CS 534 - Artificial Intelligence

Final Report - Music Genre Classification

Due May 1, 2023

Abigail Albuquerque, Adam Beauchaine, Cameron Harvey, Ryan Darcey, Samuel Uche

Introduction

Audio signal processing for machine learning is an area that has generated substantial interest in recent years. The ubiquity of audio signals in real world problems, as well as their abundance of observable features has made the analysis and classification of audio signals a well explored topic of machine learning models. Less explored is the specific case problem of music genre classification, or the capacity of a model to analyze a musical audio file and determine its genre characteristics. This can be a challenging task as music genres have no clearly defined characteristics, with some works being immune to classification altogether. The complexity of this problem has resulted in several unique efforts to tackle. Kim et al. [1] utilize a convolutional neural network, as well as a recurrent neural network for spectrogram image processing, allowing for a robust model that considers temporal differences in songs. Annesi et al. [2] use traditional numerical data analysis of audio files such as Mel Frequency Cepstral Coefficients with a support vector machine. Our approach seeks to experiment with similar strategies as earlier research endeavors, as well as utilize our own novel data set for training our classifier, once our model is trained, we additionally aim to create a novel playlist generation tool based on our model's assessments. In so doing, we fulfill the following experimental goals:

1. Creating our own data set of audio files from Spotify
2. Creating a model capable of classifying audio files into various genres as after training on our own data.
3. Creating a playlist generation tool to provide a user with a playlist spawned from a given song based on the model's assessments.

Our report details our efforts to meet these goals, as well as documents our technical procedures, and roadblocks encountered, as well as a final analysis of our success.

Building our Dataset

We decided to model our dataset off of the popular music dataset ‘GTZAN’ on which several music genre classification studies have been conducted. The ‘GTZAN’ dataset contains 1000 songs, with 100 songs distributed across 10 genres. The 10 genres covered are blues, classical, country, electronic, hip-hop, jazz, metal, pop, reggae and rock. To create our own dataset, we decided to gather data directly from Spotify. We developed a dataset building program using the Spotify API library for python called ‘spotipy’. This library allowed us to leverage API calls that provide 30 second MP3 ‘preview’ clips, which we would use to build our dataset. This was instrumented by acquiring lists of spotify artist IDs, grabbing each artist’s 10 most popular song IDs, and grabbing preview files for each artist. This approach was selected largely for convenience, as all of these data fields are easily accessible via the Spotify API.

A problem that we faced while attempting to build our dataset was that the API does not return preview links to all the requested tracks. This was a known bug within the API that we found challenging to work around during initial data set construction. While we considered simply brute forcing around the issue by trying multiple artist IDs until each returned 10 songs successfully, we developed a more elegant solution allowing us to preserve our original approach for dataset building.

The work-around we found to tackle this problem was using selenium to scrape the preview link from the browser-based Spotify preview player. For a song with a Spotify track id value of <ID>, we navigated to https://open.spotify.com/embed/track/<ID>, started to play the preview, and captured the url where the site sent a request for the preview’s MP3 file. With this approach, we were able to get a 30 second sample for an arbitrary song on Spotify.

Preliminary Study

To set a baseline for our model and put the accuracy of our model into perspective, we performed 2 preliminary tasks that are described below.

How well do humans do?

Our first preliminary task involved conducting a survey to see how well humans did in classifying genres according to spotify. Eleven random songs were chosen from our dataset, and the participants were asked to classify each song into the genre they saw fit. Twenty people participated in the survey. We saw that most respondents guessed 5 out of the 11 genres correctly. However, the average was a 6.11 out of 11, placing the human 'accuracy' at 55.55%. The distribution of the total points scored by the participants can be seen in figure 1.

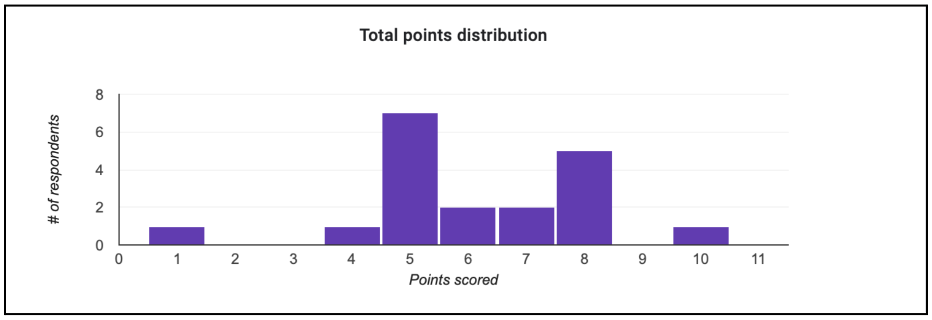


Figure 1: The graph shows the total point distribution of the survey.

