STAT 4984 Final Project - Music Genre Classification

Loveish Sarolia[*]
CMDA and Economics Student
Virginia Tech
`loveish@vt.edu`

Abstract

Music genre classification labels are useful when organizing music into various bins depending on sound profile. Humans have created bins based on characteristics of music to separate music into these differing categories. Some of the determinants of genre are rhythm, harmony, vocals, and other individual components of songs that are combined to make a song meaningful. As the number of artists and songs grows, the need for an accurate classification method for music genres becomes increasingly important. Humans are used to validate music genre labels assigned by these automated methods, but this method of validating is not sustainable as the field continues to grow. Large music sharing platforms such as Spotify, Apple Music, and Soundcloud rely heavily on the innovation in artificial intelligence and machine learning to cut costs and save time. Deep learning can make the process easier by limiting human error and providing an unbiased classification. Currently, some of the methods most commonly being used for this task are support vector machine and k-nearest neighbors; however, deep learning can utilize feature extraction and use more complex datasets to more accurately classify genres.

https://github.com/LoveishSarolia/STAT4984_DeepLearning_FinalProject

1 Introduction

Music genre classification is an interesting problem due to the recent prevalence of "beat swaps", where the instrumental of the music changes in the middle to provide a sort of emphasis to the artist's lyrics. I would like to determine if the music genre classification model I implement will be able to accurately classify the genre of a song based on the prevalent beat. This classification method is important as it can help providers such as Spotify or Apple Music more accurately suggest music to users that fits their preferred sound profile. To accomplish this task, I plan to utilize Feature Extraction to separate the vocals from the instrumentals to give the model the bass and treble frequencies while discarding noise. One of these resulting graphs will be spectrograms which serves as a graphical visualization of the frequencies of a signal, in this case it will be a snippet of music. The other graph will be a wavelet defined by a mathematical function that can flatten the number of dimensions to accurately plot the audio's signature from the .wav file for time-frequency graphing as a secondary source. These graphical representations will be inputted into a feed forward neural network (FFNN). The FFNN will determine a pattern between specific genres and the inputted wavelet/spectrogram. The goal of an accurate FFNN that can classify music genres would be to expand into multi-label genre classification and assigning music with "beat swaps" that are more difficult to categorize, even by human labelers.

[*]Senior Data Science and Economics Student at Virginia Tech

2 Related work

A paper by Oramas et al. 2018 [2] created a convolutional neural network to classify music into their respective genres while aiming to make the model more accurate than a series of humans classifying the same data. Their results showed a fairly similar grouping between the CNN and the participants; however, they were both not very accurate with an average of 70% validation accuracy. Upon inspecting the three CNN architectures they created, I noticed the second CNN approach with an input size of 384 seemed to be the best performing. Based on that, I decided to increase my input layer to 512 in my dense layer to allow more features rather than the dimension reduction they performed. Their dataset also seemed to have too many labels, resulting in a less accurate classifying model.

The next paper studied was by Dias et al. 2022 [1] where they wanted to test various machine learning approaches to classify music into either single or multi-genre labels. They decided to use the GTZAN dataset which served as a helpful set of instructions for the pre-processing stages of the data. The limitation of their analysis found CNN to be sub-80% accurate when testing. My goal was to use a similar architecture to the feed-forward neural network they implemented to make my validation accuracy higher. Additionally, their test loss seemed to be very inconsistent over the epochs. I decided to increase the number of epochs dramatically while also decreasing the test size per iteration.

The third paper I decided to use as a baseline was by Pelchat and Gelowitz (2020) [3] where they went through a detailed explanation of the mathematical background of music genre classification at every stage. I used that as a guide for my methodology. They discussed the formulas for converting the .wav files into spectrograms and wavelets which I used as I implemented the functions. Additionally, their architectures referenced from literature reviews served as the primary inspirations for my FFNN architecture and it layout. However, the number of hyper-parameters they decided to use seemed to be excessive for the dataset and only seemed to increase computation time.

