Aphelion Music:

A Music Genre Classification System

23.11.2021

Team 5:

Aditya Panwar

Aneesh

Naman Gupta

Prasanna Telawane

GitHub Repo: https://github.com/adityapanwar94/ME781/

Overview

As the quantity of music being released on internet platforms is on a constant rise, genre classification becomes an important task and has a wide variety of applications. Based on the current statistics, almost 60,000 songs are uploaded to Spotify every day [1] and hence the need for reliable data required for search and storage purposes climbs in proportion. As the title suggests, the project was completed in two phases. The first phase of this work aims at classifying audio files based on genres. Several machine learning and deep learning algorithms were evaluated for solving the classification problem and the highest validation accuracy (90.10%) was observed for the model trained using Convolutional Neural Networks(CNN).

RASIC and Project timeline:

https://adityapanwar.notion.site/ME781-Project-8de0a1533e6e40359efb6b311f547439

Project Goals

Major Goals:

1. Create a Music Genre Classification System
    a. Check various models' capabilities
    b. Predict the genres using the best model
2. Creation of product marketing video, a repository for codes in development

Within-project Goals:

1. Identify and download dataset/s
2. Define RASIC and timeline
3. Conceptual project design
4. Coding and testing
5. Sample Dashboard

Milestones

The initial phase of the product development involved a thorough literature survey and study about various applications, researches and patents available for Music genre classification. More importantly, various productivity tools like Notion along with work division, RASIC charts, timeline etc. were determined during this period. Furthermore, consistent discussions were held with the teammates in order to evaluate the relevance of the various approaches used.

A few project milestones:

I. Feature Extraction Research

Finding and plotting different features of sounds including FFT, Spectrogram, Zero Crossing Rate, MFCC, and various other features for understanding the application of each in genre classification.

II. Features Selection

From the various identified features MFCC (Mel Frequency Cepstral Coefficients) were found to be the most reliable and adequate.

III. Model Selection

Assessing the training and testing accuracy of various models and finally selecting the most suitable model: Convolutional Neural Network (CNN)

IV. Prediction / Genre Classification

Finally predicting and Genre classification using the selected model.

V. Designing a UI/UX dashboard template

Designing a dashboard/website for classifying one or multi songs and creating a playlist based on the genres.

Software Used:

- Python:

  The programming language used in the product is Python

- Librosa (Version: 0.8.0):

  librosa is a python package for music and audio analysis. It provides the building blocks necessary to create music information retrieval systems.[2]

- NumPy:

  NumPy offers comprehensive mathematical functions, random number generators, linear algebra routines, Fourier transforms, and more.[3]

- Matplotlib:

  Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.[4]

- sci-kit learn:

  Scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms.[5]

- TensorFlow:

  TensorFlow is an end-to-end open-source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML-powered applications.[6]

- Keras:

  Keras is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library. [7]

- Pydub :

  It is a library for manipulating audio with a simple and easy high-level interface. [8]

A brief description:

Codes and other generated files are in this GitHub repository.

Dataset Used:

The dataset used in this project is called GTZAN dataset. This dataset was used for the well-known paper in genre classification "Musical genre classification of audio signals " by G. Tzanetakis and P. Cook in IEEE Transactions on Audio and Speech Processing 2002.

The files were collected in 2000-2001 from a variety of sources including personal CDs, radio, microphone recordings, in order to represent a variety of recording conditions.

The dataset consists of 1000 audio tracks each 30 seconds long. It contains 10 genres, each represented by 100 tracks. The tracks are all 22050 Hz Mono 16-bit audio files in .wav format.

Genres are: **blues, classical, country, disco, hip hop, jazz, metal, pop, reggae & rock.**

More details can be found here and the dataset can be downloaded from Marsyas homepage.

Audio Features:

As mentioned previously, a few different features were successfully/unsuccessfully extracted in this project. Following is a brief description of the same:

1. Audio feature extraction: Audio features help understand differences between different genres.

- Amplitude envelope - Amplitude envelopes give the shape of how loud the audio sample is. This helps us to identify the Timbre of the audio files and to help detect the onsets for sound detection, and noise filtering.

- Zero-Crossing Rate - The zero-crossing rate (ZCR) is the rate at which a signal changes from positive to zero to negative or from negative to zero to positive. Its value has been widely used in both speech recognition and music information retrieval, being a key feature to classify percussive sounds.

- Spectral Centroid - The spectral centroid is a measure used in Digital Signal Processing to characterise a spectrum. It indicates where the centre of mass of the spectrum is located. Perceptually, it has a robust connection with the impression of the brightness of a sound.

- Spectrograms - A spectrogram is a visual representation of the spectrum of frequencies of a signal as it varies with time. When applied to an audio signal, spectrograms are sometimes called sonographs, voiceprints, or voicegrams. When the data are represented in a 3D plot they may be called waterfalls. Spectrograms are used extensively in the fields of music, linguistics, sonar, radar, speech processing, etc.

- Mel Frequency Cepstral Coefficients - They are derived from a type of cepstral representation of the audio clip (a nonlinear ”spectrum of-a-spectrum”). The difference between the cepstrum and the Mel-frequency Cepstrum is that in the MFC, the frequency bands are equally spaced on the Mel scale, which approximates the human auditory system’s response more closely than the linearly spaced frequency bands used in the normal spectrum. This frequency warping can allow for better representation of sound, for example, in audio compression.

