

Table of Contents

INTRODUCTION	2
OVERVIEW	2
PURPOSE	2
LITERATURE ANALYSIS	3
EXISTING PROBLEM	3
PROPOSED SOLUTION	4
THEORITICAL ANALYSIS.....	7
BLOCK DIAGRAM	7
HARDWARE/SOFTWARE DESIGNING	7
EXPERIMENTAL INVESTIGATION	8
FLOWCHART	13
RESULT	13
ADVANTAGES & DISADVANTAGES	16
APPLICATIONS	16
CONCLUSION	17
FUTURE SCOPE	17
BIBILOGRAPHY	18

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

PROPOSED SOLUTION:

Kullback-Lieber (KL):

Divergence The fundamental calculation in our k-NN training is to figure out the distance between two songs. We compute this via the Kullback-Leibler divergence . Consider $p(x)$ and

$q(x)$ to be the two multivariate Gaussian distributions with mean and covariance corresponding to those derived from the MFCC matrix for each song. Then, we have the following:

$$2KL(p||q) = \log \frac{|\Sigma_q|}{|\Sigma_p|} + Tr(\Sigma_q^{-1}\Sigma_p) + (\mu_p - \mu_q)^T \Sigma_q^{-1}(\mu_p - \mu_q) - d$$

However, since KL divergence is not symmetric but the distance should be symmetric

$$D_{KL}(p, q) = KL(p||q) + KL(q||p)$$

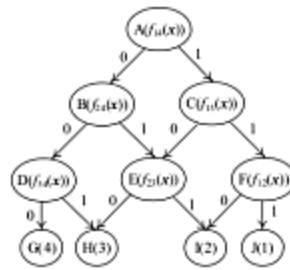
k-Nearest Neighbors (k-NN):

The first machine learning technique we applied is the k-nearest neighbors (k-NN) because existing literature has shown it is effective considering its ease of implementation. The class notes on knearest neighbors gave a succinct outline of the algorithm which served as our reference.

k-Means:

For unsupervised k-means clustering to work on our feature set, we wrote a custom implementation because we had to determine how to represent cluster centroids and how to update to better centroids each iteration. To solve this, we chose to represent a centroid as if it were also a multi-variate Gaussian distribution of an arbitrary song (which may not actually exist in the data set), and initialized the four centroids as four random songs whose distances (as determined by KL divergence) were above a certain empirically determined threshold. Once a group of songs is assigned to a centroid, the centroid is updated according to the mean of the mean vectors and covariance matrices of those 2 songs, thus represented as a new song that is the average of the real songs assigned to it. Finally, as random initialization in the beginning and number of iterations are the two factors with notable influence on the cluster outcomes, we determined the iteration number empirically and repeatedly run k-means with different random initial centroids and pick the best, as determined by the calculated total percent accuracy.

Multi-Class Support Vector Machine (DAG SVM):



A directed acyclic graph of 2-class SVMs

A directed acyclic graph of 2-class SVMs SVM classifiers provide a reliable and fast way to differentiate between data with only two classes. In order to generalize SVMs to data falling into multiple classes (i.e. genres) we use a directed acyclic graph (DAG) of two-class SVMs trained on each pair of classes in our data set. We then evaluate a sequence of twoclass SVMs and use a process of elimination to determine the output of our multi-class classifier.

Neural Networks:

We tried neural networks because it has proved generally successful in many machine learning problems. We first pre-process the input data by combining the mean vector and the top half of the covariance matrix (since it is symmetric) into one feature vector. As a result, we get $15 + (1+15) * 15 = 255$ features for each song.

