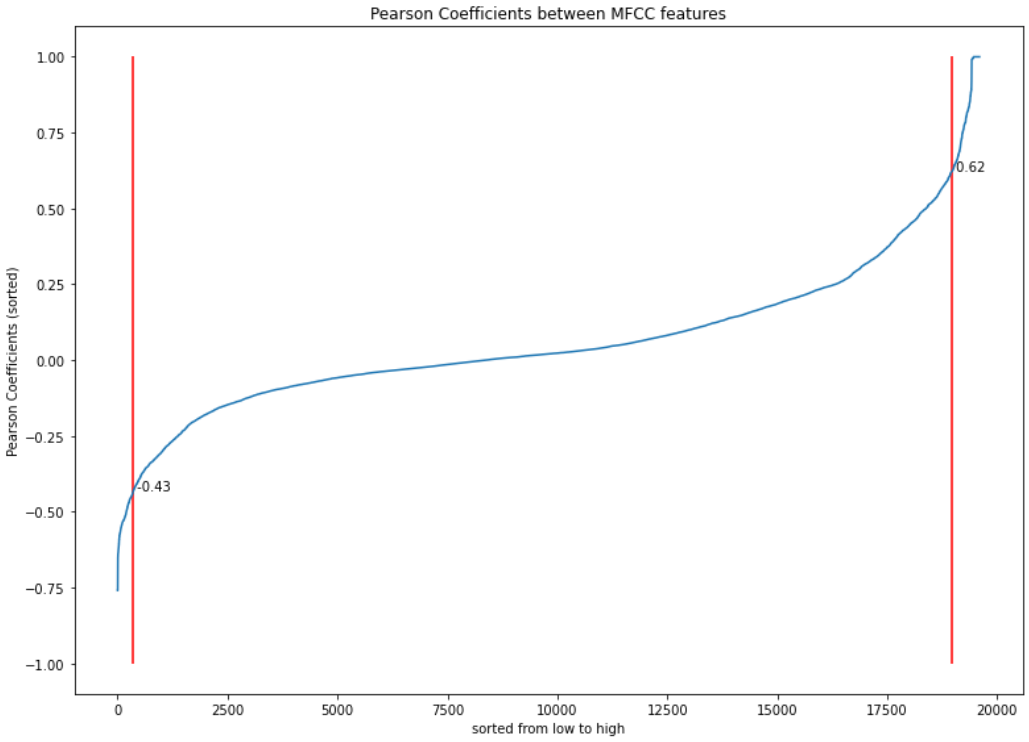
There are only a few pairs with a pearson r value of over .7 or under -.7. The one large negative value is the median/skew pair from sample 11 which has an r-value of -.867 and a p-value below .001.

As calculating the Mel-scale Frequency Cepestral 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



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.

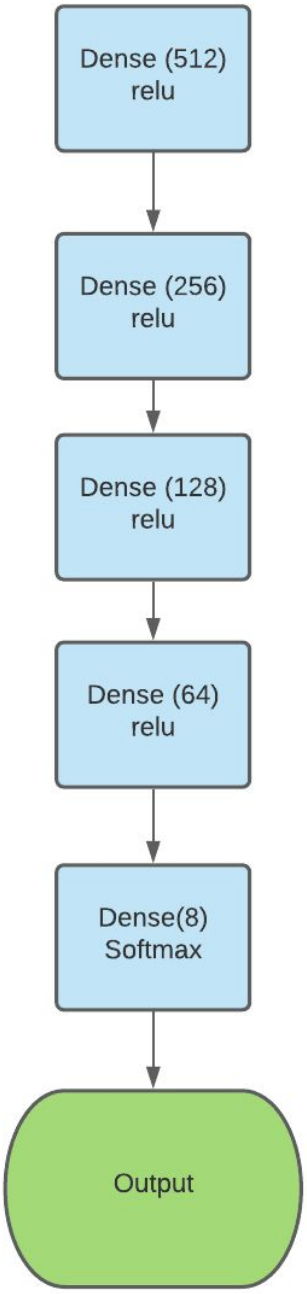
Logistic Regression

First, a logistic regression with default parameters achieved an overall accuracy of .21.

