Music Genre Recognition

Meenal Jain
MT21050

Prakhar Kumar
MT21066

Amuru Hareesh
MT21009

1 Introduction

Music Genre Classification can be used to identify different types of genres of music. Music has different types of genres like hip-hop, rock, country, etc. And some genres may overlap with one another. Due to this, it is very difficult to put music into one genre of music as it may have parts that belong to another genre of music. But grouping the music into one genre is important because each individual is different and has a different taste in music. By grouping, we can allow users to listen to the music which they like. It will take audio file as input and classify the genre of that input. FFT and Mel Spectrum features will be used to convert audio files to feature vectors. For FFT and MEL we remove first 10% and last10% of the audio files as starting and ending of an audio file won't provide much info about the music. After getting the features we will train multiple models (classical + state of the art) and compare the performance on test data.

2 Problem Statement

The problem is to develop an ML/DL application that can classify music into different genres.

3 Dataset

GTZAN Dataset has 1000 songs and 10 classes/genres. Each genre has 100 songs of 30 seconds each. There are 10 genres of music, each genre will be given a number from 0 through 9. All the tracks are in .wav format.
Classes/MusicGenres:
reggae, pop, classical, disco, jazz, metal, hip-hop, rock, country, blues - 100 Music Files for each.
Number of samples - 1000, Total classes - 10.
Link - https://github.com/Hguimaraes/gtzan.keras
The sampling rate(frequency) for each genre is 22050.

4 Literature Survey

(1) Convolutional Neural Networks Approach for Music Genre Classification This paper has applied CNN's advantages and characteristics in audio to implement a music genre, classification model. They have used Librosa to convert audio files into their corresponding Mel spectrums as a preprocessing part.Their working principle is to obtain the MFCC as preprocessed data. Then the preprocessed training data is used to propose a CNN model for training. This paper is composed of 5 convolutional layers. The preprocessed spectrogram is used as input and sent to the CNN model. They have used a kernel size of 3 with a stride of 1 and the activation function as ReLu, the max-pooling layer as (2, 2), and the dropout as 0.5. The average accuracy obtained on their proposed CNN model is 84% compared with three different models.

(2) Music Genre Classification: This paper built several segmentation models and trained them on the GTZAN database. This paper proposes a machine learning project to separate the type of music provided in the form of an audio file with the help of the KNN algorithm (K neighbor). Mel Frequency Cepstral Coefficients (MFCC) is used to represent time-domain formats as a few domain coefficients and also diminish this matrix description of every song by taking the vector and covariance matrix of Cepstral elements over every 20microseconds frame and keeping it as a cell-matrix. Their paper has suggested an easier way to solve the problem of separation and draw an easy to upgrade KNN model which produces acceptable results around 80% compared to the CNN model.

(3) 1D CNN Architecture for music classification The paper proposes a residual CNN architecture. The paper first uses a sliding window of size

5 seconds to make predictions. The audio file is broken into several such windows and a CNN is trained on each window. Training CNN on windows allows for the prediction of audio in audio streams as well. A prediction can be made on individual segments or taken by a majority vote over all segments for an audio file. The CNN Architecture proposed a resnet block as 1D CNN followed by Batch Normalization followed by Leaky ReLU followed by 1D Conv and again batch normalization. Which is then added to residual skip connect.

(4) Music Genre Classification with Parallel Recurrent CNN The idea for the paper is that CNN isn't able to model long-term relationships in audio files, so they combined CNNs with RNNs. The paper proposes an architecture with a CNN Block and a BiDirectional RNN Block. The hybrid structure not only models the spatial structure using CNN but also temporal features using RNNs. The features from audio files are generated using STFT and then max pooled and convolutions are performed in parallel for both the blocks (CNN and RNN). the output of both the blocks is concatenated and a softmax layer is added for output.

(5) Deep learning-based music genre classification using spectrogram This paper has proposed a CNN-based neural network for the music genre classification using spectrogram. The authors have performed the training on the GTZAN dataset. The data from the GTZAN dataset is split into train and test in the 80-20 ratio. The dataset has ten classes of music genres such as blues, classical, country, rock, disco, reggae, hip-hop, jazz, metal, and pop—data Processing and extracting the feature values from the slices of songs using Librosa python library. The Librosa python library performs feature extraction using different signal processing techniques. In feature extraction, extracting meaningful features from slices of spectrograms such as Mel-spectrograms, Spec- centroid, Spectral roll-off, Zero-crossing rate, Spectral bandwidth, and Chromo frequencies.
A two-dimensional CNN is implemented, and it takes a binary version of spectrogram features with input shape 128x128x1 in which they have used only one channel. After the max-pooling (2,2), (64x64x1) is input to the next layer, and after max-pooling (2,2), then the output will be (32x32x1). There are five convolutional layers with the kernel size 2x2 with a stride of two and a max-pooling layer implemented after each successive layer. After CNN layers, there is a fully connected layer where each output of the previous layer feeds into the input of the fully connected layer. They have used the softmax function at the end of the output layer. They have used adam optimizer and ReLU activation function. ReduceLRonPlateau reduces the learning rate when a matric has stopped improving. The model is trained with different batch sizes with 200 epochs. As the batch size increases, the performance also increases. The model performs with above 90 % accuracy for the GTZAN test dataset.