2. Music Genre Classification: In sound processing, the Mel-Frequency Cepstrum (MFC) is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear Mel scale. Mel-frequency Cepstral Coefficients (MFCCs) are coefficients that collectively make up an MFC. MFCCs are extracted from all the audio files in the GTZAN dataset. Each audio file is of 30 seconds duration, which is further divided into 10 segments to derive the MFCCs. A

JSON file is created to store the MFCCs of all the audios with their respective genres.

Features Extracted:

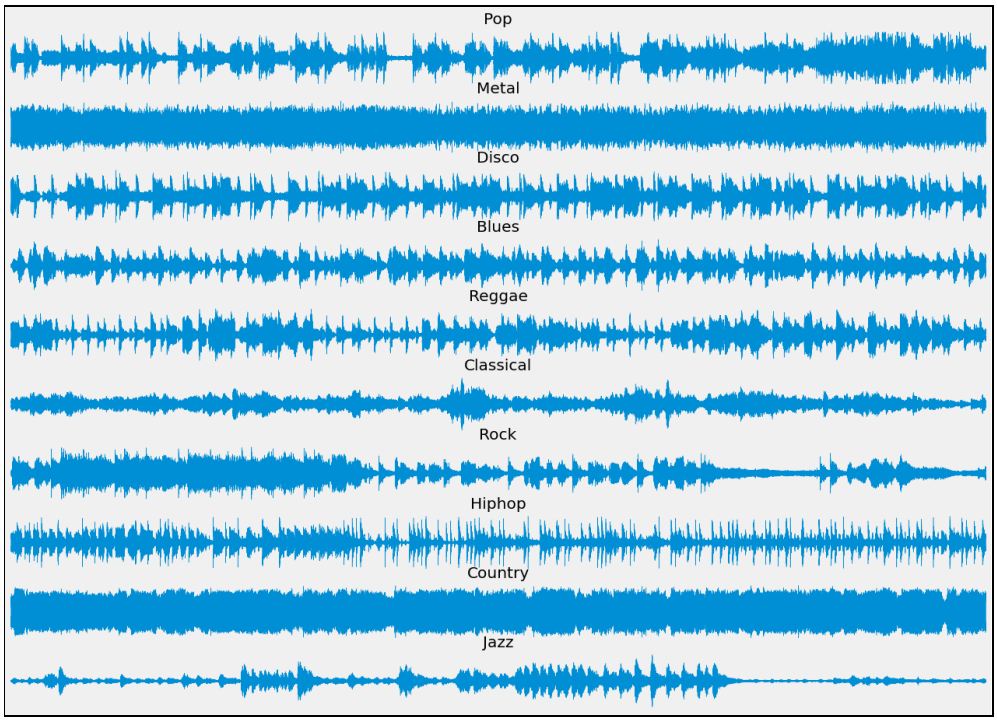


Fig: Genres Waveplots

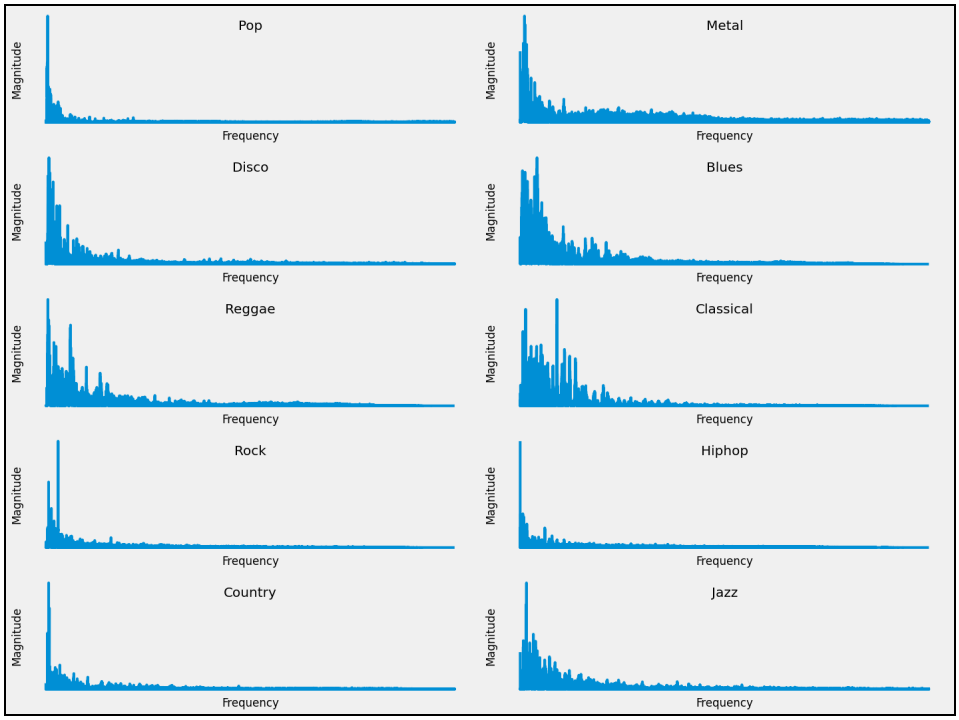
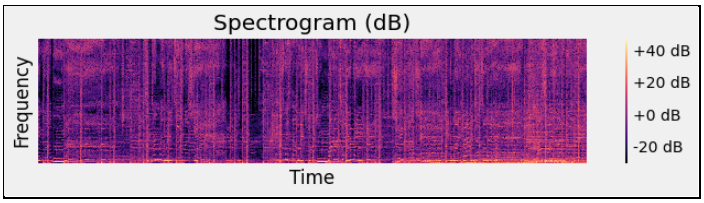