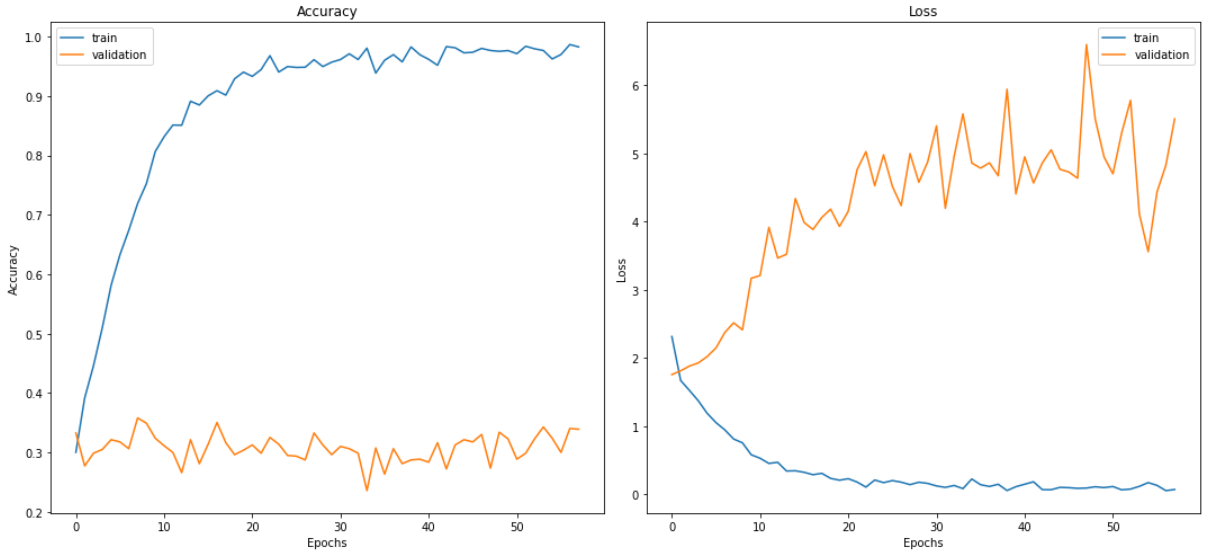
Classification Report (test set)

	precision	recall	f1-score	support
0	0.26	0.35	0.30	96
1	0.15	0.21	0.18	98
2	0.15	0.13	0.14	100
3	0.32	0.19	0.24	100
4	0.24	0.25	0.25	100
5	0.23	0.16	0.19	100
6	0.08	0.08	0.08	100
7	0.30	0.34	0.32	100
accuracy			0.21	794
macro avg	0.22	0.21	0.21	794
weighted avg	0.22	0.21	0.21	794

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


Dense (512)
relu
Dense (256)
relu
Dense (128)
relu
Dense (64)
relu
Dense(8)
Softmax
Output

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 (test set)
              precision    recall  f1-score   support

           0       0.34      0.50      0.40        96
           1       0.19      0.14      0.16        98
           2       0.19      0.13      0.15       100
           3       0.47      0.38      0.42       100
           4       0.30      0.39      0.34       100
           5       0.34      0.33      0.34       100
           6       0.18      0.19      0.19       100
           7       0.42      0.41      0.41       100