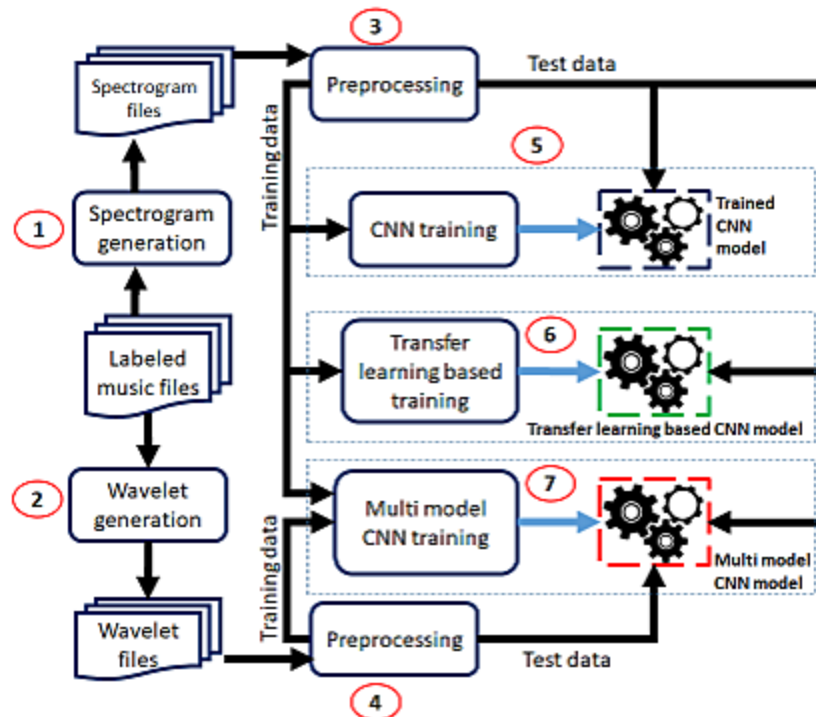
Genre	Classical	Jazz	Metal	Pop
Vector	(1, 0, 0, 0)	(0, 1, 0, 0)	(0, 0, 1, 1)	(0, 0, 0, 1)

We then process the output data by assigning each genre to an element in the set of the standard orthonormal basis in R^4 for our four genres, as shown in the table below: We then split the data randomly by a ratio of 70-15-15: 70% of the data was used for training our neural network, 15% of the data was used for verification to ensure we dont over-fit, and 15% of the data for testing. After multiple test runs, we found that a feedforward model with 10 layers for our neural network model gives the best classification results.

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

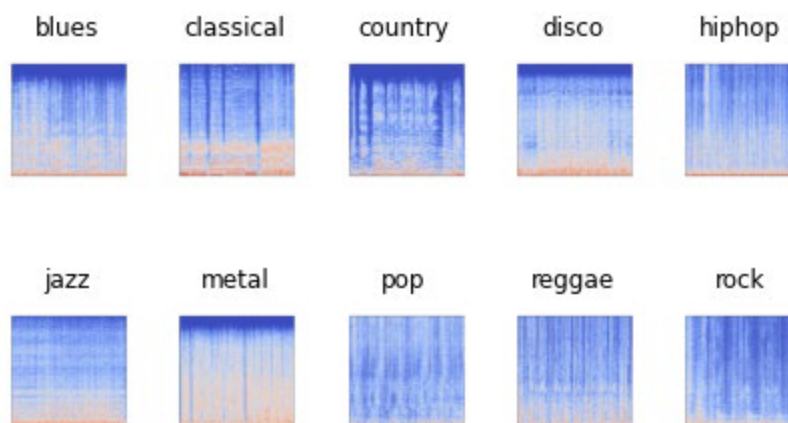
Python 2.7 or above

Jupyter Notebook.