3 Dataset and Features

The dataset that was be used to train/validate/test the music genre classification model was the GTZAN dataset. It consists of 1000 audio files that are 30-seconds long, divided into 10 genres. The genres included in the GTZAN dataset were blues, classical, country, disco, hiphop, jazz, metal, pop, reggae, and rock. The format of each of the audio files is 22050Hz Mono 16-bit (.wav). The data is not split by default so I will split it into a 67-33 division with 67% of the data being used for the training and 33% of the data being used for testing/validating. The music files needed to be converted into spectrograms and wavelets (which are visual representations of the 3 second feature extracted .wav files). Initially, the dataset created and utilized by Oramas et al. 2018 [2] was looked into as a starting point for genre classification; however, upon further inspection, their data seemed to be skewed towards pop and rock which did not give a fair distribution to the remaining genres. Additionally, their genre labels were more specific than the GTZAN labels so the difference between rock and alternative rock or jazz and European jazz would have given the model a much more difficult time when predicting labels.

4 Methods

For this project, a feed-forward neural network (FFNN) was used to extract features from the music snippet spectrograms/wavelets and identify which genre the snippet most closely aligns with. Typically, a FFNN would take a flattened input vector as its input; however, in music genre classification, audio signals that have been converted into spectrograms and wavelets are used. A FFNN consists of several layers of fully connected nodes, with each node connected to every node in the previous and following layers. The input layer receives the flattened spectrograms and wavelets then passes them to the first hidden layer. The hidden layers process this input by applying a non-linear transformation to the spectrogram/wavelet which extracts features such as frequency over time or rhythmic repetition in the waveforms. The output layer will receive the outputs from the previous layers and use those in conjunction with a softmax activation function to produce a probability distribution for the audio signals that will classify them to one of the ten genre labels.

For the architecture used, the first layer was a fully connected layer that contained 512 neurons along with a ReLU activation function. The layer takes in a tensor with shape (batch_size x X_train.shape[1]). Next, a dropout layer with a 20% dropout rate for the neurons. It is used to help prevent overfitting. Next is another fully connected layer with 256 neurons and a ReLU activation function. It is accompanied by another dropout layer with a 20% dropout rate. This is continued in neuron decrements of 128, 64 and is finalized with a fully connected layer containing 10 neurons and a softmax activation function. This layer will serve as the classification layer that will group the music into their respective genres.

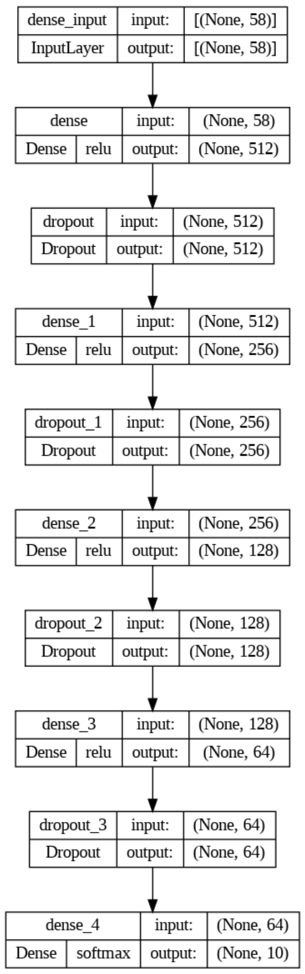
dense_input | InputLayer | input: [(None, 58)] | output: [(None, 58)]
dense | Dense | relu | input: (None, 58) | output: (None, 512)
dropout | Dropout | input: (None, 512) | output: (None, 512)
dense_1 | Dense | relu | input: (None, 512) | output: (None, 256)
dropout_1 | Dropout | input: (None, 256) | output: (None, 256)
dense_2 | Dense | relu | input: (None, 256) | output: (None, 128)
dropout_2 | Dropout | input: (None, 128) | output: (None, 128)
dense_3 | Dense | relu | input: (None, 128) | output: (None, 64)
dropout_3 | Dropout | input: (None, 64) | output: (None, 64)
dense_4 | Dense | softmax | input: (None, 64) | output: (None, 10)

Figure 1: Model Architecture Flowchart

The spectrograms were plotted by using feature extraction to gather the frequency of each 30 second snippet of the .wav files to plot a spectral image representation of each song. These spectrograms serve as a visual representation of the frequency content of the audio snippets over time. These can help identify patterns in the audio snippets whic can correspond to determinants asuch as rhythm or harmony that can help more easily classify each song based on its genre. Spectrograms can also help overcome one of the most common issues in music genre classification which is when two or more of these characteristics overlap, causing confusion between what genre is associated.