    accuracy                           0.31       794
   macro avg       0.30      0.31      0.30       794
weighted avg       0.30      0.31      0.30       794
```

Over the set of:

- 8 classes
- 1000 samples each class

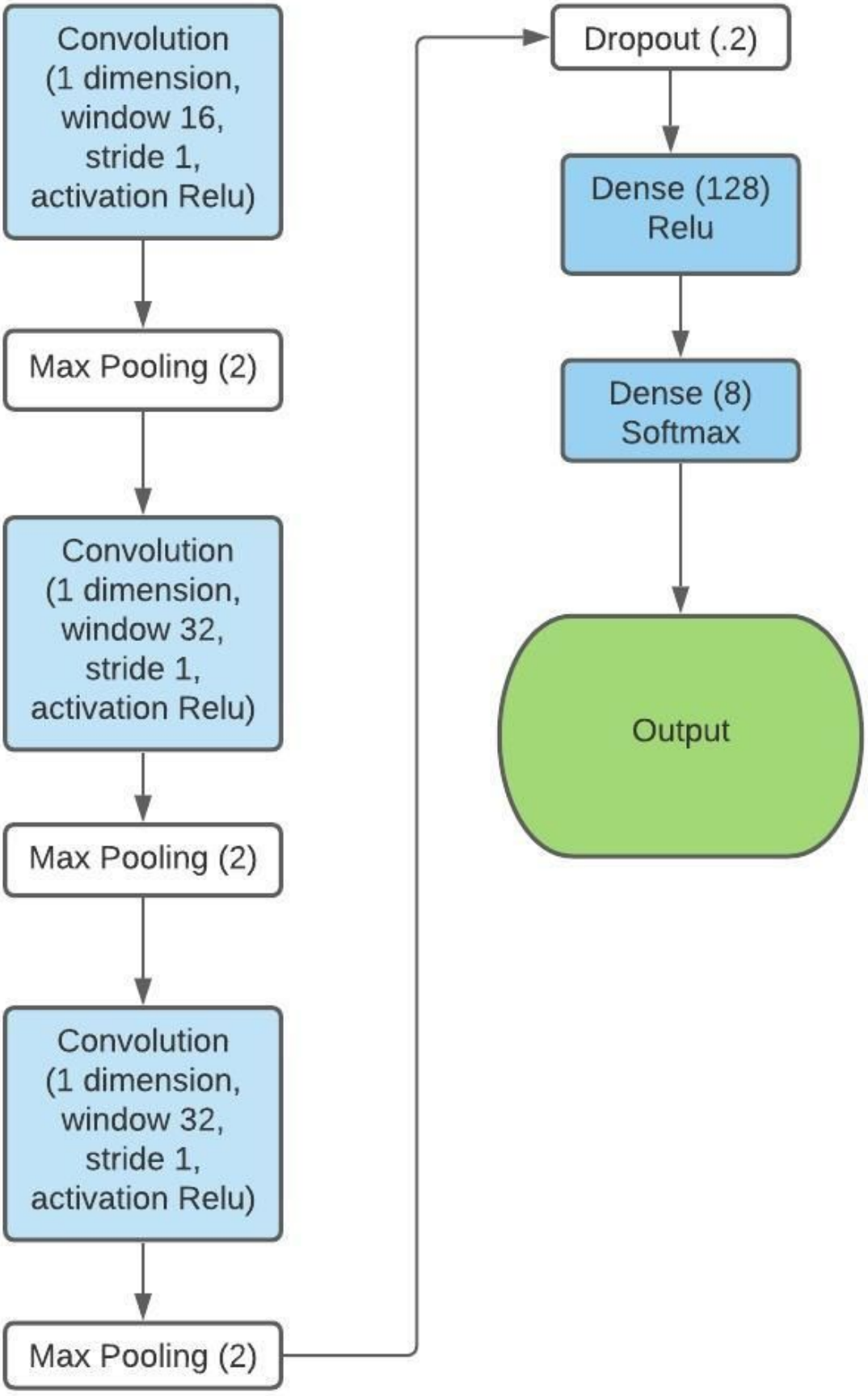
A baseline model of a fully connected neural net achieves better accuracy over the test set than a Logistic Regression:

Model	Accuracy(all classes)	Train Time
Logistic Regression	.22	2m 36s
Neural Net	.31	2m 24s

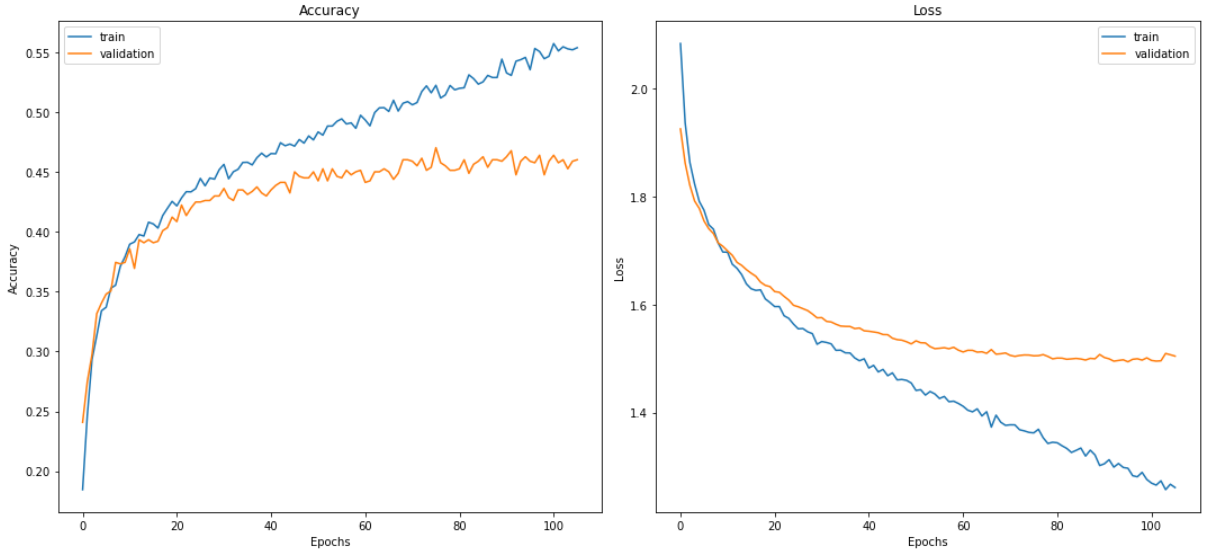
Extended Modeling

1. Convolutional Neural Net

Further modeling I tried 3 different neural net architectures. The first was a Convolutional Nerual Net (CNN) with the following architecture:


Convolution (1 dimension, window 16, stride 1, activation Relu)