4. EXPERIMENTAL INVESTIGATIONS

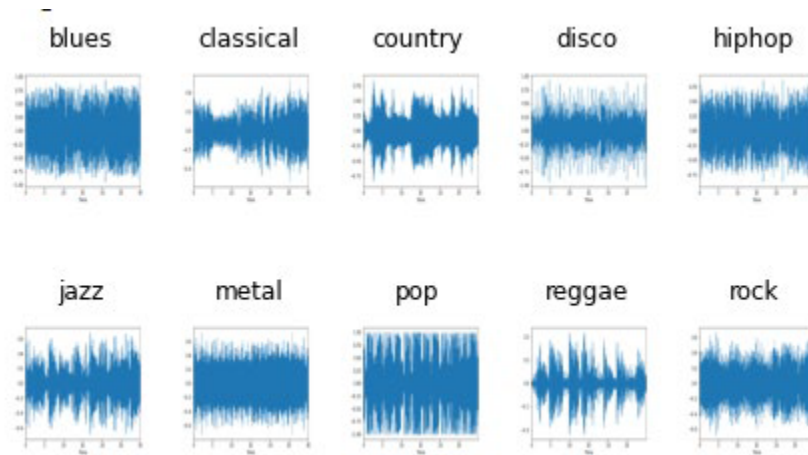
Spectrogram generation:

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:

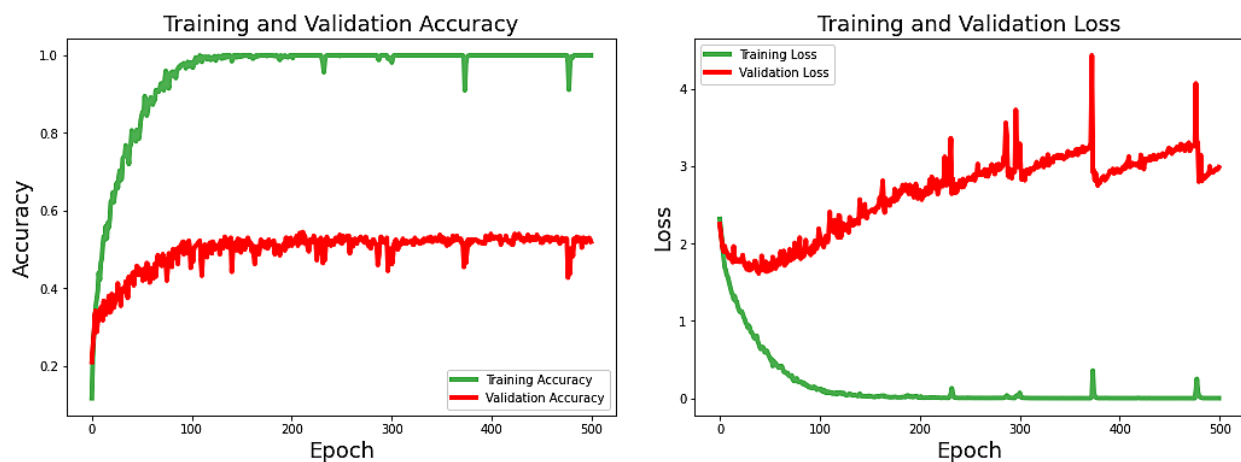
It is clear that we treat our data as image data. After generating spectrograms and wavelets, we apply general image preprocessing steps to generate training and testing data. Each image is of size (256, 256, 3).

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 (MaxPooling2D)	(None, 64, 64, 32)	0
conv2d_2 (Conv2D)	(None, 64, 64, 64)	18496
max_pooling2d_2 (MaxPooling2D)	(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:

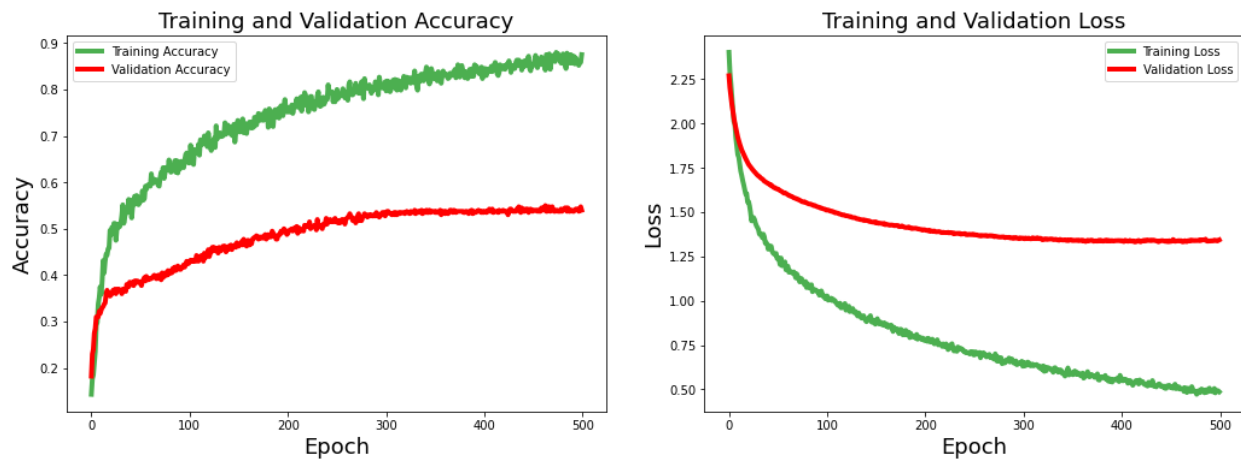
We have only 60 samples of each genre for training. In this case, transfer learning could be a

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 (Func	(None, 8, 8, 1280)	2257984
global_average_pooling2d (G	(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.

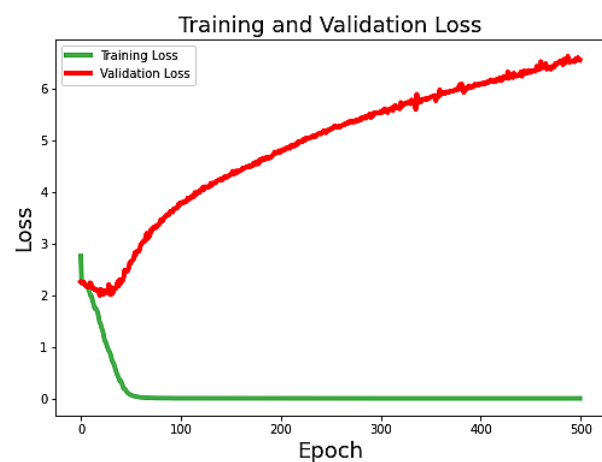
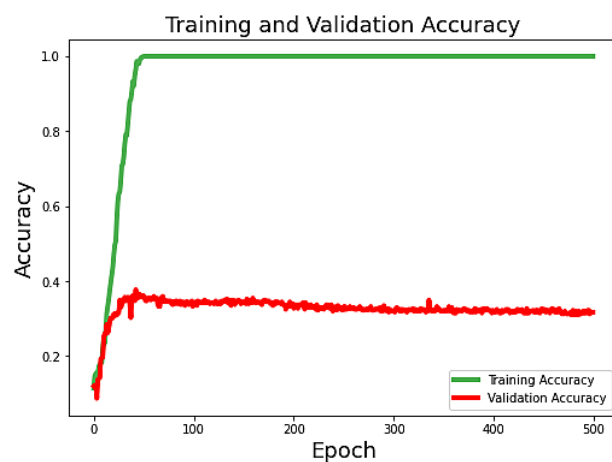


Multimodal training:

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

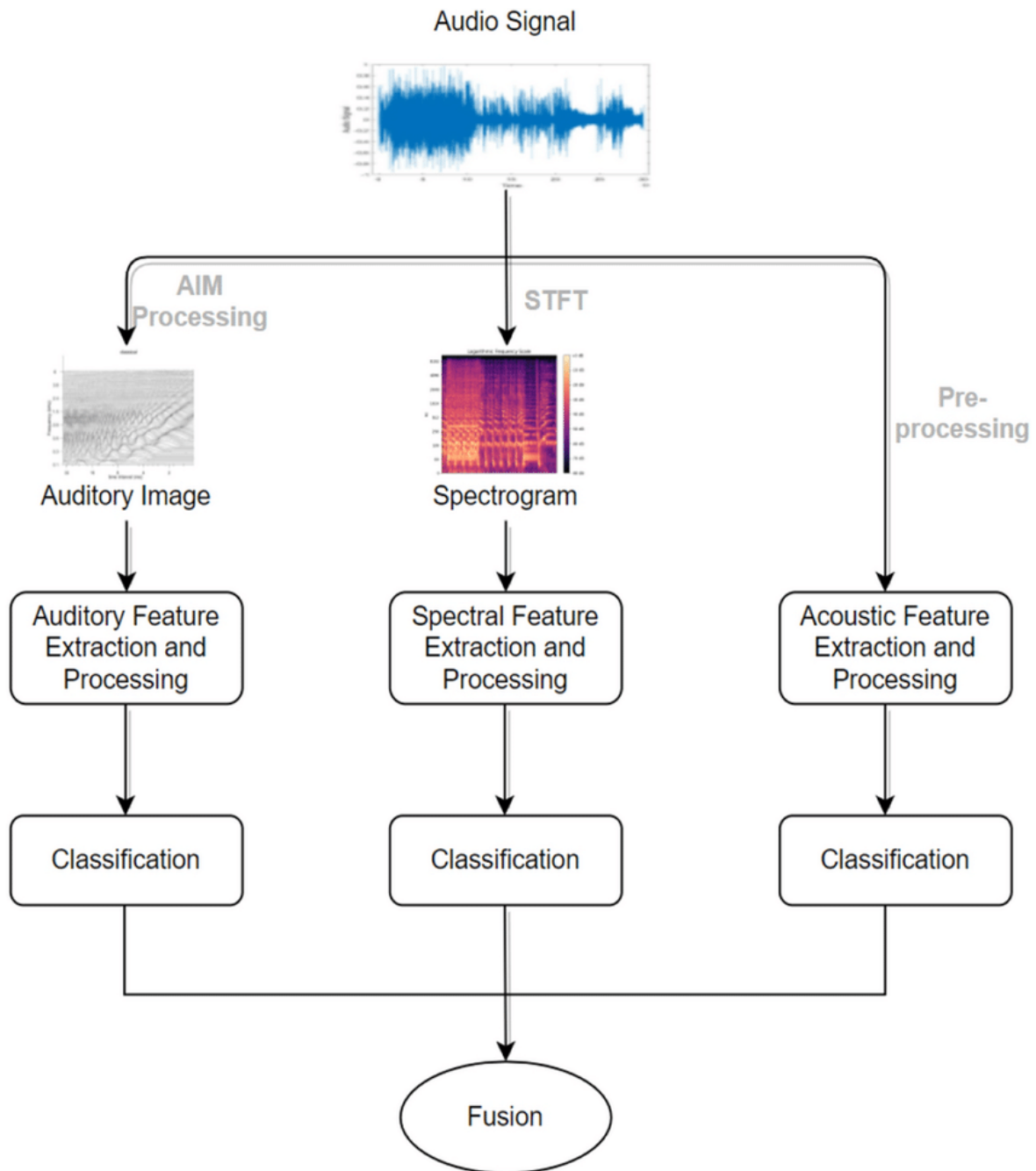
of the model with respect to epochs.