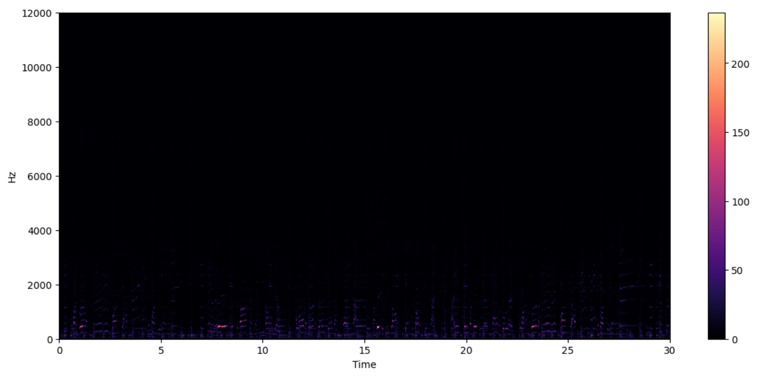

Figure 2: Randomly Generated Spectrogram

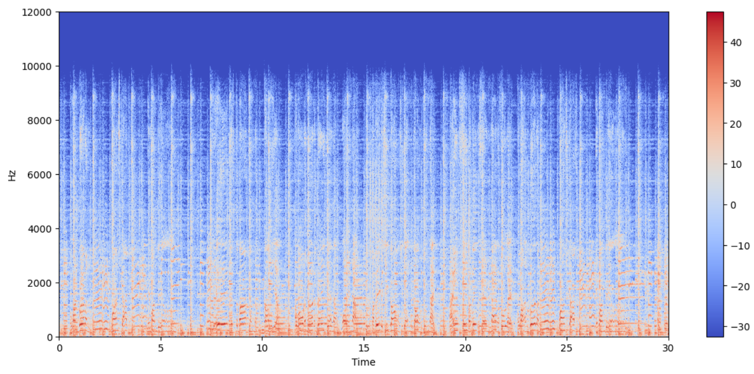

Figure 3: Randomly Generated Spectrogram - Inverse for Visibility

The wavelets were graphed using the continuous wavelet transforms equation

$$\psi(t) = 2sinc(2t) - sinc(t) = \frac{\sin(2\pi t) - \sin(\pi t)}{\pi t}$$

where they can perform feature extraction to scale frequencies over time and measured by mean, variance, etc. Wavelets can be used with spectrograms to provide a secondary point of analysis for the classification tool.

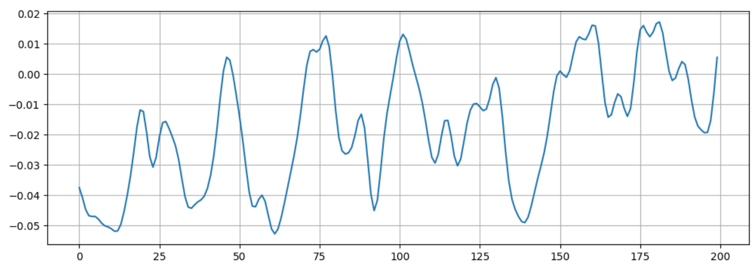

Figure 4: Randomly Generated Wavelet

The base architecture for the music genre classification algorithm is based on a FFNN. The specific dimensions were created from a combination of the papers and sources deemed relevant. The loss function that was used is the sparse categorical cross-entropy loss function.