Baseline Models

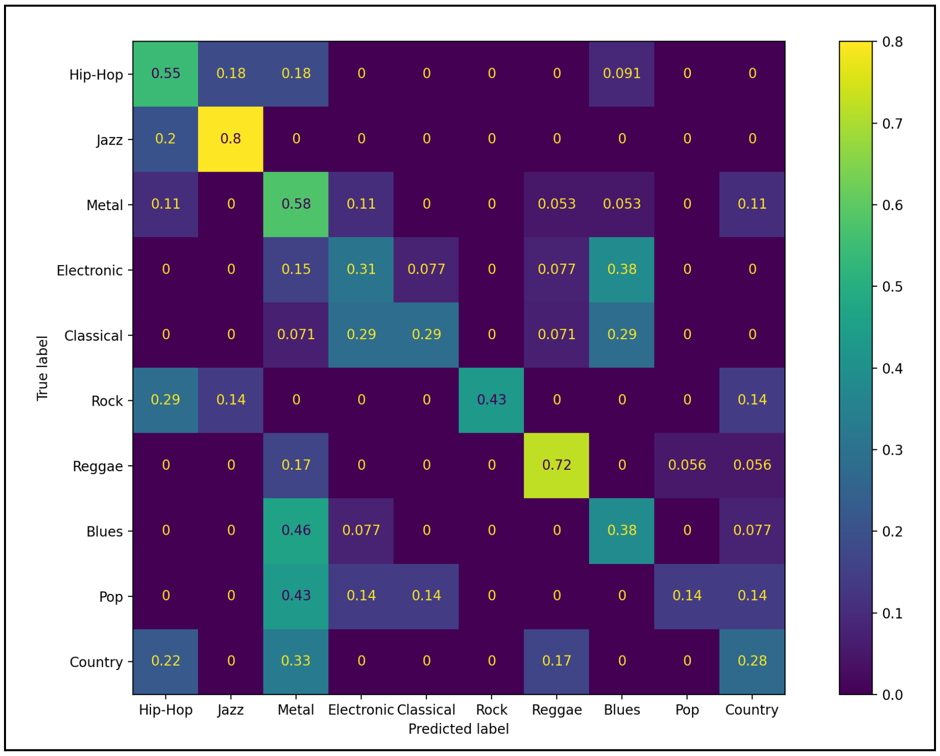
The second preliminary task involved creating baseline models, which would serve as a reference point for measuring our progress and determining where and by how much we could improve in terms of accuracy. The 'base' models used numerical data extracted from the 30-second audio clips in our dataset. The feature set was built following guidance from a variety of sources, without any targeted research. It contained 26 features including the zero-crossing rate, spectral centroid, spectral roll-off, spectral bandwidth, root mean squared energy, chroma filters and mel-frequency cepstral coefficients. The features were extracted using an audio and music processing library called 'librosa', along with libraries like 'soundfile' and 'scipy'. Simpler machine learning models like KNN and Logistic Regression, were built on this initial feature set. The 'original' dataset, with ~600 songs was used for these models. The cross-validation accuracy of the models is highlighted in Table 1.

Model	Accuracy
Naive Bayes	49.23 %
KNN	45.38 %
Decision Trees	38.46 %

Model	Accuracy
Logistic Regression	63.08%
SVM	59.23%
Random Forest	53.84%

Table 1: The different baseline models and their cross-validation accuracy built on the original dataset with ~600 songs and 26 features.

As we can see, the cross-validation accuracy for most of the models tends to lie around the 50%-60% range. We also built confusion matrices as seen in Figure 2 and 3 for a couple of these models, to see which genres our model was classifying more poorly than others. Using this information, we could learn and improve our feature set for the final model.