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. 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

12

Classical

Country

Disco

Hip-hop

Jazz

Metal

Pop

Reggae

Rock

The dataset has the following folders:

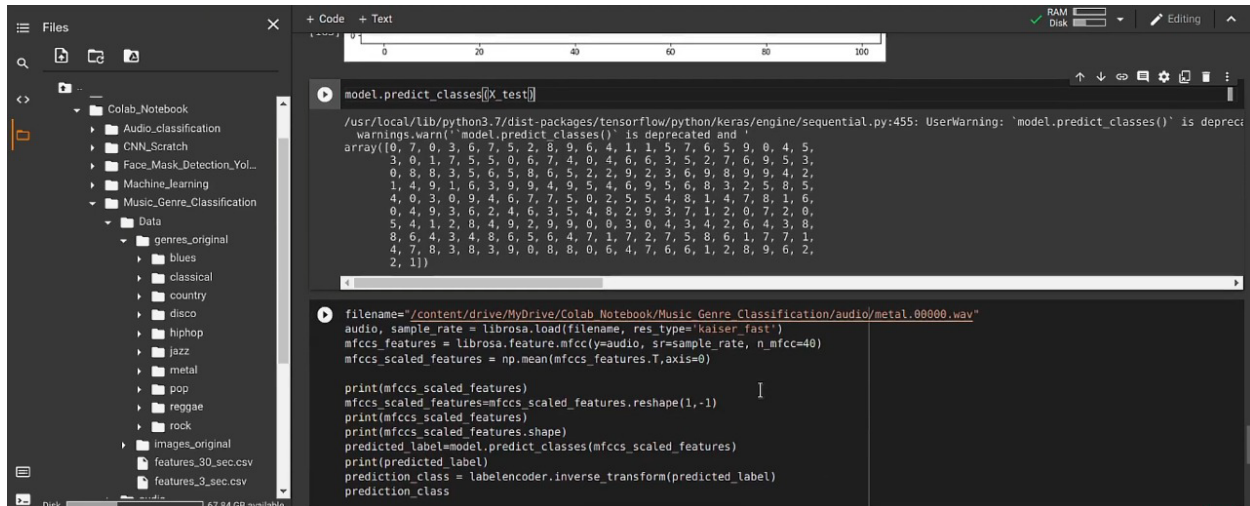
Genres original — A collection of 10 genres with 100 audio files each, all having a length of 30 seconds (the famous GTZAN dataset, the MNIST of sounds)

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.

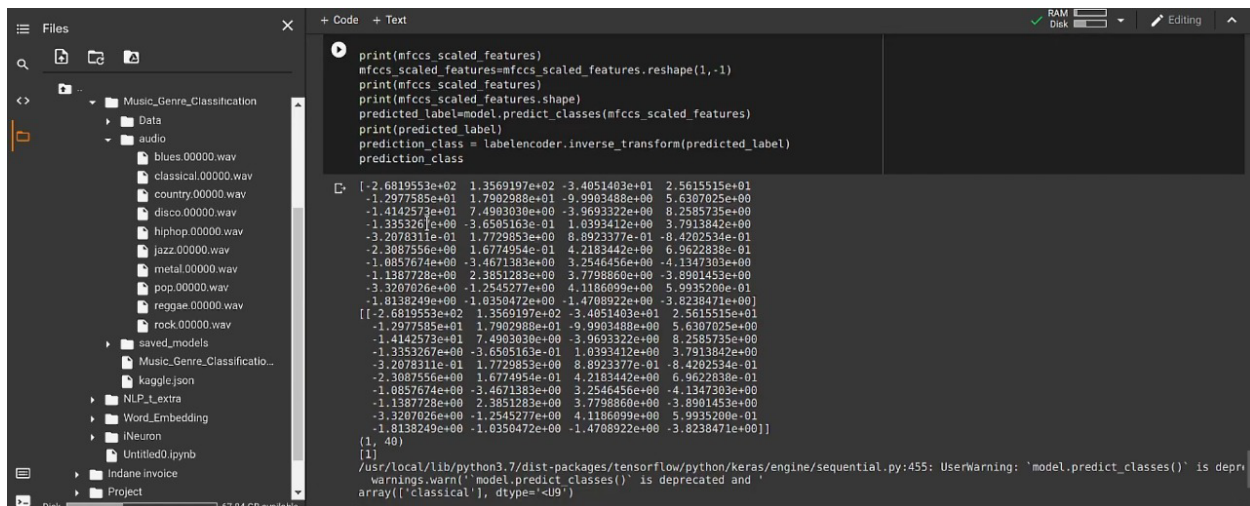
2 CSV files — Containing features of the audio files. One file has for each song (30 seconds long) a mean and variance computed over multiple features that can be extracted from an audio file. The other file has the same structure, but the songs are split before into 3 seconds audio

files.

