University of California, Merced
Computer Science and Engineering 168 Distributed Software Systems
Music Genre Classification
Apsara Fite (100321584), Andy Hernandez (100343512), Nathan Vasquez (10034441), Xavier Ybarra (100342148)

Final Report

Abstract

To demonstrate the capabilities of TensorGlow in contexts outside image detection, we developed a neural network for classifying music by genre. We trained it on a dataset of music retrieved from Kaggle, consisting of 100 songs per genre. However, we felt that 100 songs was insufficient, so we supplemented the dataset with another 100 additional songs for each genre. The raw .wav files were sampled and converted into MFCC data for trainability using Librosa. An MFCC is essentially an iteration of a Fourier transform intended for sound processing. From there, we split the data into a training set and a testing set for our model at a ratio of 80-20, respectively.

The overall shape of our sequential model consists of 4 ReLU layers, followed by a softmax for output. We then fit the model to the training set in batches of 32 and set the number of epochs to 30. We determined that 30 epochs would be the best fit for our model by evaluating loss values, finding that the model doesn't become more accurate past around 30 epochs.

As a result, we had a model that could correctly label the genre of the training set 64.8% of the time, with varying accuracy by genre. Classical is the genre that is usually correctly identified by the model with an accuracy of 92.7%. Blues is somewhat more difficult for the model to predict with an accuracy of only 44.0%. That being said, this is still over three times more accurate than if the model were to randomly guess the genre (⅛ chance, 12.5%).

1 Knowledge Attained

By creating this music classifying model, the group learned how to use the Librosa library, use the TensorFlow library, train a model, and interpret the results.

The first step was learning how to get our idea into a working model. This included finding a way to numerically represent music that could then be processed by the model. Librosa conveniently has a function that allows us to convert raw sound files to MFCC, so we followed documentation to do so (source 4). MFCC takes sound frequency data as input and for each sample interval determines the strengths of each frequency range using a fourier transformation. We chose to write the MFCC data to a JSON file to load from later.

We also learned how to build sequential models, as well as how the Keras Adam optimizer enables a model to learn. Finally, we learned how to represent prediction data as confusion matrices.

2 Training

We acquired wav files from the GTZAN dataset available on Kaggle, using 7 genres of 100 songs each: blues, classical, disco, jazz, metal, pop, and reggae. We then supplemented the dataset with 100 additional songs per genre, as well as adding an additional genre: phonk. Each .wav file was sampled to 15 seconds in length and transformed to MFCC data using Librosa. The data was then split into a training and validation set at a ratio of 80-20.

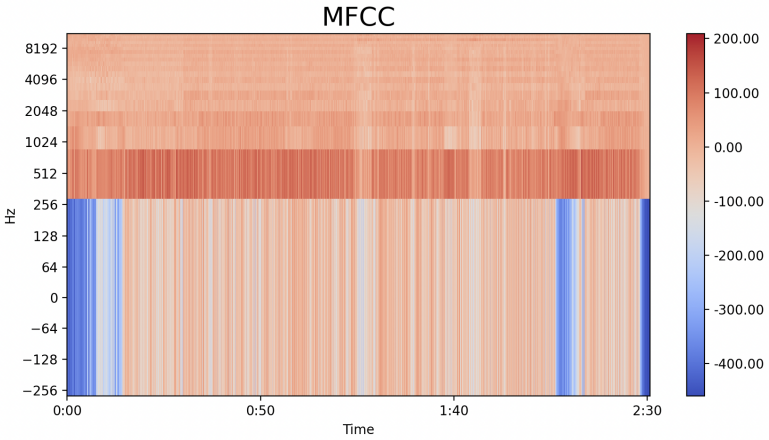

Fig. 1: Visualization of a phonk song after conversion to MFCC.

Our neural network was designed as a Keras Sequential Model. The overall shape of our model consists of 4 ReLU layers, followed by a softmax layer for output. We then fit the model to the training set in batches of 32 and set the number of epochs to 30.

The number of epochs we decided on was determined by observing loss values after each epoch. Our model rarely improved in accuracy or loss past 30 epochs, so we decided to cap it there to prevent overfitting. A similar method was used to determine that a batch size of 32 was best for our model: by observing accuracy and loss after a few trial fittings.