True \ Predicted	Hip-Hop	Jazz	Metal	Electronic	Classical	Rock	Reggae	Blues	Pop	Country
Hip-Hop	0.55	0.18	0.18	0	0	0	0	0.091	0	0
Jazz	0.2	0.8	0	0	0	0	0	0	0	0
Metal	0.11	0	0.58	0.11	0	0	0.053	0.053	0	0.11
Electronic	0	0	0.15	0.31	0.077	0	0.077	0.38	0	0
Classical	0	0	0.071	0.29	0.29	0	0.071	0.29	0	0
Rock	0.29	0.14	0	0	0	0.43	0	0	0	0.14
Reggae	0	0	0.17	0	0	0	0.72	0	0.056	0.056
Blues	0	0	0.46	0.077	0	0	0	0.38	0	0.077
Pop	0	0	0.43	0.14	0.14	0	0	0	0.14	0.14
Country	0.22	0	0.33	0	0	0	0.17	0	0	0.28

Figure 2: A confusion matrix for the KNN baseline model, where k=12. It shows the predicted genres on the X-axis and the true genres on the Y-axis. The numbers are seen as a percentage of the songs that were classified in that specific genre. As we can see, the model seems to be heavily misclassifying genres like Classical, Electronic, Pop, Blues and Country. The model seems to do well on genres like Jazz and Reggae.

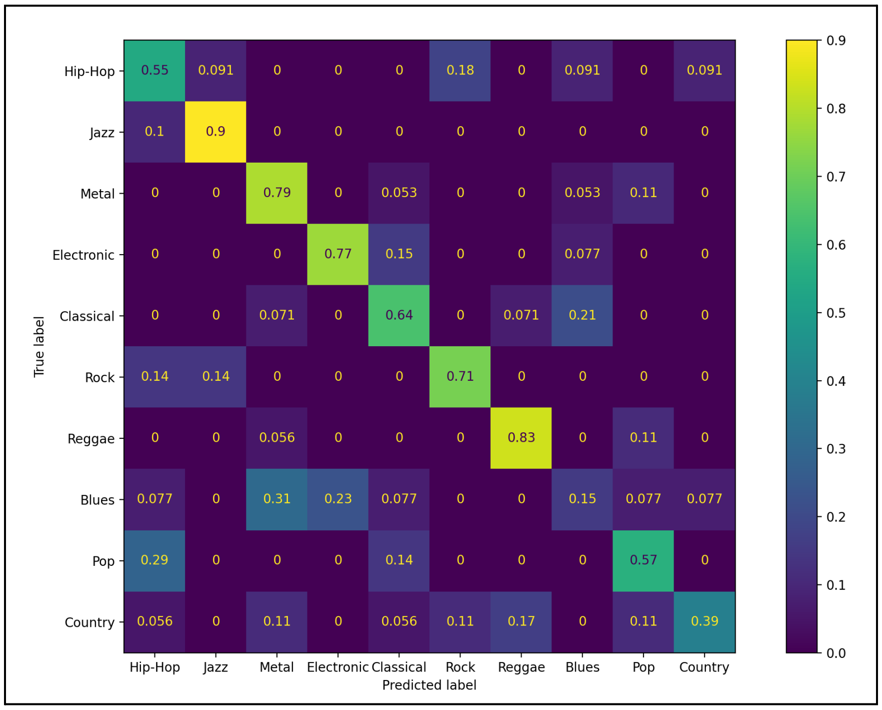


Figure 3: A confusion matrix for the Logistic Regression baseline model. It shows the predicted genres on the X-axis and the true genres on the Y-axis. The numbers are seen as a percentage of the songs that were classified in that specific genre. As we can see, the model seems to be heavily misclassifying genres like Blues, and Country. The model seems to do well on genres like Jazz, Metal and Reggae.

Methodology and Results

After building the baseline models, we decided to use 2 approaches to handle genre classification. First, using numerical features extracted from the audio clips. Second, converting the audio clips into mel-spectrogram image to perform image classification. Detailed descriptions of both the approaches, and their results are provided below. Throughout the process, we took inspiration from the research by Bahuleyan in the paper titled "Music Genre Classification using Machine Learning Techniques".

Approach 1 - Numerical Feature Extraction

Based on the results of the confusion matrices, we did some further research to see how we could improve the genre classification in specific areas. We continued to use the libraries 'librosa', ‘soundfile’ and ‘scipy’ in extracting features from the 30-second clips from our dataset. We ended up with a feature set of about 38 features. Some of the features included the mel-frequency cepstral coefficients, spectral features, the zero crossing rate, chroma filters, frequency features etc. The features were chosen to highlight the characteristics of music like the timbre, loudness and pitch. We then built different models from the data derived from this new feature set.