(6) Deep learning for music genre classification In this paper, they have used the Restricted Boltzmann machine algorithm to build deep belief neural networks for the music genre classification. Training a deep neural network with the traditional back-propagation algorithm can be stuck in local minima. They have used a restricted Boltzmann machine algorithm for the pre-training for better performance. One needs to find the parameters that minimize the energy of the configuration of parameters, which can be done by reducing the negative log-likelihood of the model. Minimizing the negative log-likelihood is the same as minimizing the free energy and finding the system's equilibrium state in the restricted Boltzmann machine(RBM). They have used the GTZAN dataset for the music genre, which has 1000 audio tracks. In the feature extraction, they have extracted the Mel frequency Cepstral Coefficient (MFCC). They have used hamming window, Ftheier transforms, Discrete cosine transforms for MFCC. They got 2600x 13 features for each sample. Further, they have reduced to 13 x 160 =2080 length of MFCC feature. Then, they have connected RBM with a multilayer network. The network consists of three hidden layers, they RBMs for each network layer except the output layer as the weight initialization. The key idea is to train RBMs iteratively between layers and stack them together on the multilayer architecture in the end. They have used Contrastive Divergence learning to update the parameters. They have used 60 % MFCC features for training and 40 % for testing. After the experimental results, it is clear that with a larger dataset Deep belief neural network improves more than a neural network and gradually outperforms it.

5 Methodology

In the methodology, first, we loaded the audio files using librosa and generated the features using Librosa. We have used two kinds of features, i.e., FFT(Fast Fourier Transform)and MFCC (Mel-frequency cepstral coefficients) for the experiments. Then, the models were trained on these features

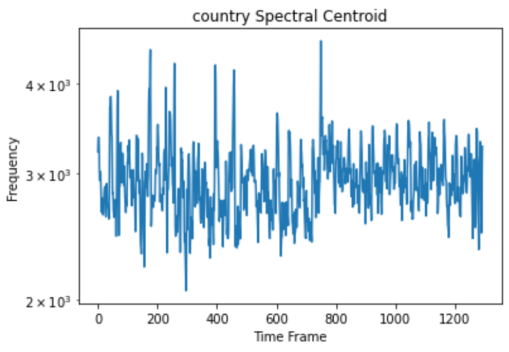


Fig 1: Spectral Centroid for country class (Data Visualization)

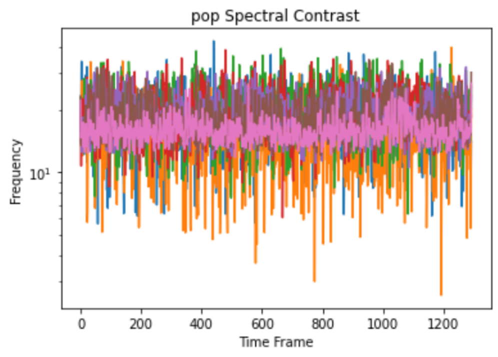


Fig 2: Spectral Contrast for pop class (Data Visualization)

5.1 Data Visualization

Different spectral features can extract using digital signal processing to characterize a spectrum. We have used Librosa's python library to extract a few features like spectral centroid, spectral roll-off, spectral bandwidth, and spectral contrast. The above figures show different spectral features for different classes.

The spectral centroid indicates where the center of mass is located in the spectrum. This feature gives importance to the impression of the brightness of a sound. Spectral roll-off tells the spectrum frequency below which a specified percentage of the total spectral energy, generally 85 percent of the complete spectrum range. Spectral Bandwidth is based on spectral centroid, which computes the order p-spectral Bandwidth. Here, p represents peaks in the spectrum. Spectral contrast considers the spectral peak, the spectral valley, and the spectral peak and valley difference in each frequency subband.