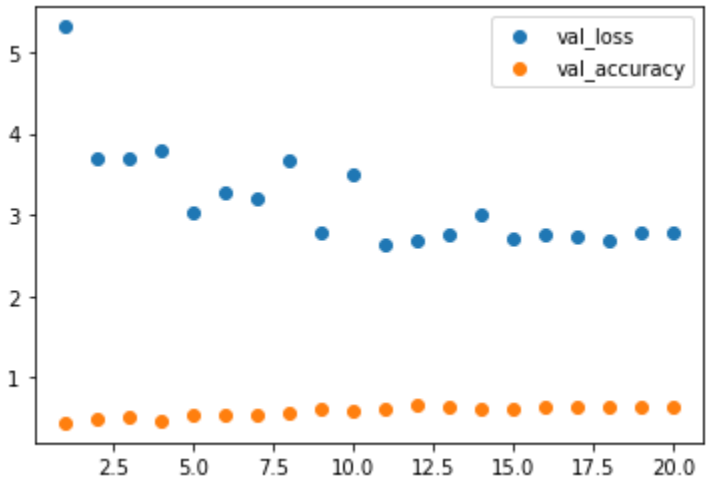

Fig. 2: Loss and Accuracy

Additionally, we also implemented an early-stopping function for the training model to further ensure that overfitting would not occur. If the validation loss and accuracy do not improve eight epochs past the current best, we restore the model to the state where it had the highest accuracy. In the above case, that was after the 12th epoch.

4 Evaluation

The performance metrics that we measured are as follows: mean average precision, mean average recall, loss, and time to train.

4.1 Workload Selection

The model used was trained with 100 songs per classification and 8 total classifications. This is considered a high amount but each sample is only 20 seconds. The model would perhaps be more accurate if the sample was the entire length of the shortest song in the dataset. The reason that each song has to be trimmed to the same length is for the same reason if the set was trained on pictures all the pictures would have to be the same size.

4.2 Platform

The group used 4 different computers to run the program. However, the platform breakdown will only describe one of the said computers. In this case it was an HP Omen 15 with a Ryzen 7 processor and 16GB of ram. The processor has 8 cores and runs at 3.6GHz and the local solid state drive with 1TB of storage. The code itself was run in a Jupyter notebook with all the files, including the music files, located on the local drive.

4.3 Metrics

Precision is given by $\frac{(true\ positive)}{(true\ positives)+(false\ positives)}$ and recall is given by $\frac{(true\ positive)}{(true\ positives)+(false\ negatives)}$. The confusion matrix can be found in the Results section. If given a target song with a set genre, precision would be able to tell us the proportion that the song would be categorized properly. By changing the false positive to a false negative, the recall equation is obtained. Recall is the proportion of correct music categorizations over all attempts. Loss is the measurement between the model and the training data. In other words, a representation of how well our model did in categorizing the music correctly.

4.4 Results

Our trained model was used to predict the genre of our testing set. The resulting predictions are represented below as confusion matrices. The first is the final guess

for each song in the testset, and the second is the average confidence the model has in its prediction of each song's genre in the testset.