Fig: Genres Waveplots



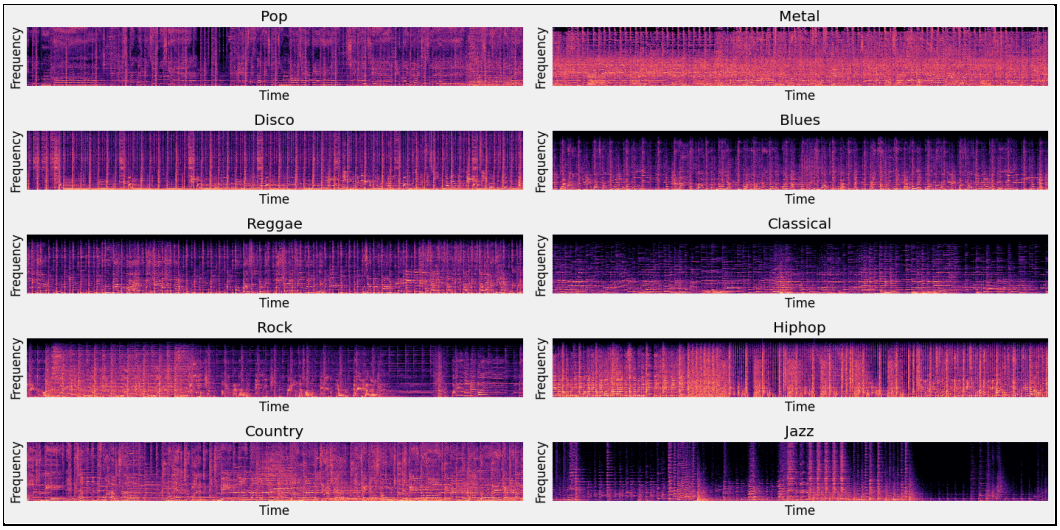


Fig: Spectrograms

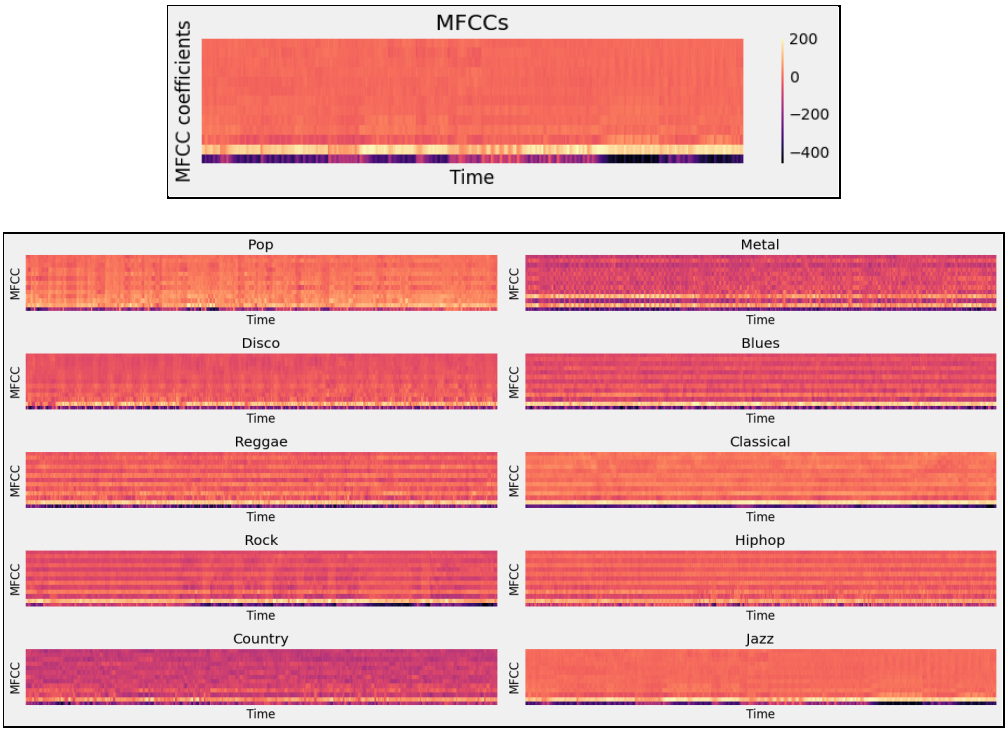


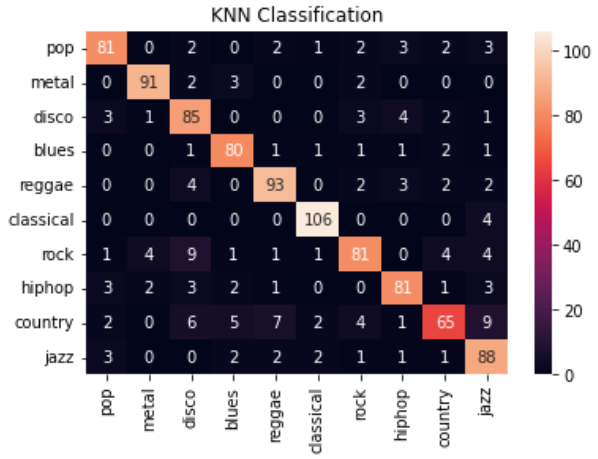
Fig: MFCC Features (USED IN FURTHER MODELS)