At this point, we encountered an issue where despite various techniques we were unable to surpass the 65%-75% accuracy mark. We realized that in order to improve accuracy, we needed to find a way to get more data.We were then able to find the workaround for the problems we were facing with spotify and gather ~1000 additional songs. We were now working with a dataset of ~1600 songs. After reading through some existing literature on music genre classification, we came across a paper released in 2016 by Zhang et al. [3] that stated that reducing the length of audio clips can boost the accuracy of the model. This is because the variability in a long music clip is reduced by using shorter clips, thus allowing the model to understand common trends across different genres. Thus, we split our audio data into 3 second clips and extracted the features on each of the 3 second clips for genre classification.

With the new data and processes, our accuracy increased to over 80% on several models! Table 2 highlights the accuracies achieved using the new approach. It is worth mentioning that we also experimented with deep-learning for the numerical features using a 1-Dimensional CNN, however we were not able to get an accuracy of over 22% with this model.

Model	Accuracy
KNN	80.94%
Random Forest	77.83%
Logistic Regression	63.82%

Model	Accuracy
MLP	89.22%
XGB	**90.71%**
CatBoost	89.03%

Table 2: The different models and their cross-validation accuracy built on the final dataset with ~1600 songs and 38 features. As we see, the model XGB (Extreme Gradient Boosting) performs the best with a 90.71% cross validation accuracy.

We can compare the diagonals for the KNN baseline and approach 1 models through Figure 4 and 2 to see how the classification has improved for different genres. As we can see, the diagonal of Figure 4 is much brighter, and the accuracy in any genre is not below 70%.

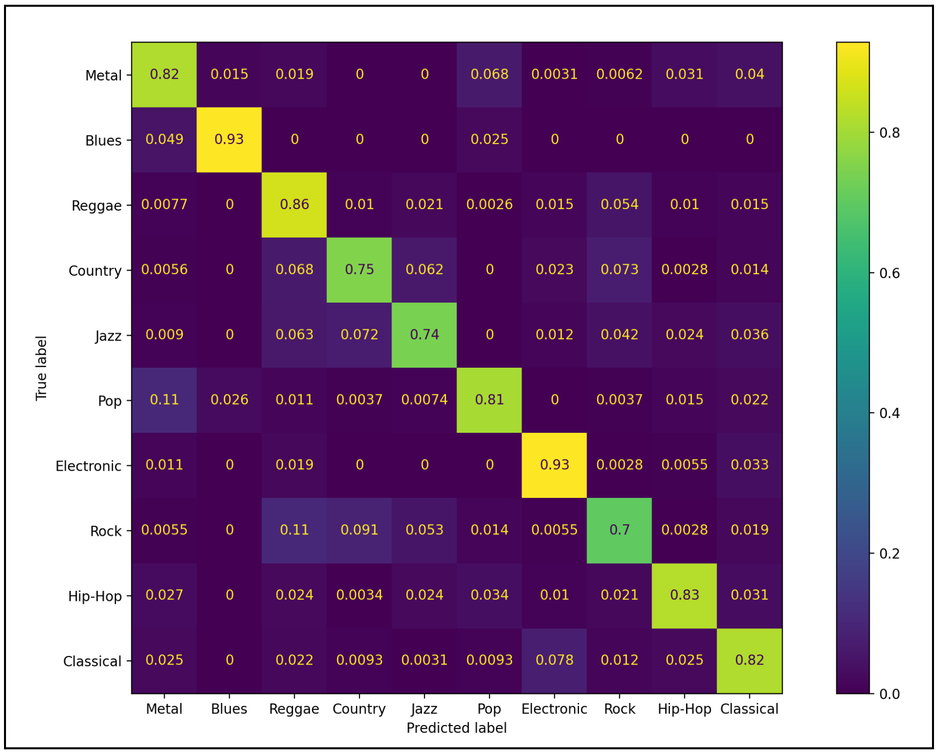


Figure 4: A confusion matrix for the KNN classifier with k=12. It shows the predicted genres on the X-axis and the true labels on the Y-axis. The numbers are seen as a percentage of the songs that were classified in that specific genre.

