Table of Contents

1. INTRODUCTION

OVERVIEW :

Music classification is an interesting problem with many applications, from Drinkify (a program that generates cocktails to match the music) to Pandora to dynamically generating images that complement the music. However, music genre classification has been a challenging task in the field of music information retrieval (MIR). Music genres are hard to systematically and consistently describe due to their inherent subjective nature.

In this paper, we investigate various machine learning algorithms, including k-nearest neighbor (kNN), k-means, multi-class SVM, and neural networks to classify the following four genres: classical, jazz, metal, and pop. We relied purely on Mel Frequency Cepstral Coefficients (MFCC) to characterize our data as recommended by previous work in this field [5]. We then applied the machine learning algorithms using the MFCCs as our features.

Lastly, we explored an interesting extension by mapping images to music genres. We matched the song genres with clusters of images by using the Fourier-Mellin 2D transform to extract features and clustered the images with k-means.

PURPOSE:

It aims to predict the genre using an audio signal as its input. The objective of automating the music classification is to make the selection of songs quick and less cumbersome. If one has to manually classify the songs or music, one has to listen to a whole lot of songs and then select the genre. This is not only time-consuming but also difficult. Automating music classification can help to find valuable data such as trends, popular genres, and artists easily. Determining music genres is the very first step towards this direction.

2. LITERATURE SURVEY

EXISTING PROBLEM:

Data Retrieval Process:

Marsyas (Music Analysis, Retrieval, and Synthesis for Audio Signals) is an open source software framework for audio processing with specific emphasis on Music Information Retrieval Applications. Its website also provides access to a database, GTZAN Genre Collection, of 1000 audio tracks each 30 seconds long. There are 10 genres represented, each containing 100 tracks. All the tracks are 22050Hz Mono 16-bit audio files in .au format. We have chosen four of the most distinct genres for our research: classical, jazz, metal, and pop because multiple previous work has indicated that the success rate drops when the number of classifications is above 4.

Thus, our total data set was 400 songs, of which we used 70% for training and 30% for testing and measuring results.

We wrote a python script to read in the audio files of the 100 songs per genre and combine them into a .csv file. We then read the .csv file into Matlab, and extract the MFCC features for each song. We further reduced this matrix representation of each song by taking the mean vector and covariance matrix of the cepstral features and storing them as a cell matrix, effectively modeling the frequency features of each song as a multi-variate Gaussian distribution. Lastly, we applied both supervised and unsupervised machine learning algorithms, using the reduced mean vector and covariance matrix as the features for each song to train on.

Mel Frequency Cepstral Coefficients (MFCC):

For audio processing, we needed to find a way to concisely represent song waveforms. Existing music processing literature pointed us to MFCCs as a way to represent time domain waveforms as just a few frequency domain coefficients.

To compute the MFCC, we first read in the middle 50% of the mp3 waveform and take 20 ms frames at a parameterized interval. For each frame, we multiply by a hamming window to smooth the edges, and then take the Fourier Transform to get the frequency components. We then map the frequencies to the mel scale, which models human perception of changes in pitch, which is approximately linear below 1kHz and logarithmic above 1kHz. This mapping groups the frequencies into 20 bins by calculating triangle window coefficients based on the mel scale, multiplying that by the frequencies, and taking the log. We then take the Discrete Cosine Transform, which serves as an approximation of the Karhunen-Loeve Transform, to decorrelate the frequency components. Finally, we keep the first 15 of these 20 frequencies since higher frequencies are the details that make less of a difference to human perception and contain less information about the song. Thus, we represent each raw song waveform as a matrix of cepstral features, where each row is a vector of 15 cepstral frequencies of one 20 ms frame for a parameterized number of frames per song.

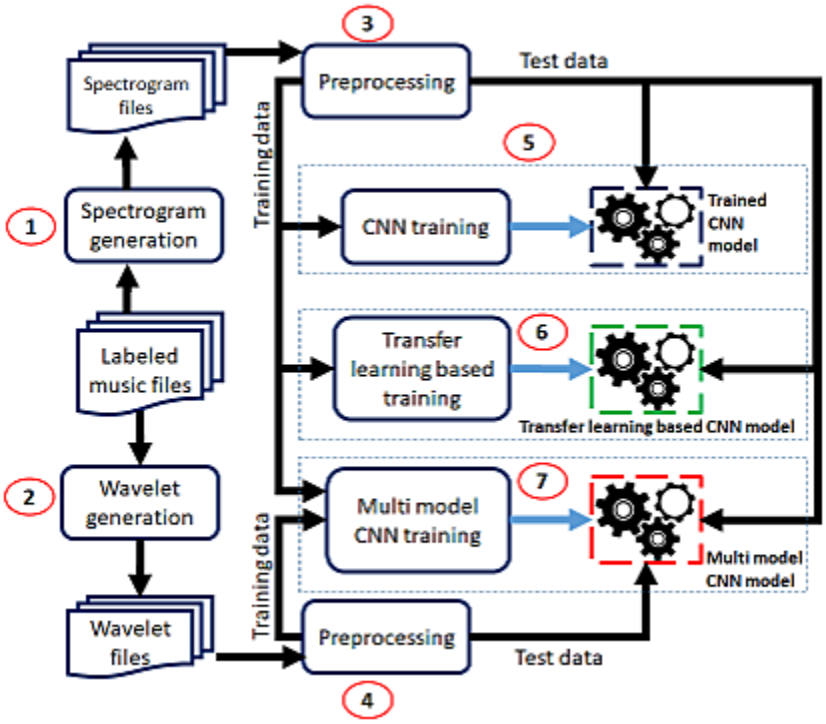
We further reduce this matrix representation of each song by taking the mean vector and covariance matrix of the cepstral features over each 20ms frame, and storing them as a cell matrix. Modeling the frequencies as a multi-variate Gaussian distribution again compressed the computational requirements of comparing songs with KL Divergence.

MCC FLOW

3. THEORITICAL ANALYSIS

BLOCK DIAGRAM:

represents the overview of our methodology for the genre classification task. We will discuss each phase in detail. We train three types of deep learning models to explore and gain insights from the data.



HARDWARE/SOFTWARE DESIGNING:

Requirements are the minimal configurations of a device and software required for the model to work properly and efficiently.