Models:

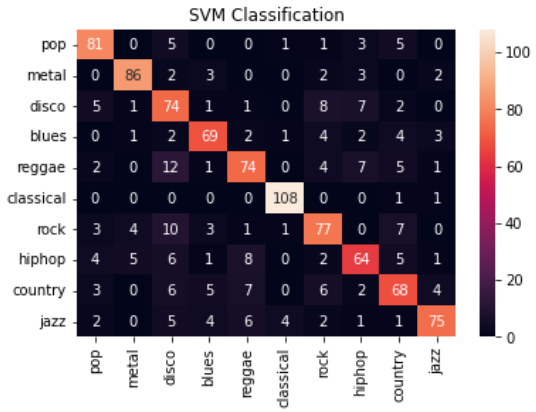
Models Employed:

1. Machine learning (ML) is the study of computer algorithms that improve automatically through experience and the use of data. The MFCC features were first scaled, and then dimensionality reduction was performed using Principal Component Analysis (PCA). Three common algorithms, **KNN, SVM and Random Forest** have been implemented.

a. **The K-Nearest Neighbors algorithm (KNN)** is a non-parametric classification method. In KNN classification, the output is a class membership. An object is classified by a plurality vote of its neighbours, with the object being assigned to the class most common among its 'k' nearest neighbours (k is a small positive integer). If k = 1, then the object is simply assigned to the class of that single nearest neighbour. This KNN model had a training accuracy of 99.88% and a validation accuracy of 85.1%. The model was heavily overfitting the data.

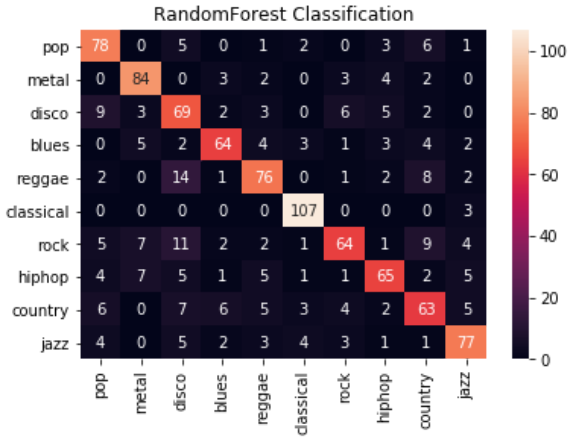


b. **Support Vector Machines(SVM)** are supervised learning models with associated learning algorithms that analyze data for classification. SVM maps training examples to points in space so as to maximise the width of the gap between the two categories. New examples are then mapped into the same space and are predicted to belong to a category based on the side of the gap they fall. This SVM model used for genre classification had a training accuracy of 87.24% and a validation accuracy of 77.8%.

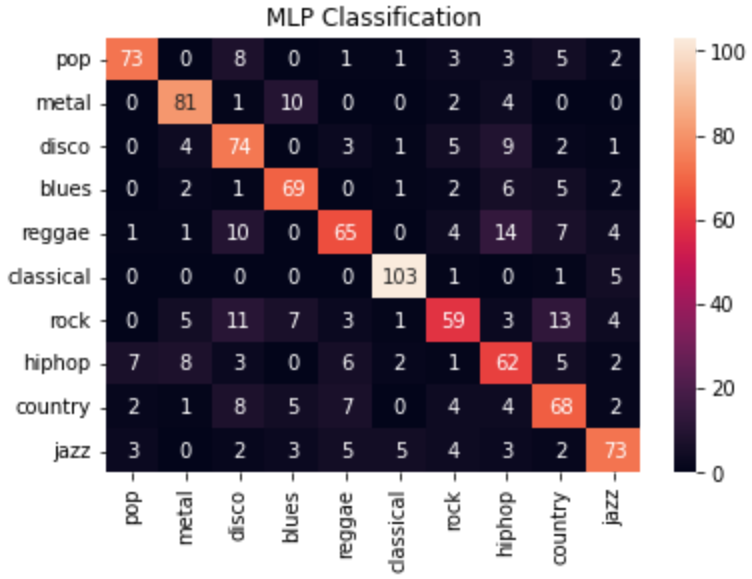


c. **Random Forests or Random Decision Forests** are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at the time of training. For classification tasks, the output of the random forest is the class selected by most trees. Random forests are more robust to outliers and overfitting problems. Random Forest algorithm is implemented by first performing grid search cross-validation to determine the optimum hyperparameters. The

accuracy achieved was 98.51% for the training data and 74.7% for the validation data.