The best model for the numerical features approach was XGB also known as Extreme Gradient Boosting with an accuracy of 90.71%. It works by building a series of decision trees and iteratively improving their accuracy by adjusting the weights of misclassified samples [5]. The models' learning rate was set to 0.05 and the number of trees was set to 1000. The confusion matrix for the XGB model can be seen in Figure 5.

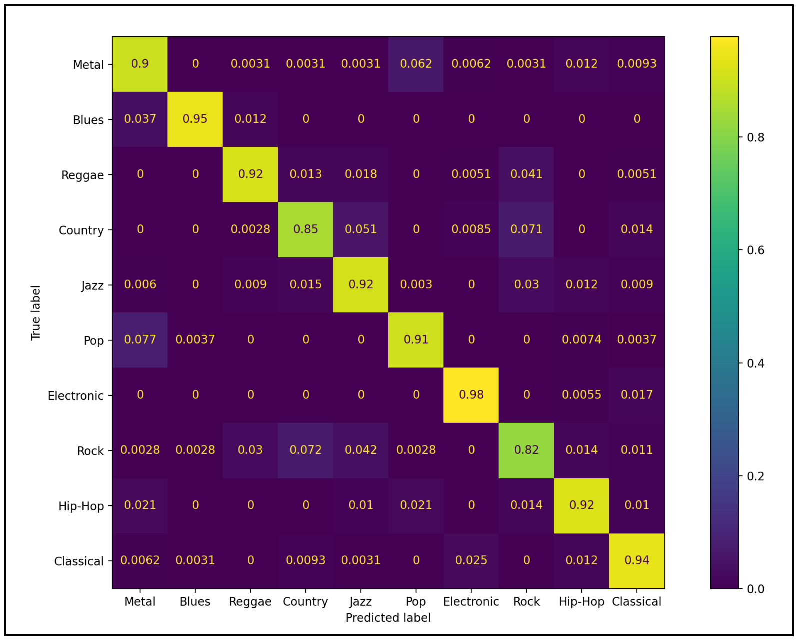


Figure 5:A confusion matrix for the XGB Classifier. It shows the predicted genres on the X-axis and the true labels on the Y-axis. The numbers are seen as a percentage of the songs that were classified in that specific genre. As we can see, the model does a very good job of classifying a variety of genres, with over an 80% accuracy across all.

Approach 2 - Image Classification

Our second approach was built around the classification of the Mel-Spectrograms Images. Mel-Spectrograms have two important changes as compared to regular spectrograms. They use the Mel-Scale, instead of the frequency on the Y-axis and use the decibel scale instead of the amplitude to indicate colors [6]. The X-axis continues to show the time. We generated Mel-Spectrograms for 30 second clips for all ~1600 songs in our dataset. Figure 6 shows examples of Mel-Frequency Spectrograms plotted from our dataset.

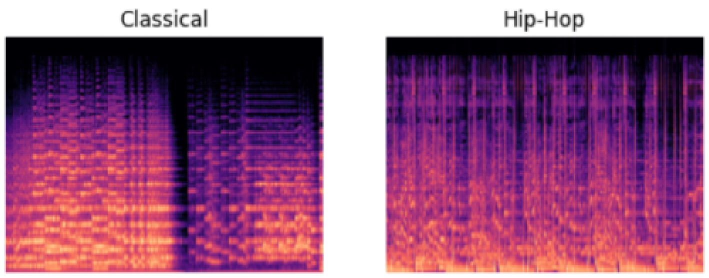
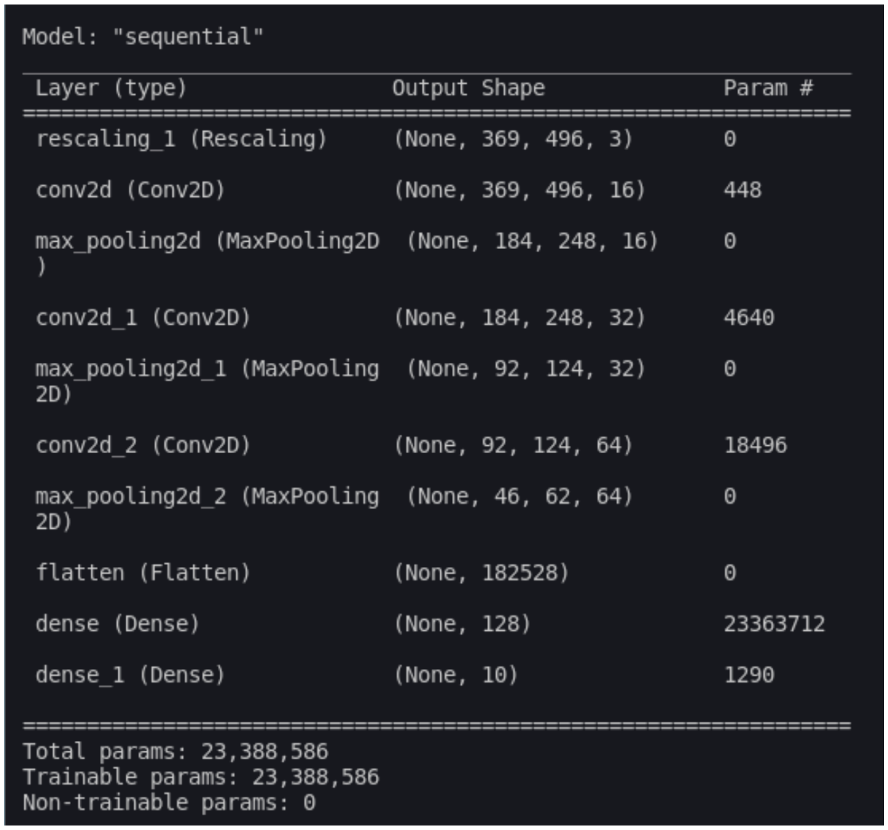