Max Pooling (2)
Convolution (1 dimension, window 32, stride 1, activation Relu)
Max Pooling (2)
Convolution (1 dimension, window 32, stride 1, activation Relu)
Max Pooling (2)
Dropout (.2)
Dense (128) Relu
Dense (8) Softmax
Output

The model was trained with a batch size of 16, and with an early stopping condition of a less than .0001 accuracy improvement over 50 epochs. It stopped after 106 epochs, with the following accuracy and loss over the training and validation sets:



Applied to the test set, the results were:

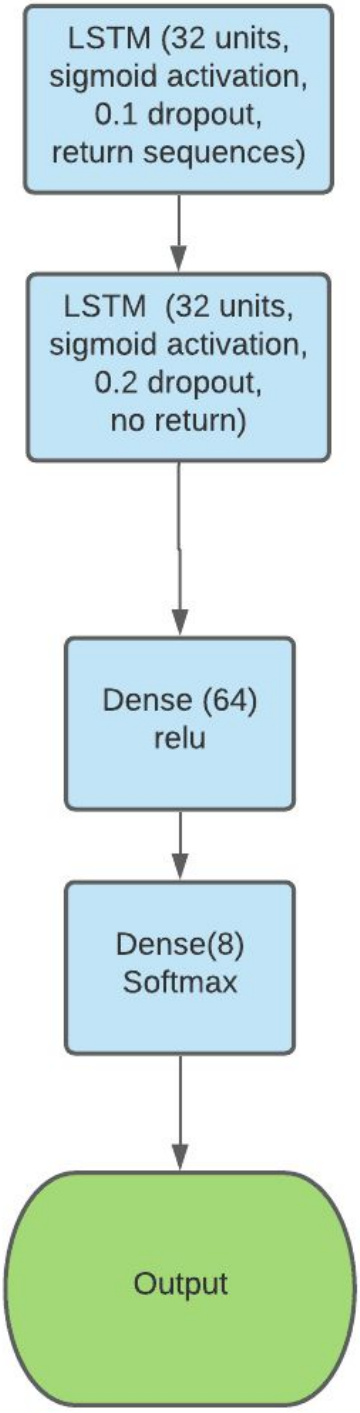
Classification Report (test set)

	precision	recall	f1-score	support
0	0.47	0.42	0.44	96
1	0.13	0.10	0.11	98
2	0.18	0.24	0.21	100
3	0.49	0.71	0.58	100
4	0.33	0.40	0.36	100
5	0.41	0.36	0.38	100