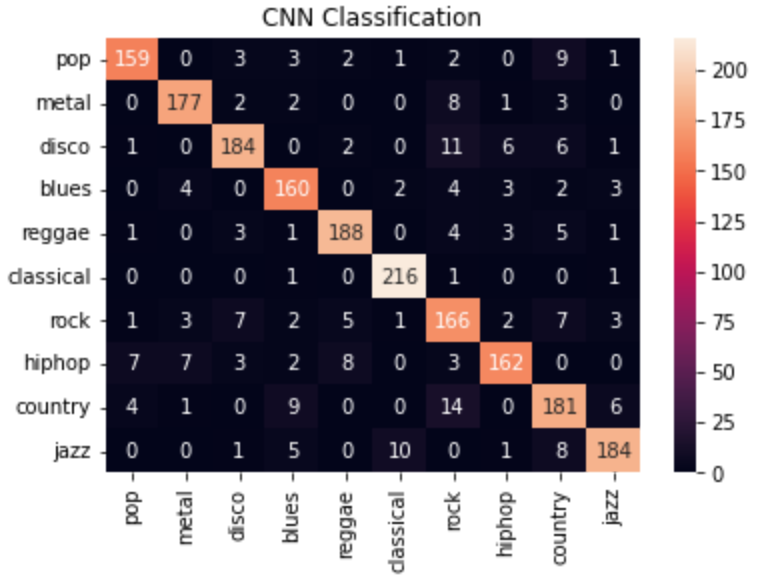


2. An Artificial Neural Network (ANN): is based on a collection of connected units or nodes called artificial neurons. Each connection transmits a signal to other neurons. An artificial neuron that receives a signal then processes it and can signal other neurons connected to it. The ”signal” at a connection is a real number, and the output of each neuron is computed by some non-linear function of the sum of its inputs. The connections are called edges. Neurons and edges typically have a weight that is used for adjustments as learning proceeds. The weight increases or decreases the strength of the signal at a connection. Neurons may have a threshold such that a signal is sent only if the aggregate signal crosses that threshold. Typically, neurons are aggregated into layers. Different layers may perform different transformations on their inputs. Signals travel from the first layer (the input layer), to the last layer (the output layer), possibly after traversing the layers multiple times.

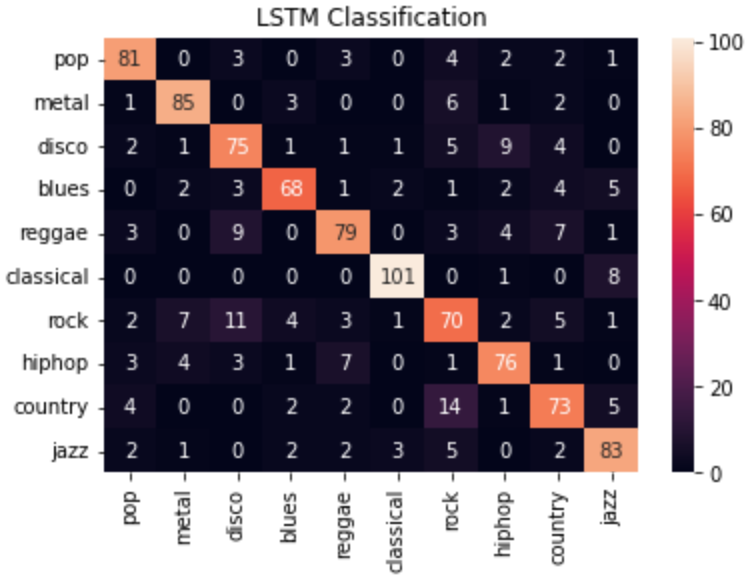


A 9 layer deep neural network, with dropout for regularization and batch normalization for stabilizing the learning process, was trained on the MFCCs extracted with a batch size of 32, for 75 epochs. The Adam optimizer is used with a learning rate of 0.01 and cross-entropy loss is monitored along with accuracy as a metric. This model achieved a training accuracy of 75.13% and a validation accuracy of 70.20%.

3. A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm that can take in an input image, assign importance (learnable weights and biases) to various aspects or objects in the image and can differentiate one from the other. The pre-processing required in a ConvNet is much lower when compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters or characteristics. A ConvNet is constructed by using 2D convolutional layers, batch normalization layers, and max-pooling layers for downsampling the feature maps. The optimization algorithm used is Adam and the learning rate is set to 0.001. Cross-entropy loss and accuracy are used as metrics and the network is trained for 40 epochs on data with a batch size of 32. The final training accuracy attained was 98.99% and a validation accuracy of 90.10%.



4. Long Short-Term Memory (LSTM) is an artificial Recurrent Neural Network (RNN) architecture[1] used in the field of deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections. It can not only process single data points (such as images), but also entire sequences of data (such as speech or video). An LSTM network was built consisting of two LSTM layers and three fully connected dense layers. Adam optimizer with a learning rate of 0.0001 was used. Cross-entropy loss and accuracy were set as the metrics and the network was trained for 40 epochs on data with a batch size of 32. Training and validation accuracies achieved were 99.82% and 88.20% respectively.