Hardware requirements:

Graphics Processing Unit (GPU)

Intel Core i3 processor or above 2.2

Software requirements:

Windows 7 or above / Linux

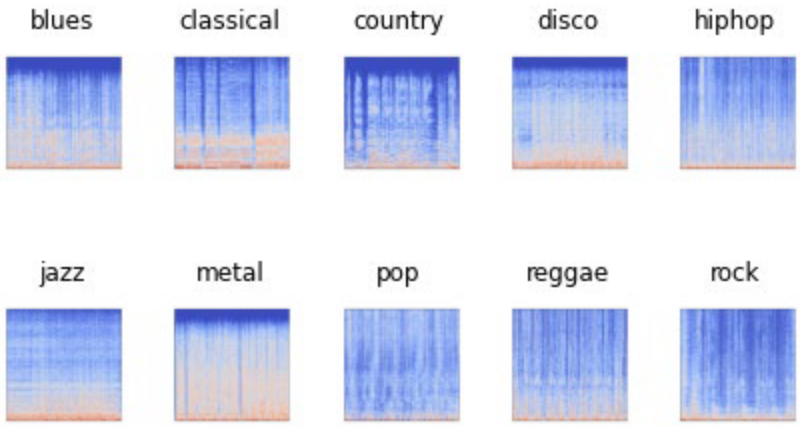
Python 2.7 or above

Jupyter Notebook.

4. EXPERIMENTAL INVESTIGATIONS

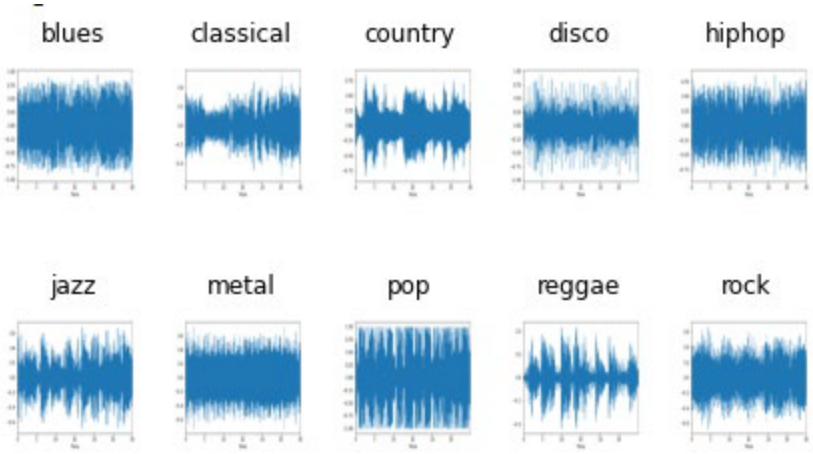
Spectrogram generation:

A spectrogram is a visual representation of the spectrum signal frequencies as it varies with time. We use librosa library to transform each audio file into a spectrogram. Figure below shows spectrogram images for each type of music genre.



Wavelet generation:

The Wavelet Transform is a transformation that can be used to analyze the spectral and temporal properties of non-stationary signals like audio. We use librosa library to generate wavelets of each audio file. Figure shows wavelets of each type of music genre.



Spectrogram and Wavelet preprocessing:

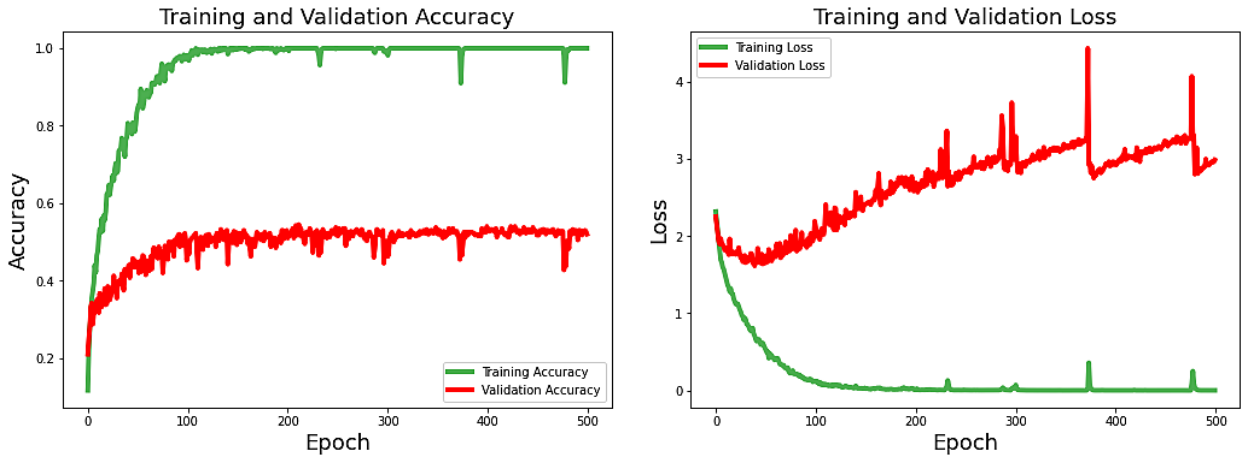
It is clear that we treat our data as image data. After generating spectrograms and wavelets, we apply general image preprocessing steps to generate training and testing data. Each image is of size (256, 256, 3).

Basic CNN model training:

After preprocessing the data, we create our first deep learning model. We construct a Convolution Neural Network model with required input and out units. The final architecture of our CNN model is shown in Figure . We use only spectrogram data for the training and testing.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 256, 256, 32)      896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 128, 128, 32)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 128, 128, 32)      9248
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 64, 64, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 64, 64, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 32, 32, 64)        0
_________________________________________________________________
dropout (Dropout)            (None, 32, 32, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 65536)             0
_________________________________________________________________
dense (Dense)                (None, 128)               8388736
_________________________________________________________________
dense_1 (Dense)              (None, 10)                1290
=================================================================
Total params: 8,418,666
Trainable params: 8,418,666
Non-trainable params: 0
_________________________________________________________________
```

We train our CNN model for 500 epochs with Adam optimizer at a learning rate of 0.0001. We use categorical cross-entropy as the loss function. Figure 05 shows the training and validation losses and model performance in terms of accuracy.



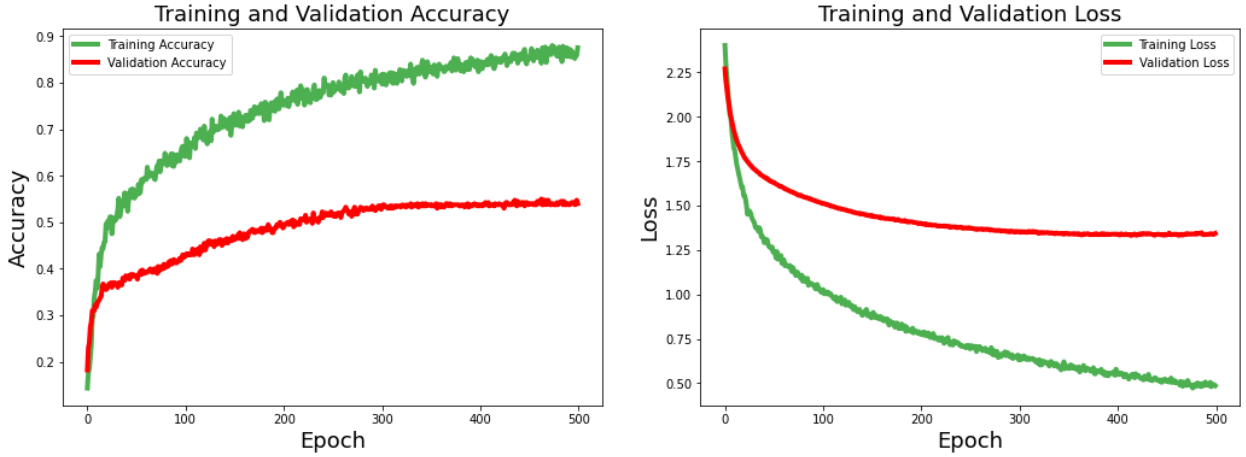
Transfer learning-based model training:

We have only 60 samples of each genre for training. In this case, transfer learning could be a

useful option to improve the performance of our CNN model. Now, we use the pre-trained mobilenet model to train the CNN model. A schematic architecture is shown in Figure .

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
mobilenetv2_1.00_224 (Functi (None, 8, 8, 1280)        2257984
_________________________________________________________________
global_average_pooling2d (Gl (None, 1280)              0
_________________________________________________________________
dropout (Dropout)            (None, 1280)              0
_________________________________________________________________
dense (Dense)                (None, 10)                12810
=================================================================
Total params: 2,270,794
Trainable params: 12,810
Non-trainable params: 2,257,984
_________________________________________________________________
```

The transfer learning-based model is trained with the same settings as used in the previous model. below figure shows the training and validation loss and model performance in terms of accuracy. Here, also we use only spectrogram data for the training and testing.



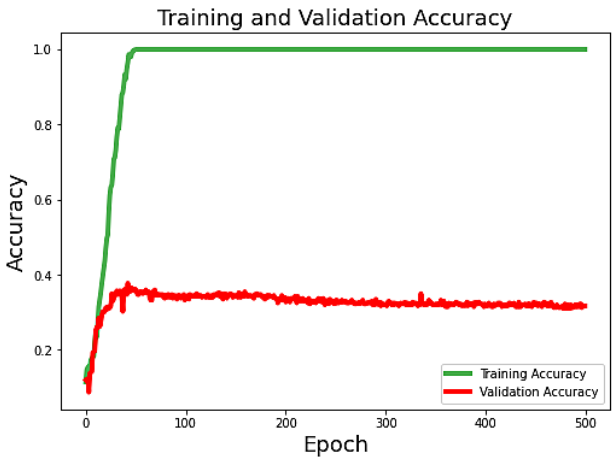
Multimodal training:

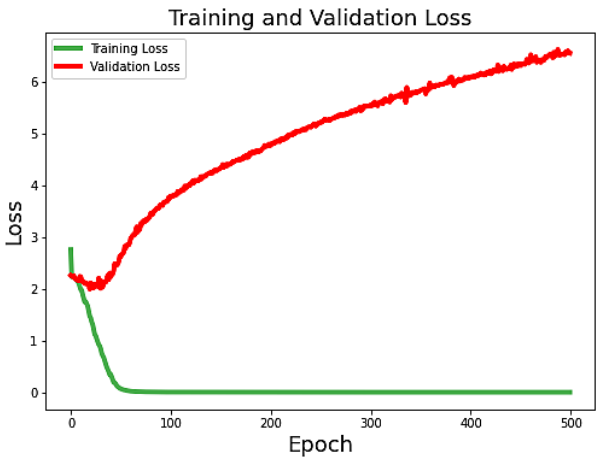
We will pass both spectrogram and wavelet data into the CNN model for the training in this experiment. We are using the late-fusion technique in this multi-modal training. Table represents the architecture of our multi-modal CNN model. Figure shows the loss and performance scores

of the model with respect to epochs.

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 256, 256, 3)	0	
input_2 (InputLayer)	[(None, 256, 256, 3)	0	
conv2d (Conv2D)	(None, 256, 256, 32)	896	input_1[0][0]
conv2d_2 (Conv2D)	(None, 256, 256, 32)	896	input_2[0][0]
max_pooling2d (MaxPooling2D)	(None, 128, 128, 32)	0	conv2d[0][0]
max_pooling2d_2 (MaxPooling2D)	(None, 128, 128, 32)	0	conv2d_2[0][0]
conv2d_1 (Conv2D)	(None, 128, 128, 64)	18496	max_pooling2d[0][0]
conv2d_3 (Conv2D)	(None, 128, 128, 64)	18496	max_pooling2d_2[0][0]
max_pooling2d_1 (MaxPooling2D)	(None, 64, 64, 64)	0	conv2d_1[0][0]
max_pooling2d_3 (MaxPooling2D)	(None, 64, 64, 64)	0	conv2d_3[0][0]
dropout (Dropout)	(None, 64, 64, 64)	0	max_pooling2d_1[0][0]
dropout_1 (Dropout)	(None, 64, 64, 64)	0	max_pooling2d_3[0][0]
flatten (Flatten)	(None, 262144)	0	dropout[0][0]
flatten_1 (Flatten)	(None, 262144)	0	dropout_1[0][0]
dense (Dense)	(None, 128)	33554560	flatten[0][0]
dense_1 (Dense)	(None, 128)	33554560	flatten_1[0][0]
tf.concat (TFOpLambda)	(None, 256)	0	dense[0][0] dense_1[0][0]
dense_2 (Dense)	(None, 32)	8224	tf.concat[0][0]
dense_3 (Dense)	(None, 10)	330	dense_2[0][0]