6	0.17	0.07	0.10	100
7	0.52	0.56	0.54	100
accuracy			0.36	794
macro avg	0.34	0.36	0.34	794
weighted avg	0.34	0.36	0.34	794

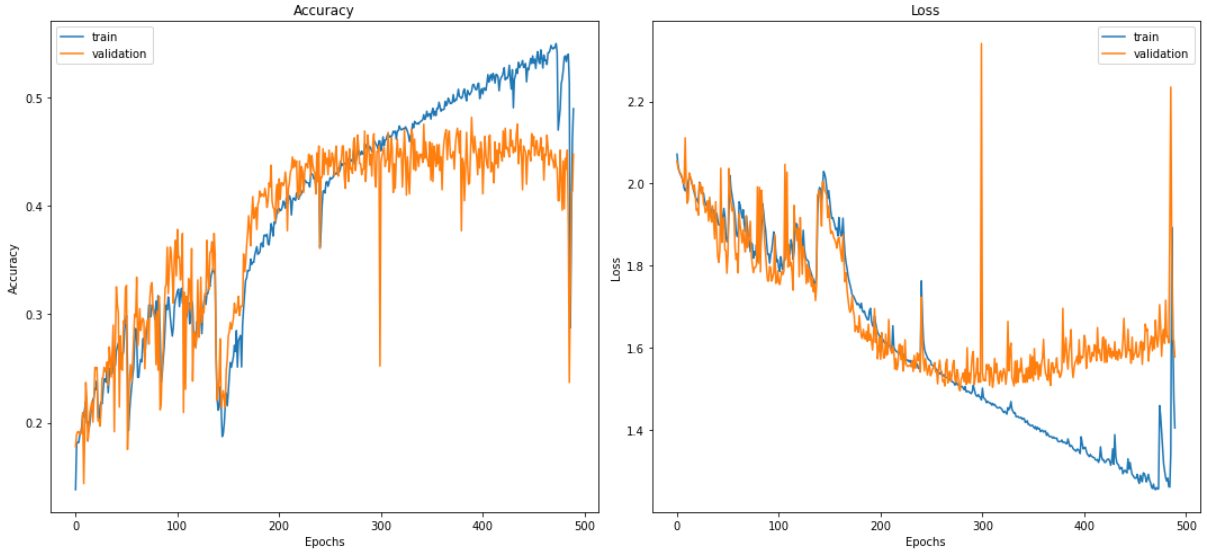
This is an improvement over the baseline model.

2. Long Short Term Memory

Next, because there are naturally features in music (rhythm, beat, repetitive musical figures) that correlate well to time series data, I implemented a Long Short Term Memory (LSTM) neural network. The architecture is as follows:



Training took longer than the CNN, stopping after 300 epochs. Each epoch also took longer.



Otherwise, the results were similar to those of the CNN:

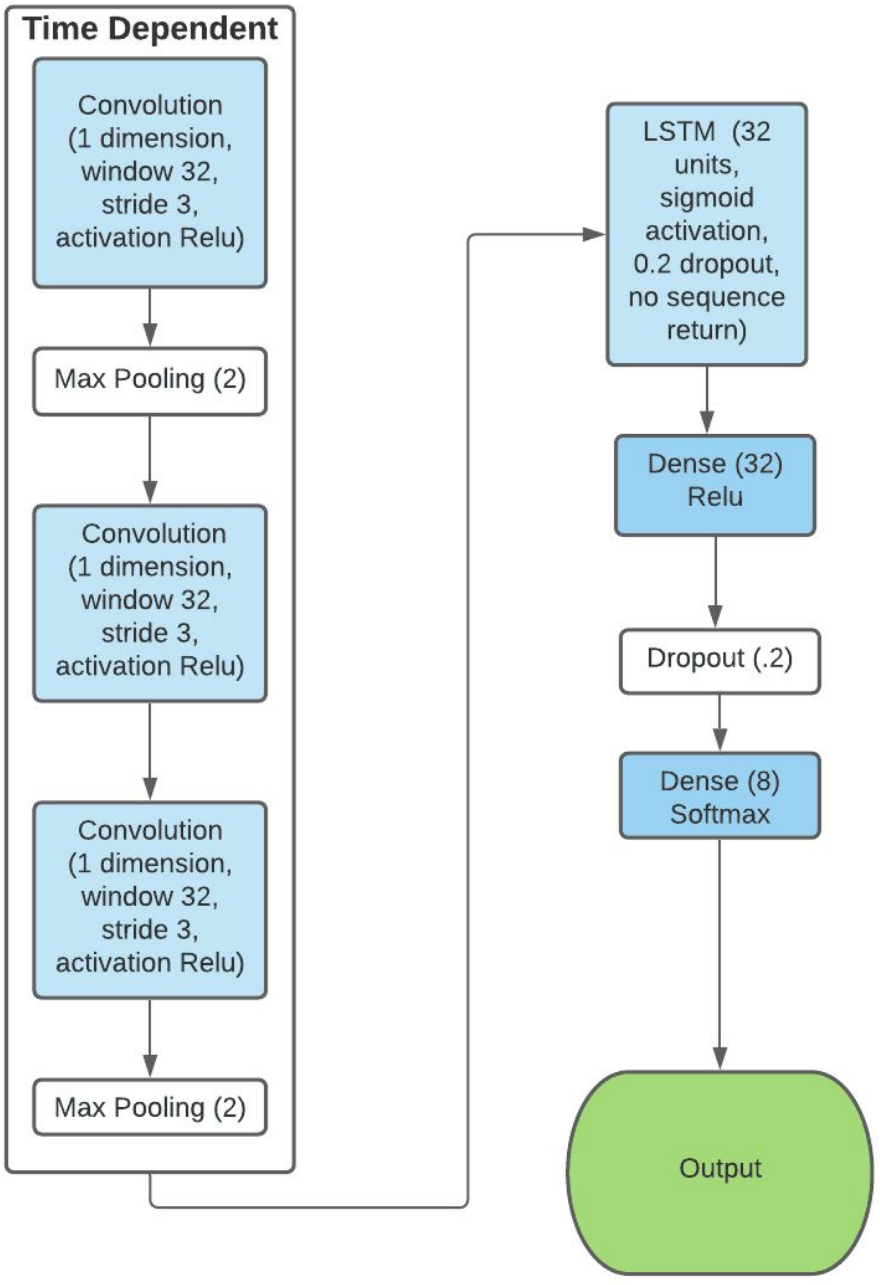
```
Classification Report (test set)
              precision    recall  f1-score   support