Models' Comparison:

The various algorithms used and the validation accuracy obtained for the same are shown in the table. The loss and accuracy functions are also plotted for each algorithm as shown in the figures below.

Table 1: Models' Comparison

Model	Accuracy (in %)
CNN	90.10
LSTM	88.20
ML (kNN)	85.10
ANN	71.80
RNN	70.10

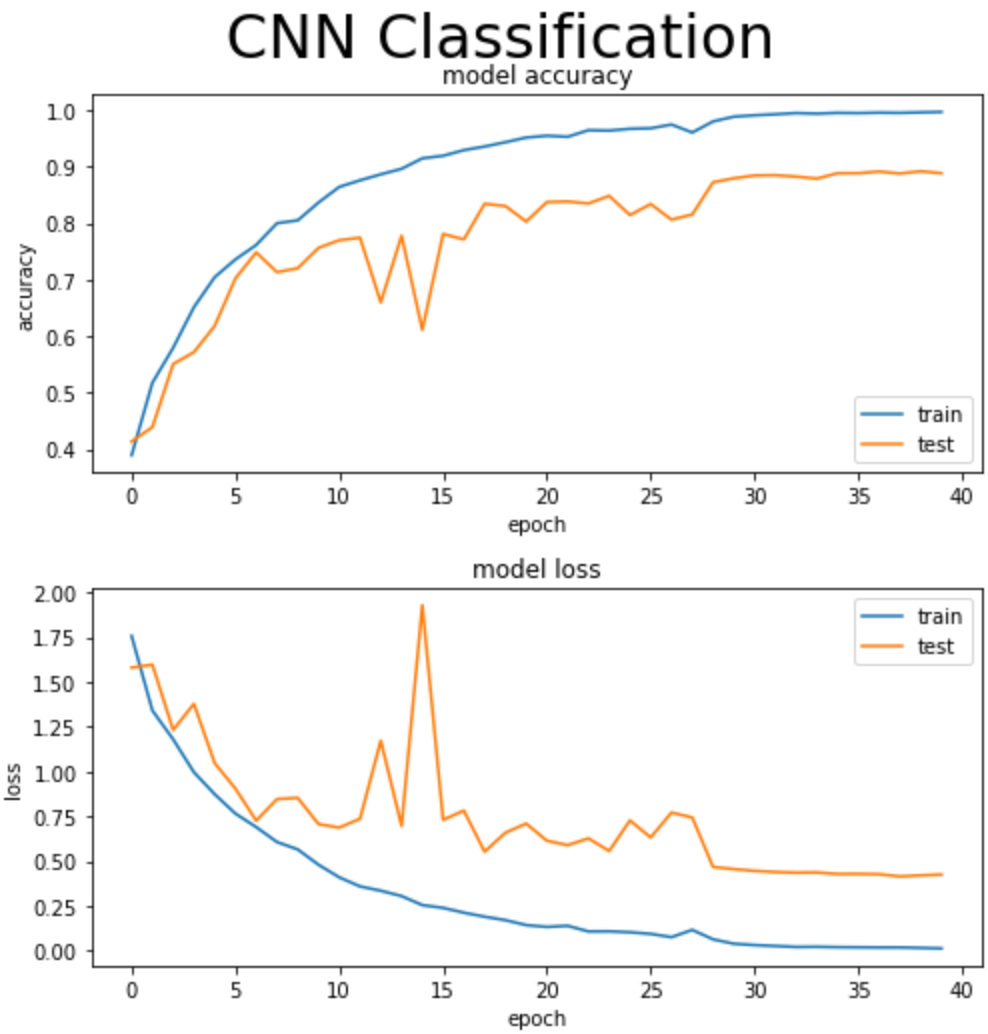


Fig: CNN Classfication Plot

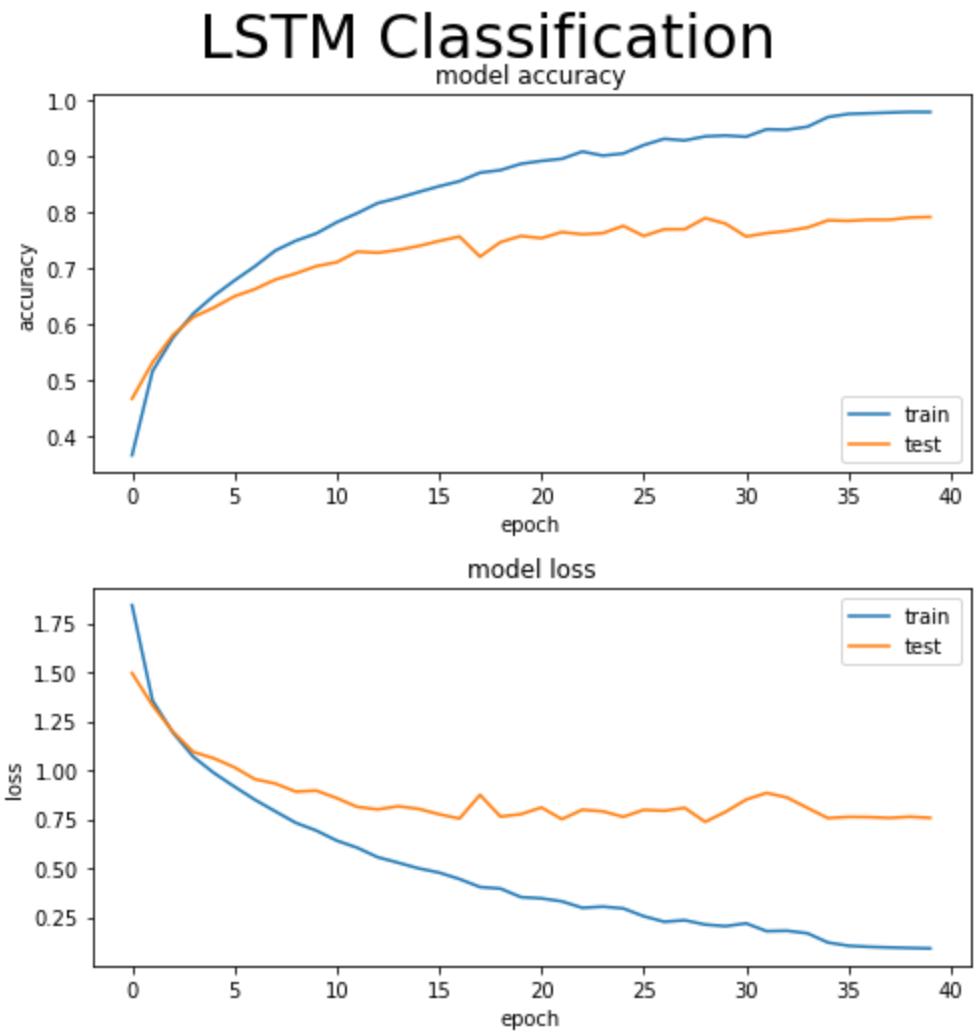