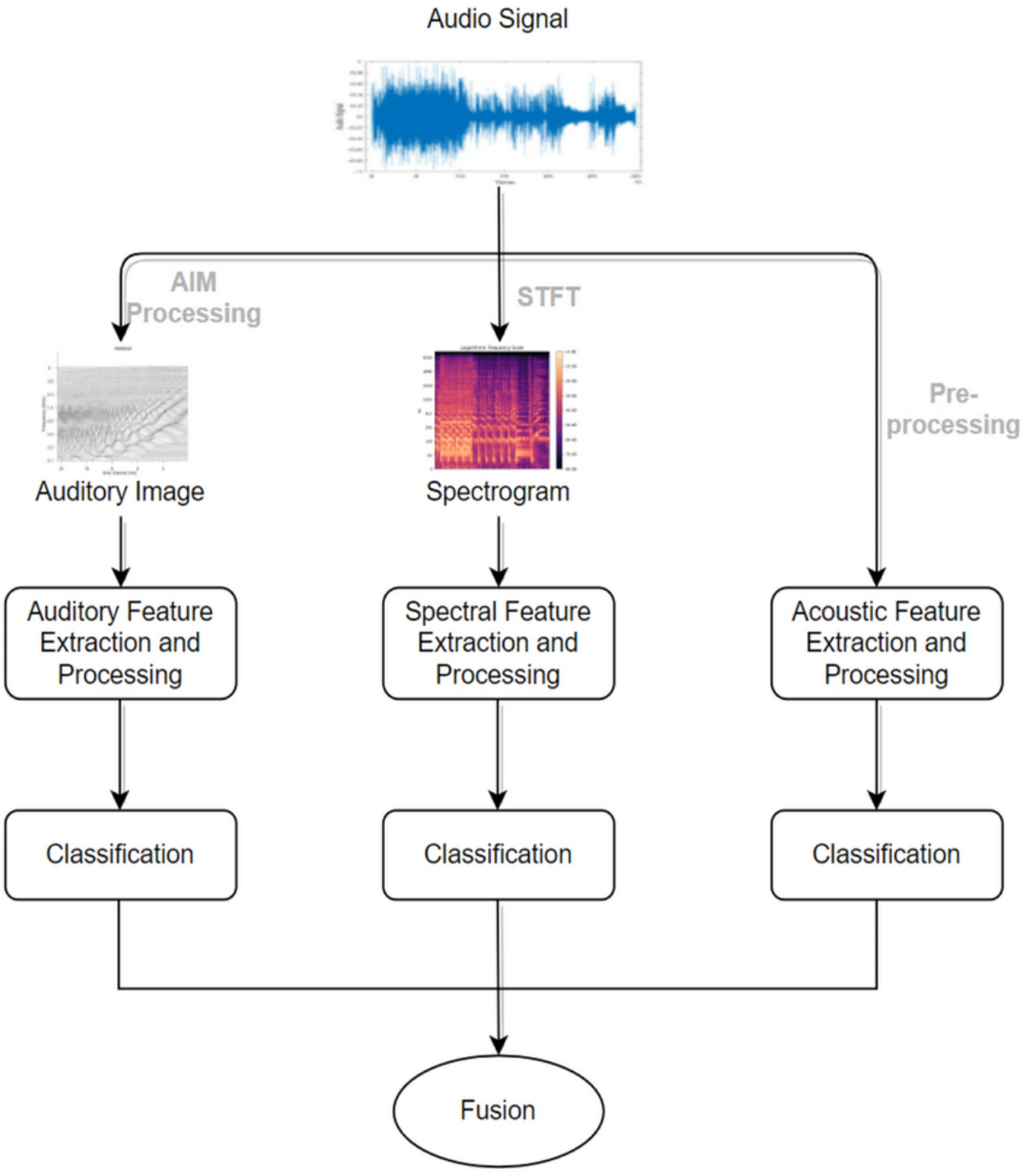
Total params: 67,156,458
Trainable params: 67,156,458
Non-trainable params: 0





5. FLOWCHART

Flowchart of the proposed music genre classification



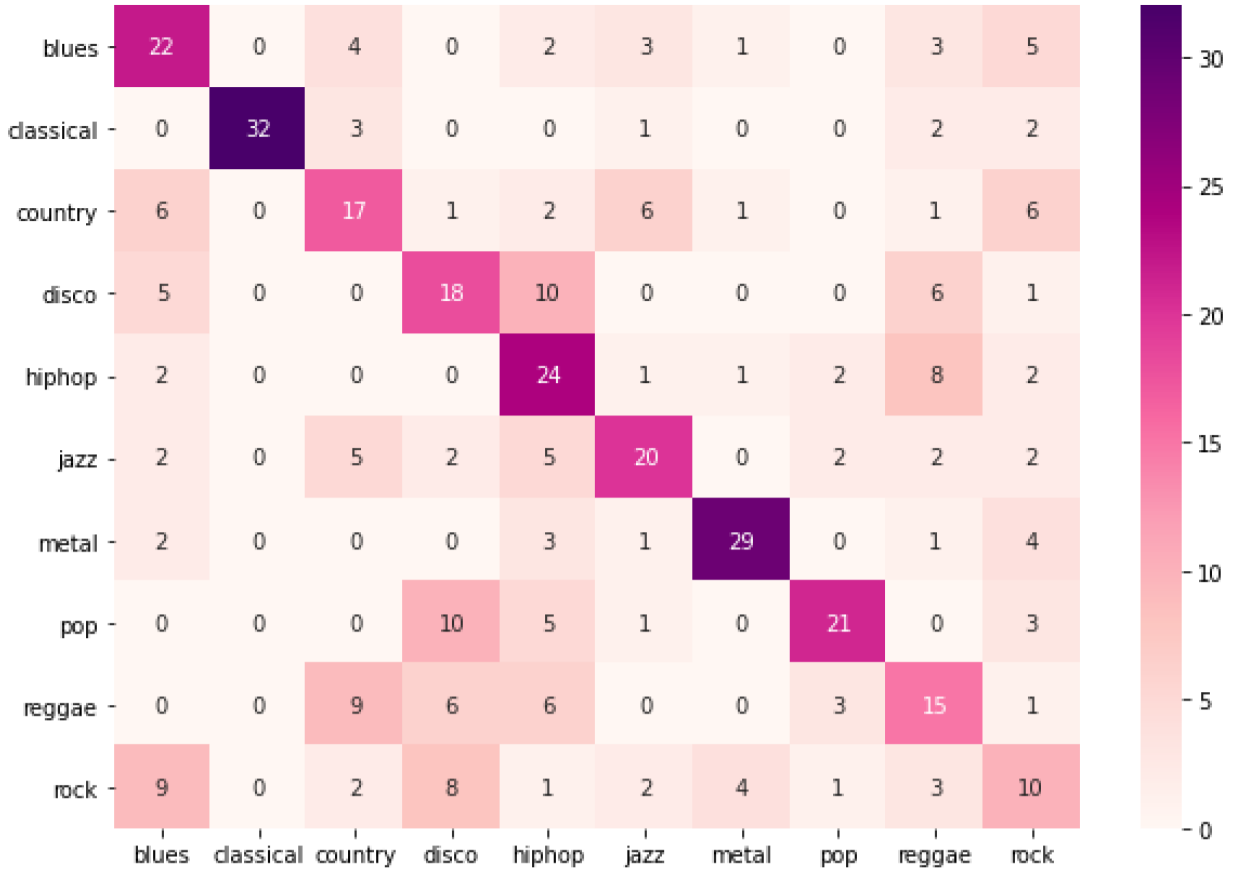
6. Testing the models:

After training our models, we test each model on the 40% test data. We calculate precision, recall, and F-score for each music genre (class). Our dataset is balanced; therefore, the macro average and weighted average of precision, recall, and F-score are the same

a. Basic CNN model

Figure presents the results of our CNN model on the test data. CNN model was able to classify “classical” genre music with the highest F1-score. CNN performed worst for “Rock” and “reggae” genre music. Figure shows the confusion matrix of the CNN model on the test data

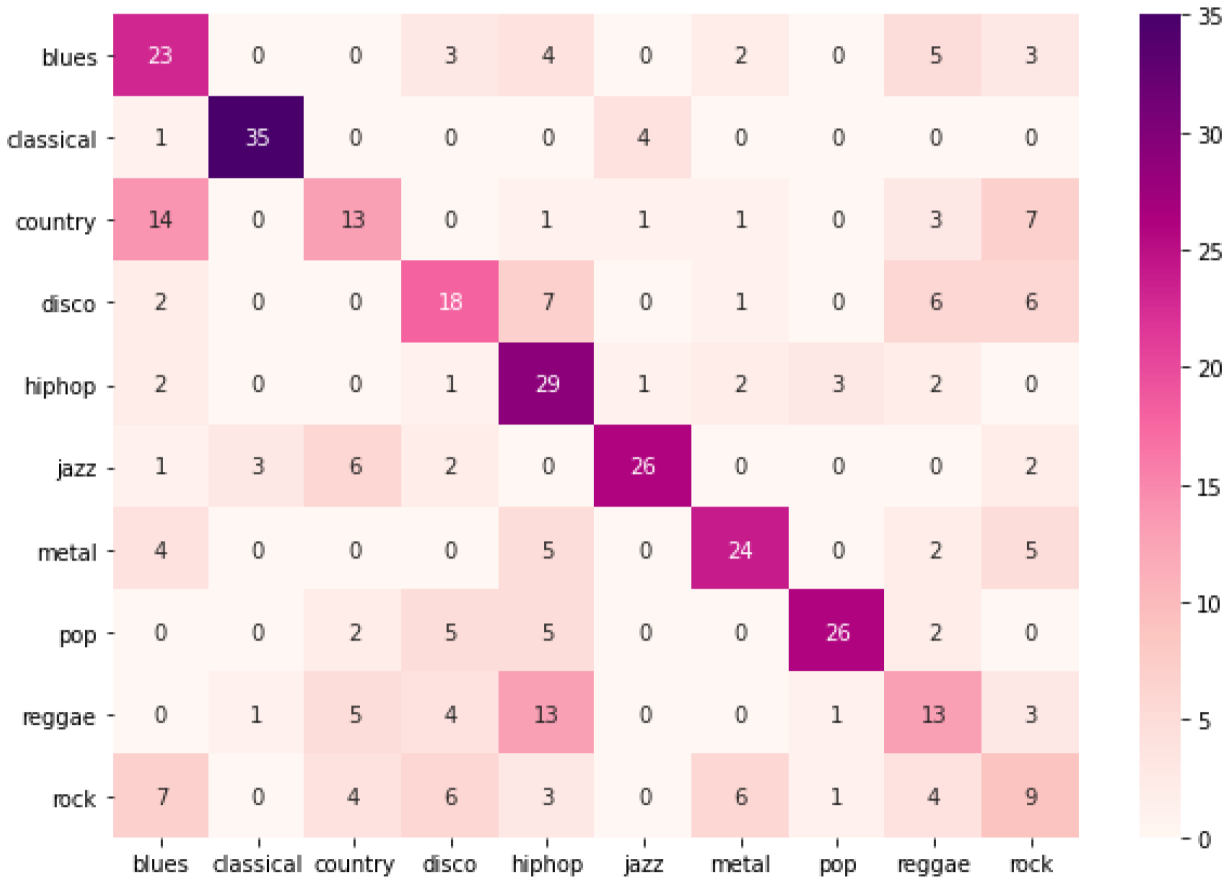
	precision	recall	f1-score	support
blues	0.46	0.55	0.50	40
classical	1.00	0.80	0.89	40
country	0.42	0.42	0.42	40
disco	0.40	0.45	0.42	40
hiphop	0.41	0.60	0.49	40
jazz	0.57	0.50	0.53	40
metal	0.81	0.72	0.76	40
pop	0.72	0.53	0.61	40
reggae	0.37	0.38	0.37	40
rock	0.28	0.25	0.26	40
accuracy			0.52	400
macro avg	0.54	0.52	0.53	400
weighted avg	0.54	0.52	0.53	400



b. **Transfer learning based model**

We used the transfer learning technique to improve the performance of genre classification. Figure presents the results of the transfer learning-based model on test data. F1-score for “hiphop”, “jazz”, and “pop” genres increased due to transfer learning. If we look at overall results, we have achieved only a minor improvement after applying transfer learning. Figure shows the confusion matrix for the transfer learning model on the test data.