Figure 6: Mel-Spectrograms of a song from the classical and hip-hop genres in our dataset.

We then used the images to perform image classification using a CNN. The tensorflow library was used for this process, and the architecture of the CNN used can be seen in Figure 7. It first rescales the image data so the RGB values go from 0 to 1. It then has 3 convolutional blocks each with a convolution layer and a max pooling layer. The output layer is a fully connected layer using ReLU as its activation function.



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling_1 (Rescaling)     (None, 369, 496, 3)       0

 conv2d (Conv2D)             (None, 369, 496, 16)      448

 max_pooling2d (MaxPooling2D  (None, 184, 248, 16)     0
 )

 conv2d_1 (Conv2D)           (None, 184, 248, 32)      4640

 max_pooling2d_1 (MaxPooling  (None, 92, 124, 32)      0
 2D)

 conv2d_2 (Conv2D)           (None, 92, 124, 64)       18496

 max_pooling2d_2 (MaxPooling  (None, 46, 62, 64)       0
 2D)

 flatten (Flatten)           (None, 182528)            0

 dense (Dense)               (None, 128)               23363712

 dense_1 (Dense)             (None, 10)                1290

=================================================================
Total params: 23,388,586
Trainable params: 23,388,586
Non-trainable params: 0
```

Figure 7: Architecture of our image classification CNN model.

The image classification model was able to get a cross-validation accuracy of between ~91% and ~95% after 10 epochs of training. Thus, this fairly simple CNN was the best classification model in our study. Figure 8 shows the training and validation accuracy and loss for a run of the CNN.

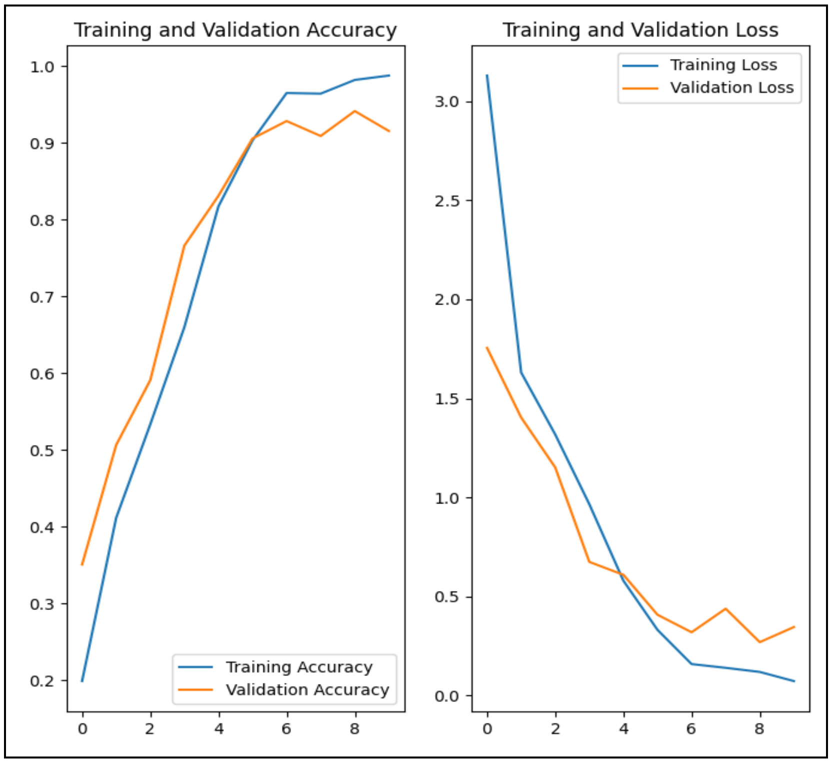


Figure 8: Training and validation accuracy and loss graphs for the CNN.

Playlist Generator

We found that our classification model could return a list of probabilities corresponding to the model's confidence that a song should be classified in each of the 10 genres. We developed a playlist generation method using this list of probabilities as a "song embedding", developed based on the word embeddings discussed in class for natural language processing. We stored these song embeddings for every song we classified, in order to refer to them later for playlist generation. For a playlist with desired size n and one initial song, we found the n-1 songs with the highest cosine similarity between their song embedding and the song embedding of the initial song. From our tests with a playlist size between 5 and 20, this typically resulted in playlists of a single genre matching the initial song, with similar instrumentation and vocal range, although the subgenre and themes of the songs varied more widely. We believe that this method could be developed further, especially with values across overlapping subgenres, in order to quickly ascertain the similarity of songs. We believe this method is comparable to other playlist generation techniques like cosine similarity of a song's numerical features, with greater possibility in the future.