Fig: LSTM Classfication Plot

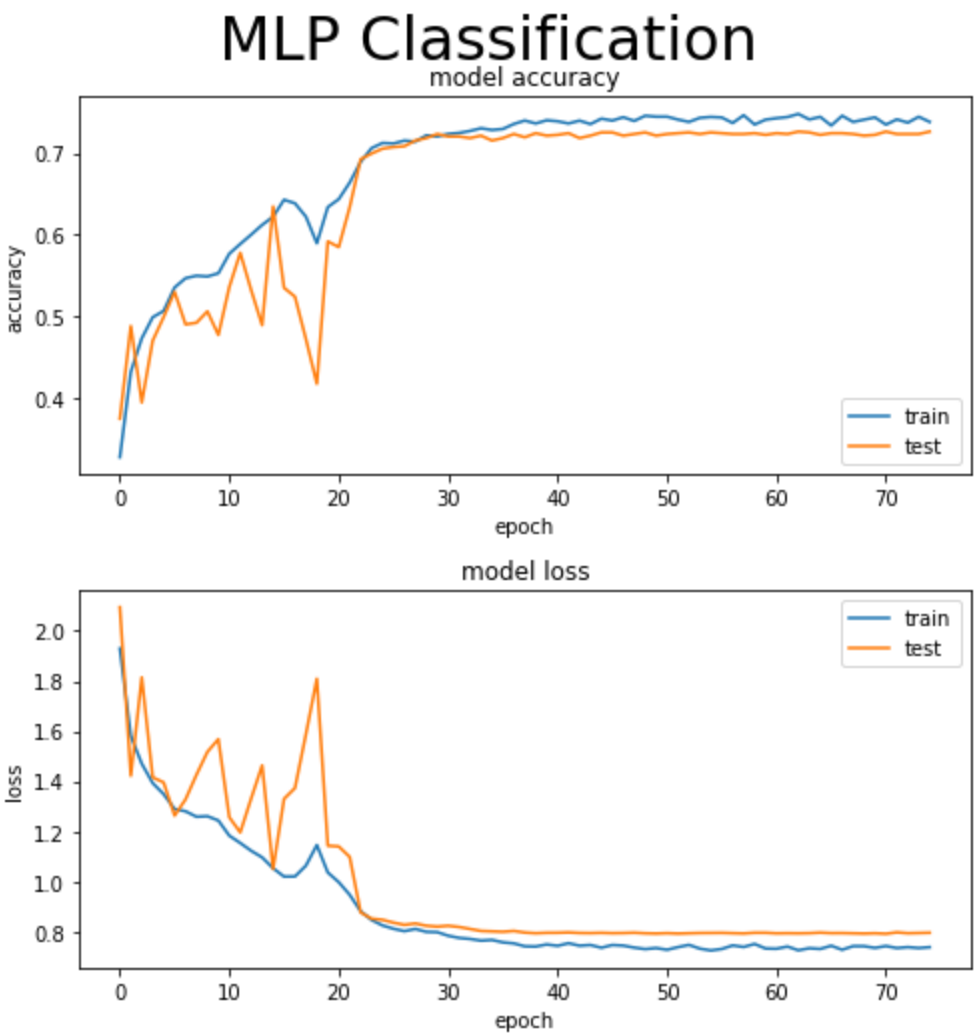


Fig: ANN Classification Plot

Unit Testing:

In computer programming, unit testing is a software testing method by which individual units of source codesets of one or more computer program modules together with associated control data, usage procedures, and operating procedures are tested to determine whether they are fit for use.