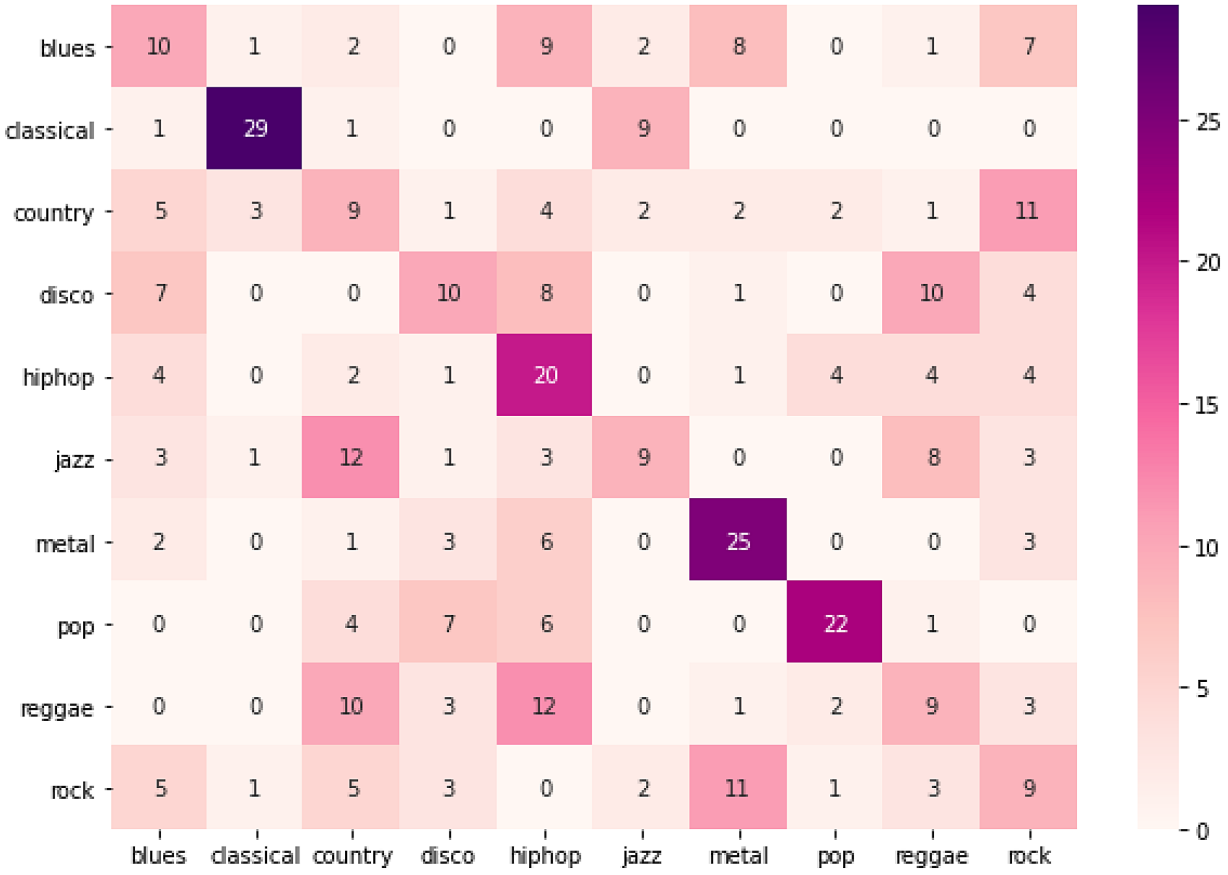
	precision	recall	f1-score	support
blues	0.43	0.57	0.49	40
classical	0.90	0.88	0.89	40
country	0.43	0.33	0.37	40
disco	0.46	0.45	0.46	40
hiphop	0.43	0.72	0.54	40
jazz	0.81	0.65	0.72	40
metal	0.67	0.60	0.63	40
pop	0.84	0.65	0.73	40
reggae	0.35	0.33	0.34	40
rock	0.26	0.23	0.24	40
accuracy			0.54	400
macro avg	0.56	0.54	0.54	400
weighted avg	0.56	0.54	0.54	400



c. Multimodal-based model:

We have used both spectrogram and wavelet data to train the multimodal-based model. In the same way, we perform the testing. We have found very surprising results. Instead of improvement, our performance reduced drastically. We have achieved only 38% of F1-score while using a multi-modal approach. Figure 16 shows the confusion matrix of the multimodal-based model on the test data

	precision	recall	f1-score	support
blues	0.27	0.25	0.26	40
classical	0.83	0.72	0.77	40
country	0.20	0.23	0.21	40
disco	0.34	0.25	0.29	40
hiphop	0.29	0.50	0.37	40
jazz	0.38	0.23	0.28	40
metal	0.51	0.62	0.56	40
pop	0.71	0.55	0.62	40
reggae	0.24	0.23	0.23	40
rock	0.20	0.23	0.21	40
accuracy			0.38	400
macro avg	0.40	0.38	0.38	400
weighted avg	0.40	0.38	0.38	400



7. RESULT

For this project, the dataset that we will be working with is GTZAN Genre Classification dataset which consists of 1,000 audio tracks, each 30 seconds long. It contains 10 genres, each represented by 100 tracks.

The 10 genres are as follows:

- Blues
- Classical
- Country
- Disco
- Hip-hop

- Jazz
- Metal
- Pop
- Reggae
- Rock

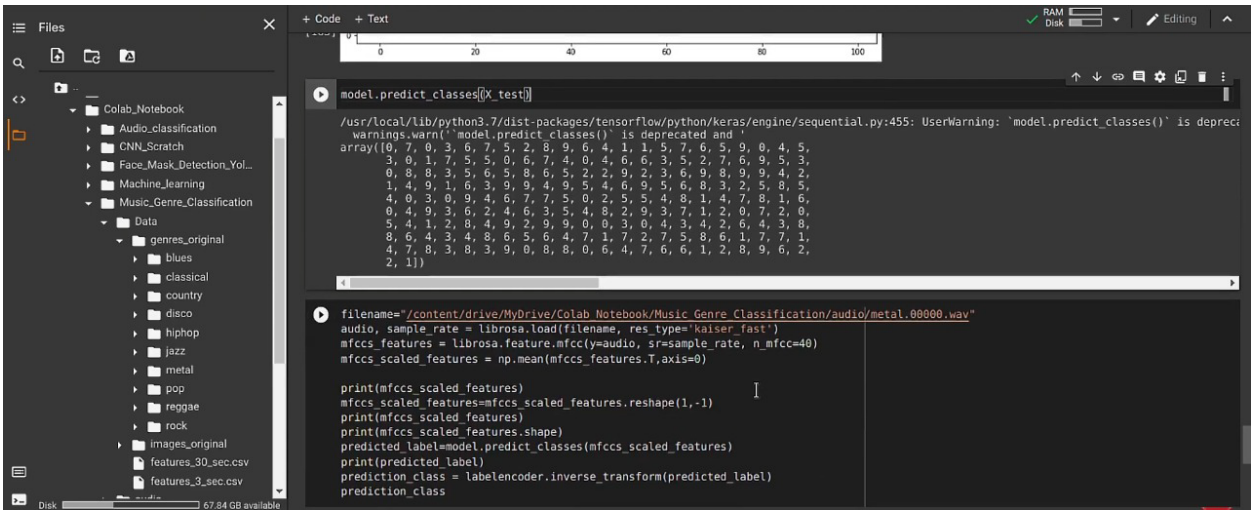
The dataset has the following folders:

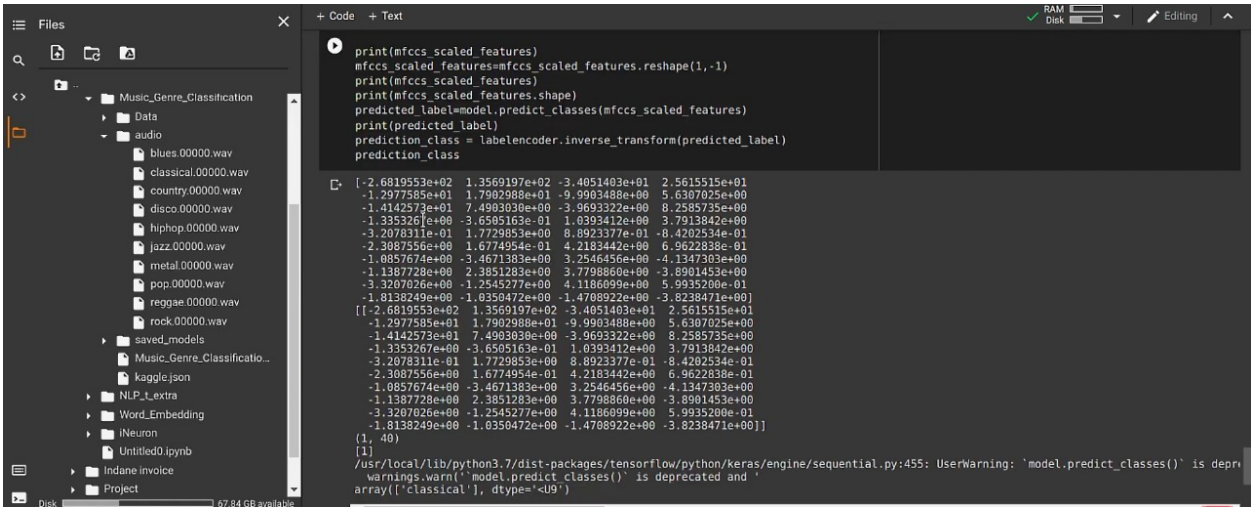
Genres original — A collection of 10 genres with 100 audio files each, all having a length of 30 seconds (the famous GTZAN dataset, the MNIST of sounds)

Images original — A visual representation for each audio file. One way to classify data is through neural networks because NN's usually take in some sort of image representation.

2 CSV files — Containing features of the audio files. One file has for each song (30 seconds long) a mean and variance computed over multiple features that can be extracted from an audio file. The other file has the same structure, but the songs are split before into 3 seconds audio files.

From here the outputs


Files
+ Code + Text
RAM
Disk
Editing
Colab_Notebook
Audio_classification
CNN_Scratch
Face_Mask_Detection_Yol...
Machine_learning
Music_Genre_Classification
Data
genres_original
blues
classical
country
disco
hiphop
jazz
metal
pop
reggae
rock
images_original
features_30_sec.csv
features_3_sec.csv
Disk
67.84 GB available
model.predict_classes(X_test)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/sequential.py:455: UserWarning: `model.predict_classes()` is deprecated and '
warnings.warn('`model.predict_classes()` is deprecated and '
array([0, 7, 0, 3, 6, 7, 5, 2, 8, 9, 6, 4, 1, 1, 5, 7, 6, 5, 9, 0, 4, 5,
3, 0, 1, 7, 5, 5, 0, 6, 7, 4, 0, 4, 6, 6, 3, 5, 2, 7, 6, 9, 5, 3,
0, 8, 8, 3, 5, 6, 5, 8, 6, 5, 2, 2, 9, 2, 3, 6, 9, 8, 9, 9, 4, 2,
1, 4, 9, 1, 6, 3, 9, 9, 4, 9, 5, 4, 6, 9, 5, 6, 8, 3, 2, 5, 8, 5,
4, 0, 3, 0, 9, 4, 6, 7, 7, 5, 0, 2, 5, 5, 4, 8, 1, 4, 7, 8, 1, 6,
0, 4, 9, 3, 6, 2, 4, 6, 3, 5, 4, 8, 2, 9, 3, 7, 1, 2, 0, 7, 2, 0,
5, 4, 1, 2, 8, 4, 9, 2, 9, 9, 0, 0, 3, 0, 4, 3, 4, 2, 6, 4, 3, 8,
8, 6, 4, 3, 4, 8, 6, 5, 6, 4, 7, 1, 7, 2, 7, 5, 8, 6, 1, 7, 7, 1,
4, 7, 8, 3, 8, 3, 9, 0, 8, 8, 0, 6, 4, 7, 6, 6, 1, 2, 8, 9, 6, 2,
2, 1])
filename="/content/drive/MyDrive/Colab Notebook/Music Genre Classification/audio/metal.00000.wav"
audio, sample_rate = librosa.load(filename, res_type='kaiser_fast')
mfccs_features = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=40)
mfccs_scaled_features = np.mean(mfccs_features.T,axis=0)
print(mfccs_scaled_features)
mfccs_scaled_features=mfccs_scaled_features.reshape(1,-1)
print(mfccs_scaled_features)
print(mfccs_scaled_features.shape)
predicted_label=model.predict_classes(mfccs_scaled_features)
print(predicted_label)
prediction_class = labelencoder.inverse_transform(predicted_label)
prediction_class