$$y\_true = y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

$$y\_pred = \widehat{y} = \begin{bmatrix} \widehat{y_1} \\ \widehat{y_2} \\ \widehat{y_3} \end{bmatrix}$$

$$CE = -\sum_{i=1}^{i=N} y\_true_i \cdot \log(y\_pred_i)$$

$$CE = -\sum_{i=1}^{i=N} y_i \cdot \log(\widehat{y_i})$$

$$\implies CE = -\left[y_1 \cdot \log(\widehat{y_1}) + y_2 \cdot \log(\widehat{y_2}) + y_3 \cdot \log(\widehat{y_3})\right]$$

To implement all of the aforementioned features of the neural network, Keras will be used as it makes hyper-parameter tuning simpler and has more documentation supporting its uses in music genre classification than Tensorflow or PyTorch. Keras solved many of the issues initially incurred when trying to convert .wav files into spectrograms using spectral rolloff. This issue did not seem to be present when looking at other similar code in Keras.

5 Results

For the convolutional neural network architecture, the hyperparameters used were the number of neurons for each fully connected layer, the activation function assigned to each of the layers, the dropout rate between layers, and the input shape passed in. The number of neurons for the layers were (512, 256, 128, 64, 10), respectively. The activation functions used were ReLU for all of the fully connected layers except the final one which utilized a softmax activation function to produce a probability distribution for each of the music genre labels. The dropout rate over the entire architecture was 0.2 which means 20% of the neurons are randomly dropped during training between each layer, which can be used to prevent overfitting. The input shape for the first layer is the number of columns in the X_train variable. As specified previously, these hyperparameters were easier to tune in Keras which is part of the reason it is being utilized. The metrics for measuring model accuracy were validation accuracy and loss. These two methods were used as they were easy to implement in Keras. Initially, F1-score was considered to validate model accuracy but it was soon scrapped as it gave similar enough results to the validation accuracy gathered from the model while being more difficult to implement. Among most of the papers and articles implementing music genre classification using deep learning, they used validation accuracy and loss to show overall model accuracy.

The FFNN implemented seemed to have a top out accuracy of 90.35% as shown in 5, which was much higher than the papers examined. Likely, most of this higher prediction accuracy was due to the smaller iterative data fed in while maintaining a larger epoch size to more fully train the model.

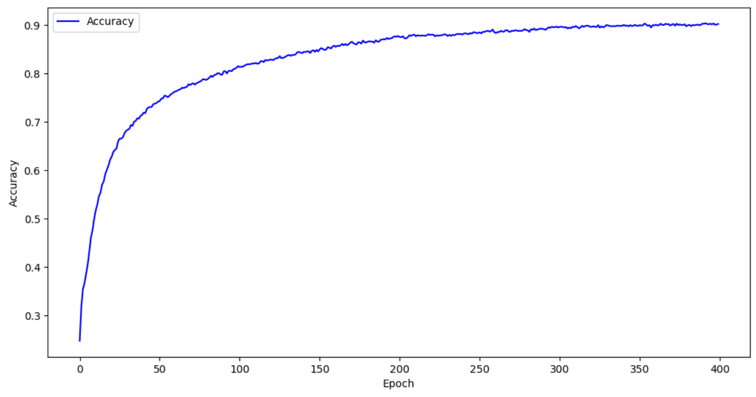

Figure 5: Model Accuracy Over 400 Epochs

The loss also seemed to converge at a much lower value than the other papers examined 6. This assures that the model was correctly implemented and the predicted output labels were close to the actual genre labels.