Our model expects an audio file in .wav format as input as also mentioned in the User Manual details. After uploading the file, we extract the MFCC features from the file and store it in a .json file which is then passed to various models. Since we have no independent units of source code, there isn't a requirement for unit testing in our model as we only input audio files which can be handled smoothly by the model.

User Interface:

Our team has created a sample website that will enable a user to browse through the song's list based on various genres, or upload one or more songs and classify them into their true genres with high accuracy. A few snapshots of the interface are shown below:

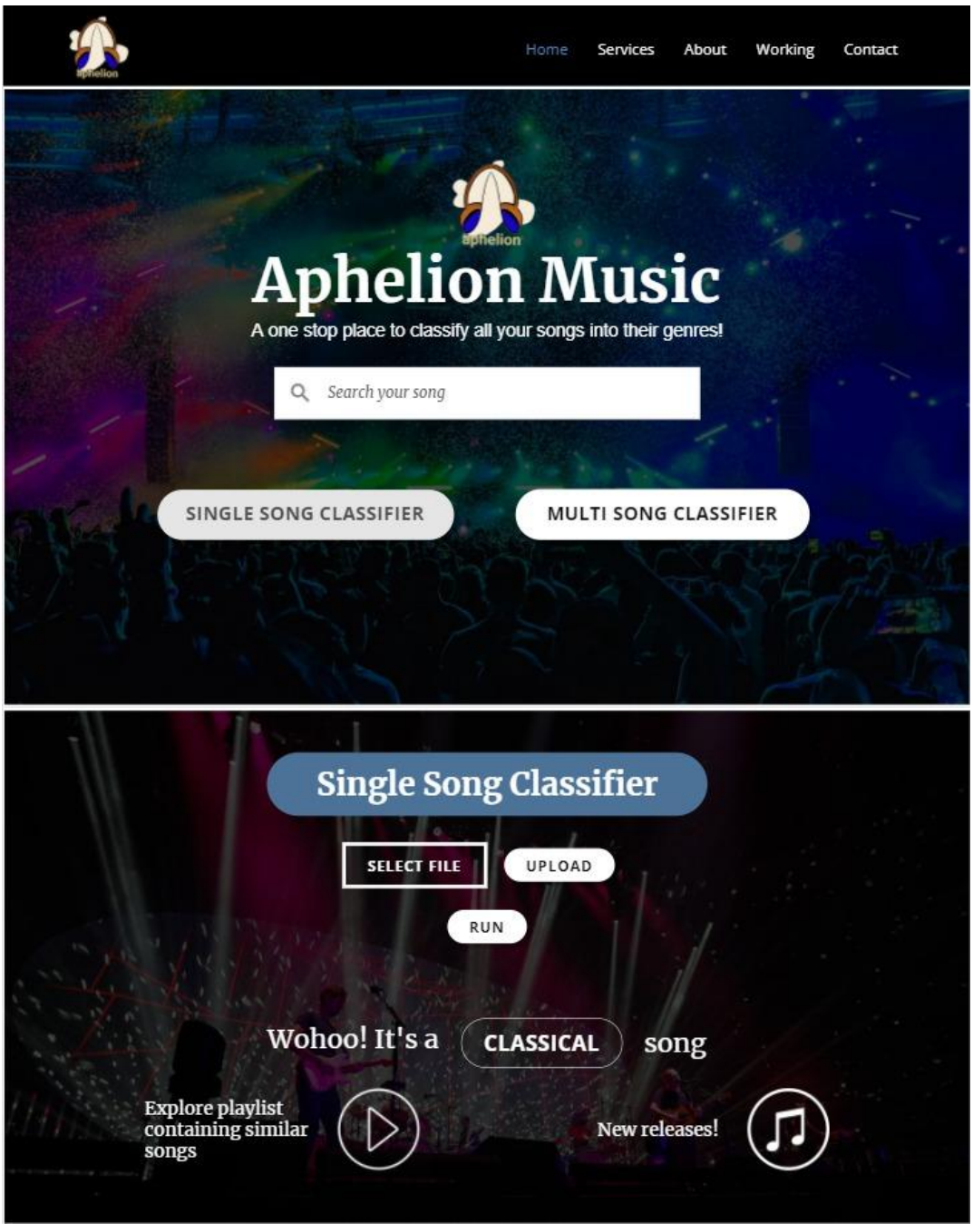


Fig: Web Interface for Aphelion Music

Fig: Mobile Interface for Aphelion Music

References:

[1]: https://www.igroovemusic.com/blog/how-many-artists-actually-make-bank-on-spotify.html?lang=en

[2]: https://librosa.org/doc/main/index.html

[3]: https://numpy.org/

[4]: https://matplotlib.org/

[5]: https://en.wikipedia.org/wiki/Scikit-learn

[6]: https://www.tensorflflow.org/

[7]: https://en.wikipedia.org/wiki/Keras

[8]: https://pypi.org/project/pydub/

[9]: Automatic Music Genre Classification Based on Modulation Spectral Analysis of Spectral and Cepstral Features

[10]: Automatic Genre Classification of Music Content

[11]: Automatic Music Genre Classification using Convolutional Neural Network

Appendix:

Logo Design: designed using Autodesk Sketchbook (Free app)