           0       0.50      0.27      0.35        96
           1       0.26      0.17      0.21        98
           2       0.22      0.23      0.23       100
           3       0.53      0.61      0.56       100
           4       0.33      0.33      0.33       100
           5       0.36      0.28      0.32       100
           6       0.27      0.31      0.29       100
           7       0.34      0.57      0.43       100

    accuracy                           0.35       794
   macro avg       0.35      0.35      0.34       794
weighted avg       0.35      0.35      0.34       794
```

In comparison, the CNN and LSTM achieved similar performance metrics, and performed similarly across the different classes. A notable difference is that the CNN performed much more quickly than the LSTM network.

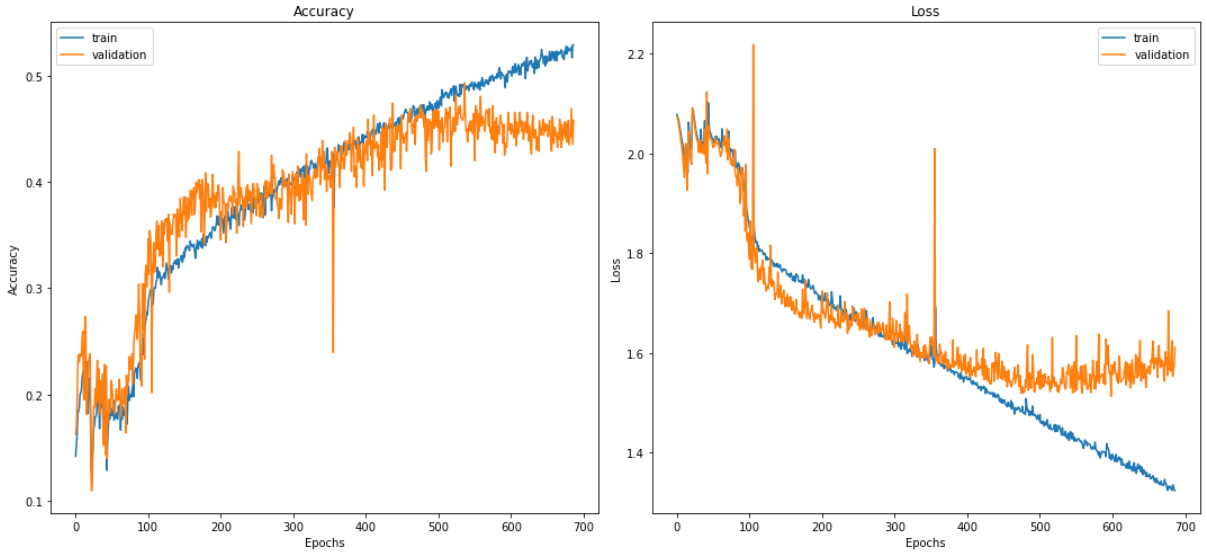
Model	Accuracy(all classes)	Train Time
CNN	.38	3m 30s
LSTM	.38	110m

Time Dependent CNN

The LSTM model performed slightly worse than the CNN, and was much more costly to run. The recognition of time dependent features in the data should still be useful for music classification. To this end I wrapped a CNN model similar to earlier in a Time-Dependent layer. The architecture is as follows:



With the same early stopping conditions as before (150 epochs with less than .0001 improvement in accuracy), this model ran for 687 epochs with the following training curve:




```
Classification Report (test set)
              precision    recall  f1-score   support

           0       0.42      0.51      0.46        96
           1       0.32      0.11      0.17        98
           2       0.34      0.48      0.40       100
           3       0.63      0.66      0.64       100
           4       0.38      0.47      0.42       100
           5       0.42      0.33      0.37       100
           6       0.34      0.31      0.32       100
           7       0.49      0.49      0.49       100

    accuracy                           0.42       794
   macro avg       0.42      0.42      0.41       794
weighted avg       0.42      0.42      0.41       794
```

Summary

This result is an improvement over both the CNN and LST models, although it takes longer to train than either.

Model	Accuracy(all classes)	Train Time
Baseline	.30	5m
CNN	.36	3m
LSTM	.35	90m
TD-CNN	.42	3h 20m

These results are better, but with each model the accuracy for the training and validation sets diverge when the validation accuracy reaches around .40 to .45. Let's look at the models' performance by class:

Class	Genre	F1 (CNN)	F1(LSTM)	F1(TD-CNN)
0	Electronic	.45	.35	.46
1	Experimental	.18	.21	.17
2	Folk	.19	.23	.40
3	Hip-Hop	.66	.56	.64
4	Instrumental	.33	.33	.42
5	International	.44	.32	.37
6	Pop	.22	.29	.32
7	Rock	.52	.43	.49

I think some of the differences in the models' performance over various classes can be explained by some of the features of the music. Hip-Hop is easier to classify because it has a combination of distinct rhythm and is the only genre with primarily spoken word rather than sung. Experimental music is hard to classify because the nature of being experimental is that instances typically don't have much in common.

Conclusions and Further Work

The TD-CNN model achieved the best performance metrics. It did take the longest, but still not a prohibitive amount of time. I think working with this model has potential for further improvement.

A few avenues of improvement come readily to mind:

- Feature selection. All the modelling here was based on the extracted MFCC features. Incorporating more features such as tempograms or the raw audio data may improve results.

- Larger dataset. Only 8000 samples were used. The FMA database contains 106,000, approximately 56,000 of which are in the 8 classes used here. Training over a larger sample should yield some improvement.

Recommendations for the Client

- Further development of the model is recommended before deployment. With only a 44% overall accuracy rate, over half of the time songs will be miscategorized. While Hip-hop will be correctly identified nearly ⅔ of the time, pop music will be correctly identified less than ⅓ of the time.
- All 3 models are strong and weak on the same classes, e.g. Hip-hop is easiest, Experimental is hardest. I think this is because of how well the features extracted represent these classes.
- Classification is only on MFCCs, which are representative of the timbre of the sounds. This seems to be insufficient for genre classification. Timbre and therefore MFCCs are a good proxy for e.g. instrumentation (which is certainly relevant) but it can be confounded by production techniques, which can be similar across genres.
- As such, adding features which capture other musical elements such as tempo and rhythm should improve performance.
- The TD-CNN model is ~18% more accurate than the CNN model but 70 times slower. We should see if classification using the CNN model over the proposed expanded feature set shows sufficient improvement and if not then continue work with the TD-CNN model.

Resources

Free Music Archive
Librosa
Tensorflow
Tensorflow IO