Conclusions

Upon reflection, we're excited to have achieved success in all three of our initial experimental goals. Our dataset construction approach proved successful through the usage of the Spotify API and Selenium browser automation.We additionally experimented with multiple classification models using numerical features as well as a convolutional neural network for image classification of spectrograms, and noted the high success rate of ~95% using the CNN approach. This leads to sizable improvements of about 35% when compared to baseline models. We were finally able to leverage our models in the construction of a novel playlist generation application, using a novel approach similar to word embeddings in sequential data modeling.

Future work in this space could likely include the evaluation of polytimbral separation as numerical features within a model, or separated spectrograms within a CNN. These numerical and image based models could also be fused together in some novel way, perhaps leading to an even better model performance, such as using CNN modeling on additional spectrograms generated by our features, such as zero crossing rate. Given the wide variety of paths forward, the future of musical genre classification sounds very exciting, and we look forward to potentially pursuing some of these options in future projects.

Libraries used

catboost, dotenv, json, librosa, matplotlib, numpy, os, pandas, pickle, requests, scipy, selenium, sklearn, soundfile, spotipy, tensorflow, xgboost

References

1. Seonhoon Kim, Daesik Kim, and Bongwon Suh. 2016. Music Genre Classification using Multimodal Deep Learning. In Proceedings of HCI Korea (HCIK '16). Hanbit Media, Inc., Seoul, KOR, 389–395. https://doi.org/10.17210/hcik.2016.01.389
2. Paolo Annesi, Roberto Basili, Raffaele Gitto, Alessandro Moschitti, and Riccardo Petitti. 2007. Audio feature engineering for automatic music genre classification. In Large Scale Semantic Access to Content (Text, Image, Video, and Sound) (RIAO '07). LE CENTRE DE HAUTES ETUDES INTERNATIONALES D'INFORMATIQUE DOCUMENTAIRE, Paris, FRA, 702–711.

3. Zhang, W., Lei, W., Xu, X., & Xing, X. (2016). Improved music genre classification with convolutional neural networks. *Proc. Interspeech 2016*, 3304-3308. http://doi.org/10.21437/Interspeech.2016-1236
4. Bahuleyan, H. (2018). *Music Genre Classification using Machine Learning Techniques*. https://arxiv.org/pdf/1804.01149.pdf?
5. Nadeem. (2022, February 25). *Introduction to XGBoost Algorithm*. Analytics Vidhya. https://medium.com/analytics-vidhya/introduction-to-xgboost-algorithm-d2e7fad76b04
6. Ketan. (2021, February 19). *Audio Deep Learning Made Simple - Why Mel Spectrograms perform better*. Ketan Doshi Blog. https://ketanhdoshi.github.io/Audio-Mel/