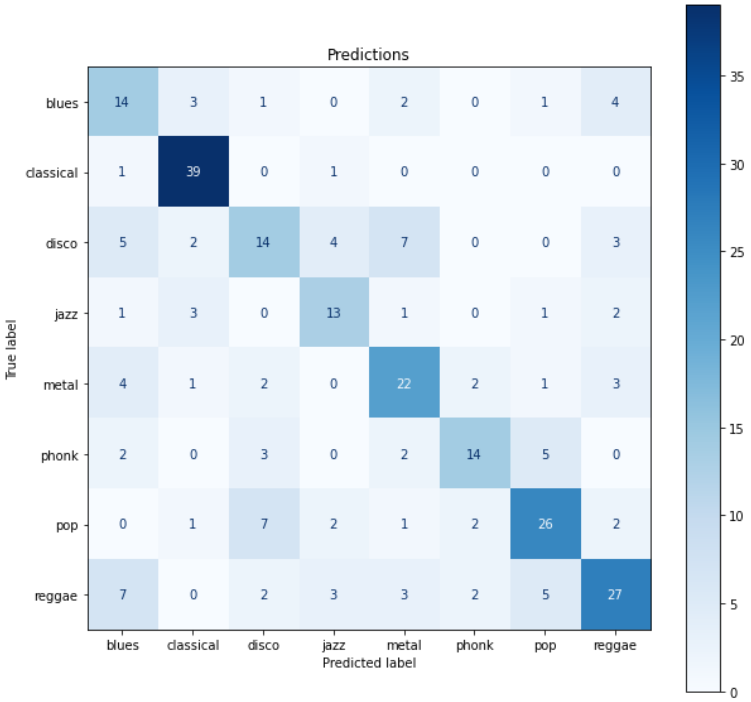

Fig. 3: Prediction Table

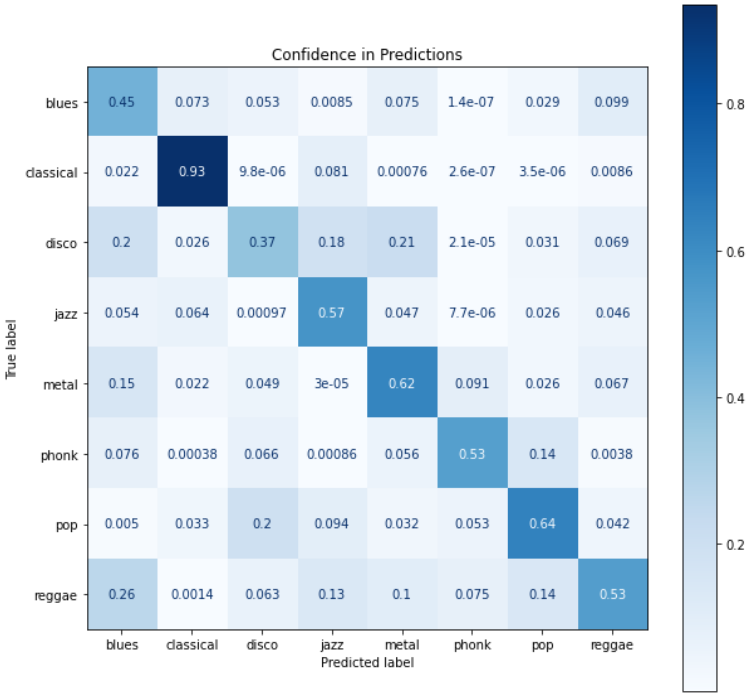

Fig. 4: Confidence in prediction table

Theoretically, a computer would be estimated to have an average recall of 12.5% if it guessed the genres at random. The resulting trained sequential model had a recall of 64.8%, so it is safe to say the model has successfully been trained to predict the genre of a given song and is not simply guessing. See the below table for the recall and precision per genre.

blues	14	3	1	0	2	0	1	4
classical	1	39	0	1	0	0	0	0
disco	5	2	14	4	7	0	0	3
jazz	1	3	0	13	1	0	1	2
metal	4	1	2	0	22	2	1	3
phonk	2	0	3	0	2	14	5	0
pop	0	1	7	2	1	2	26	2
reggae	7	0	2	3	3	2	5	27
	blues	classical	disco	jazz	metal	phonk	pop	reggae
precision	0.4117647059	0.7959183673	0.4827586207	0.5652173913	0.5789473684	0.7	0.6666666667	0.6585365854
recall	0.56	0.9512195122	0.4	0.619047619	0.6285714286	0.5384615385	0.6341463415	0.5510204082

Fig. 5: Resulting precision and recall

4.5 Observations

Some genres had a much higher recall rate than others. This may be due to the nature of the individual genres, as some genres share characteristics of one another. For example, songs from the classical and metal genres have very distinctive, unique qualities that aren't shared with any of the other genres. However, blues, jazz, and reggae can be confused for one another due to overlapping musical characteristics.

5 Conclusion

In conclusion music genre classification ended up being very similar to object classification. Some objects are very difficult for models to differentiate between and some music genres would be as well. We would be interested in extending this project to music generation, but that was not possible in the timeframe given for this project. It would be more similar to text or art generation than image classification in complexity.

References

1. Lyons. (2013, March 13). Mel Frequency Cepstral Coefficient (MFCC) tutorial. Practical Cryptography. http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/
2. Olteanu, James, O'Hare, & Lei. (2020, March 24). GTZAN Dataset - Music Genre Classification. Kaggle. https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification TensorFlow. (n.d.).
3. TensorFlow Libraries & Extensions. https://www.tensorflow.org/resources/libraries-extensions