From here the outputs



The screenshot shows a Jupyter Notebook interface. On the left, the 'Files' sidebar displays a directory structure: Colab_Notebook, Audio_classification, CNN_Scratch, Face_Mask_Detection_Vol..., Machine_Learning, and Music_Genre_Classification. Under Music_Genre_Classification, there are subfolders 'Data' and 'genres_original', and files 'features_30_sec.csv' and 'features_3_sec.csv'. The main code area contains two cells. The first cell executes `model.predict_classes([X_test])`, resulting in a long array of predicted class indices and a warning about the deprecation of `model.predict_classes()`. The second cell loads an audio file `metal.00000.wav`, processes it with MFCC features, and prints the scaled features and the predicted class label, which is 'metal'.



The screenshot shows a Jupyter Notebook interface. The 'Files' sidebar on the left shows the 'Music_Genre_Classification' folder expanded, revealing an 'audio' subfolder with various .wav files (e.g., 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) and a 'saved_models' folder. The code area contains two cells. The first cell prints the MFCC scaled features and the predicted class label, which is 'metal'. The second cell prints the scaled features and the predicted class label, which is 'metal'. A warning about the deprecation of `model.predict_classes()` is visible at the bottom.

7. ADVANTAGES AND DISADVANTAGES

For MCC

Advantages	Disadvantages
<ul style="list-style-type: none">• The recognition accuracy is high. That means the performance rate of MFCC is high.	<ul style="list-style-type: none">• In background noise MFCC does not give accurate results.
<ul style="list-style-type: none">• MFCC captures main characteristics of phones in speech.	<ul style="list-style-type: none">• The filter bandwidth is not an independent design parameter.
<ul style="list-style-type: none">• Low Complexity.	<ul style="list-style-type: none">• Performance might be affected by the number of filters.

For K. K - Nearest Neighbour

Advantages	Disadvantages
<ul style="list-style-type: none">• Robust to noisy training data (especially if we use inverse square of weighted distance as the "distance").	<ul style="list-style-type: none">• Need to determine value of parameter K (number of nearest neighbors).
<ul style="list-style-type: none">• Effective if the training data is large.	<ul style="list-style-type: none">• Distance based learning is not clear which type of distance to use and which attribute to use to produce the best results.
<ul style="list-style-type: none">• No assumptions about the characteristics of the concepts to learn have to be done.	<ul style="list-style-type: none">• 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:

Our algorithms performed fairly well, which is expected considering the sharply contrasting genres used for testing. The simpler and more naive approaches, k-NN (supervised) and k-Means (unsupervised), predictably did worse than the more sophisticated neural networks (supervised) and SVMs (unsupervised). However, we expected similar performance from SVMs and neural networks based on the papers we read, so the significant superiority of neural networks came as a surprise. A large part of this is probably attributable to the rigorous validation we used in neural networks, which stopped training exactly at the maximal accuracy for the validation data set. We performed no such validation with our DAG SVM. Most learning algorithms had the most difficulty differentiating between Metal and Jazz, except k-Means which had the most difficulty differentiating between Classical and Jazz. This corroborates the idea that qualitatively these genres are the most similar. Our image matching results can be considered reasonable from human perception but due to the subjective and nebulous nature of image-to-music-genre clusters, we found roughly 40% overlap of image types with any two given image clusters

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. BIBLIOGRAPHY

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>