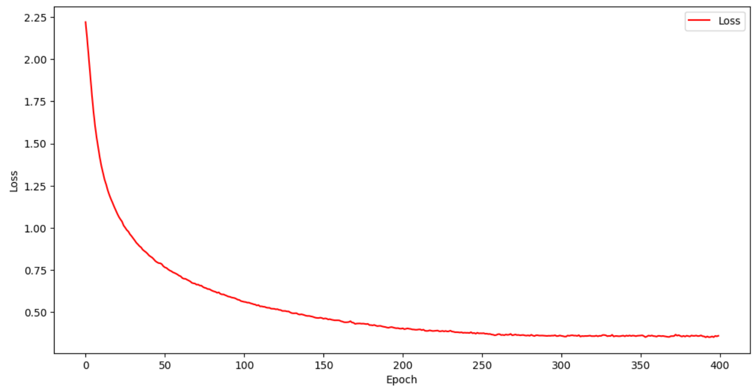

Figure 6: Model Loss Over 400 Epochs

To validate which genres were most commonly misclassified, I decided to use a confusion matrix. The confusion matrix evaluates the performance of the model by graphing the number of correct and incorrect predictions, along with which genre it was mistaken for. The confusion matrix also showed the precision between genre labels and gives the baseline information used to calculate other metrics such as F1-score. To interpret the confusion matrix, the row values correspond to the actual number of samples that belong to a specific genre while the column values correspond to the predicted genre labels. This means that for each label-label match, those are true positives. For the values that are on the row axis but not the same label, those are the values that were misclassified for that specific category. The matrix shows that rock needs to be improved upon to make the model as a whole more accurate. The most commonly misclassified label that the model misinterprets instead of rock is country. Improving this classification rate could be accomplished by providing the model with more training models of rock that are similar to country in wavelet and spectrogram structure.

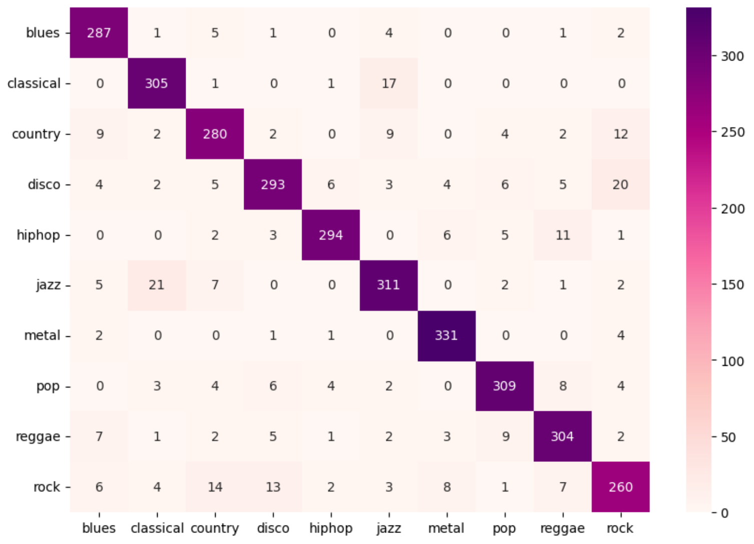

Figure 7: Confusion Matrix

6 Conclusion and Discussion

In this project, a music genre classifier was created that utilized the best aspects of currently known architectures while implementing changes that increased overall accuracy. A convolutional neural network was applied as the model that classified the GTZAN dataset, split into 67% training and 33% validation. The genres included in the GTZAN dataset were blues, classical, country, disco, hiphop, jazz, metal, pop, reggae, and rock. Overall, the feed forward has a 90.35% accuracy over 400 epochs, which seems to be at or above the most accurate CNNs or FFNNs found in papers and articles implementing the music genre classification using deep learning. This distribution of training and testing data per epoch seems to be the most computationally efficient in terms of time and accuracy. Other splits/allocations seem to take much more time, which is not feasible in larger scale classification applications. The model created could be used as a starting point for multi-genre classification; however, in order for that to happen, the dataset needs to be expanded and re-labeled. This is the next logical step before continuing to tweak hyperparameters to improve model efficiency. If more time and more collaborators were feasible then I would want to start inputting real snippets of music that have "beat swaps" to determine how the model would perform in a more complex case.

References

[1] Jessica Dias, Vaishak Pillai, Hrutvik Deshmukh, and Ashok Shah. Music genre classification & recommendation system using cnn. *Available at SSRN 4111849*, 2022.

[2] Sergio Oramas, Francesco Barbieri, Oriol Nieto Caballero, and Xavier Serra. Multimodal deep learning for music genre classification. *Transactions of the International Society for Music Information Retrieval. 2018; 1 (1): 4-21.*, 2018.

[3] Nikki Pelchat and Craig M Gelowitz. Neural network music genre classification. *Canadian Journal of Electrical and Computer Engineering*, 43(3):170–173, 2020.