Files
+ Code + Text
RAM
Disk
Editing
Music_Genre_Classification
Data
audio
blues.00000.wav
classical.00000.wav
country.00000.wav
disco.00000.wav
hiphop.00000.wav
jazz.00000.wav
metal.00000.wav
pop.00000.wav
reggae.00000.wav
rock.00000.wav
saved_models
Music_Genre_Classificatio...
kaggle.json
NLP_t_extra
Word_Embedding
iNeuron
Untitled0.ipynb
Indane invoice
Project
Disk
67.84 GB available
print(mfccs_scaled_features)
mfccs_scaled_features=mfccs_scaled_features.reshape(1,-1)
print(mfccs_scaled_features)
print(mfccs_scaled_features.shape)
predicted_label=model.predict_classes(mfccs_scaled_features)
print(predicted_label)
prediction_class = labelencoder.inverse_transform(predicted_label)
prediction_class
[-2.6819553e+02 1.3569197e+02 -3.4051403e+01 2.5615515e+01
-1.2977585e+01 1.7902988e+01 -9.9903488e+00 5.6307025e+00
-1.4142573e+01 7.4903030e+00 -3.9693322e+00 8.2585735e+00
-1.3353267e+00 -3.6505163e-01 1.0393412e+00 3.7913842e+00
-3.2078311e-01 1.7729853e+00 8.8923377e-01 -8.4202534e-01
-2.3087556e+00 1.6774954e-01 4.2183442e+00 6.9622838e-01
-1.0857674e+00 -3.4671383e+00 3.2546456e+00 -4.1347303e+00
-1.1387728e+00 2.3851283e+00 3.7798860e+00 -3.8901453e+00
-3.3207026e+00 -1.2545277e+00 4.1186099e+00 5.9935200e-01
-1.8138249e+00 -1.0350472e+00 -1.4708922e+00 -3.8238471e+00]
[[-2.6819553e+02 1.3569197e+02 -3.4051403e+01 2.5615515e+01
-1.2977585e+01 1.7902988e+01 -9.9903488e+00 5.6307025e+00
-1.4142573e+01 7.4903030e+00 -3.9693322e+00 8.2585735e+00
-1.3353267e+00 -3.6505163e-01 1.0393412e+00 3.7913842e+00
-3.2078311e-01 1.7729853e+00 8.8923377e-01 -8.4202534e-01
-2.3087556e+00 1.6774954e-01 4.2183442e+00 6.9622838e-01
-1.0857674e+00 -3.4671383e+00 3.2546456e+00 -4.1347303e+00
-1.1387728e+00 2.3851283e+00 3.7798860e+00 -3.8901453e+00
-3.3207026e+00 -1.2545277e+00 4.1186099e+00 5.9935200e-01
-1.8138249e+00 -1.0350472e+00 -1.4708922e+00 -3.8238471e+00]]
(1, 40)
[1]
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/sequential.py:455: UserWarning: `model.predict_classes()` is depr
warnings.warn('`model.predict_classes()` is deprecated and '
array(['classical'], dtype='<U9')

8. ADVANTAGES AND DISADVANTAGES

For MCC

Advantages	Disadvantages
• The recognition accuracy is high. That means the performance rate of MFCC is high.	• In background noise MFCC does not give accurate results.
• MFCC captures main characteristics of phones in speech.	• The filter bandwidth is not an independent design parameter.
• Low Complexity.	• Performance might be affected by the number of filters.

For K. K - Nearest Neighbour

Advantages	Disadvantages
• Robust to noisy training data (especially if we use inverse square of weighted distance as the "distance").	• Need to determine value of parameter K (number of nearest neighbors).
• Effective if the training data is large.	• Distance based learning is not clear which type of distance to use and which attribute to use to produce the best results.
• No assumptions about the characteristics of the concepts to learn have to be done.	• Computation cost is quite high because we need to compute distance of each query instance to all training samples. Some indexing (e.g. K-D tree) may reduce this computational cost.

8. APPLICATIONS

Genre classification is an important task with many real world applications. As the quantity of music being released on a daily basis continues to sky-rocket, especially on internet platforms such as Soundcloud and Spotify – a 2016 number suggests that tens of thousands of songs were released every month on Spotify – the need for accurate meta-data required for database management and search/storage purposes climbs in proportion. Being able to instantly classify songs in any given playlist or library by genre is an important functionality for any music

streaming/purchasing service, and the capacity for statistical analysis that correct and complete labeling of music and audio provides is essentially limitless.

9. CONCLUSION:

CNN model was able to classify “classical” genre music with the highest F1-score. CNN performed worst for “Rock” and “reggae” genre music.We used the transfer learning technique to improve the performance of genre classification. Figure presents the results of the transfer learning-based model on test data. F1-score for “hiphop”, “jazz”, and “pop” genres increased due to transfer learning. If we look at overall results, we have achieved only a minor improvement after applying transfer learning.

We have used both spectrogram and wavelet data to train the multimodal-based model. In the same way, we perform the testing. We have found very surprising results. Instead of improvement, our performance reduced drastically. We have achieved only 38% of F1-score while using a multi-modal approach

10. FUTURE SCOPE

Our project makes a basic attack on the music genre classification problem, but could be extended in several ways. Our work doesn’t give a completely fair comparison between learning techniques for music genre classification. Adding a validation step to the DAG SVM would help determine which learning technique is superior in this application. We used a single feature (MFCCs) throughout this project. Although this gives a fair comparison of learning algorithms, exploring the effectiveness of different features (i.e. combining with metadata from ID3 tags) would help to determine which machine learning stack does best in music classification. Since genre classification between fairly different genres is quite successful, it makes sense to

attempt finer classifications. The exact same techniques used in this project could be easily extended to classify music based on any other labelling, such as artist. In addition, including additional metadata text features such as album, song title, or lyrics could allow us to extend this to music mood classification as well. In regards to the image matching extension, there is room for further development in obtaining a more varied data set of images (instead of four rough image ”themes” such as nature for classical or pop artists for pop), although quantifying results is again an inherently subjective exercise. Another possible application of the music-image mapping is to auto-generate a group of suitable images for any given song, possibly replacing the abstract color animations in media players and manual compilations in Youtube videos.

11. BIBILOGRAPHY

REFERENCES:

- https://www.analyticsvidhya.com/blog/2021/06/music-genres-classification-using-deep-learning-techniques/
- https://www.clairvoyant.ai/blog/music-genre-classification-using-cnn#:~:text=It%20aims%20to%20predict%20the,and%20then%20select%20the%20genre.
- https://ijcsma.com/publications/february2018/V6I217.pdf
- https://www.kaggle.com/andradaolteanu/gtzan-dataset-music-genre-classification