5.2 Feature Extraction

Fourier Transform decomposes a periodic sound into a sum of sine waves that all vibrate and oscillate at different frequencies. With Fourier Transform, we can describe a very complex sound as long as it's periodic as a sum as the superimposition of varying sine waves at different frequencies. We used the scipy library to generate FFT-based features. We take the top 1000 values returned by the FFT function for each music.

MFCC (Mel Frequency Cepstral Spectogram) features capture many aspects of sound that can approximate the human auditory system. The way humans perceive sound and frequency is different from the machine.MFCC can match that to the human auditory system, which is very helpful in deep learning models. The MFCC features generally take 13 to 39 coefficients for the deep learning experiments from the MFCC vector. We also have considered 13 coefficients from the MFCC vector. We broke down the music into 3 second time intervals, extracted the MFCC feature vector, and took the mean across time intervals.

5.3 Architecture

After several experiments, we propose a CNN architecture for music genre recognition. The CNN architecture takes the MFCC features and reshapes that to fit in the convolutional network. The network starts with the Conv2D(128, (3,3)) layer, followed by MaxPooling2D(3,3). Conv2D(256, (3,3)),MaxPooling2D(3,3), Conv2D(256, (3,3)) and GlobalAveragePooling2D are the sequence of layers used from the previous layer. This layer connects to the sequence of fully connected layers of sizes 512,256,128, and 10. We used to drop out of 0.3 in all the convolution and dense layers with reLu as the activation function. We have used the Adam optimizer with a learning rate of 0.001 and considered categorical crossentropy as a loss function. We trained the model with 30 epochs of batch size 128 and got a testing accuracy of 81 percent.

6 Experiments

Before getting into the final CNN model, we did several experiments with the FFT and MFCC features on different models. We started with the Logistic Regression with different learning rates [0.00001, 0.0001, 0.001, 0.01, 0.1, 1] of 500 iterations each, Support Vector Classifier with different learning rates, Decision Tree with different maximum depths [3, 5, 7, 9, 11, 13, 17, 25, 50], Random Forest Classifier with different maximum depths [3, 5, 7, 9] and iterations [300, 500, 700, 900], XGBoost classifier with different maximum depths [3, 5, 7, 9] and multi-layer perceptron(1000,512,256,128,10) with batch size 64 and Adam optimizer. We experimented with all the above models with FFT features and MFCC features. Although we tried different models, we couldn't get decent results.

We started with LSTM with 64 dimensions as output with deep learning models and used Adam optimizer and sparse categorical cross-entropy as a loss with batch size 128. We have considered MFCC features for all the deep learning models. We extended our experiments to Gated Recurrent Unit and Bi-Directional GRU and found better results with these RNN models than with the classification models. Before proposing the final CNN model, we have experimented with two other CNN models with a similar setup to the proposed model but with different convolutions like Conv2D(64, (3,3)), Conv2D(128, (3,3)), and different max poolings. Although these models performed decently, the proposed architecture starting with Conv2D(128, (3,3)) followed by various convolutions like Conv2D(256, (3,3)), max-pooling layers, and dense layers outperformed all the other deep learning models.

7 Results and Analysis

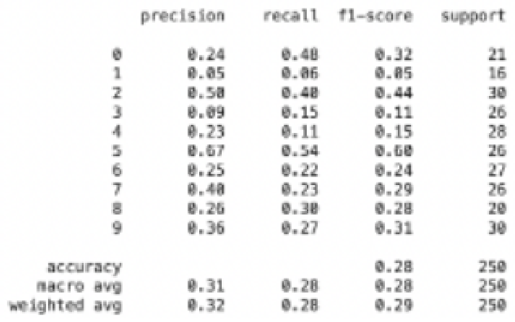
We have trained different machine and deep learning models to train the music acoustic features. The following are the results shown below.

FFT Featurizations Results

1. Decision Tree Classifier: Obtained an average accuracy of 28%. See Figure 3.

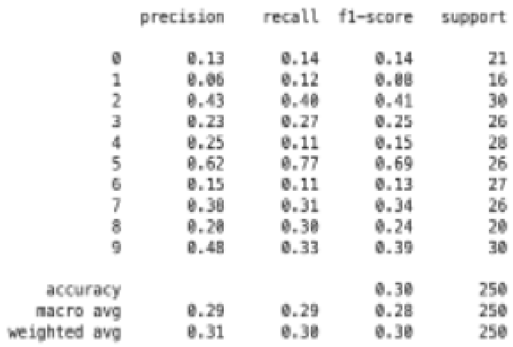
2. Logistics Regression: Obtained an average accuracy of 30%. See Figure 4.

3. Random Forest: Obtained an average accuracy of 44%. See Figure 5.



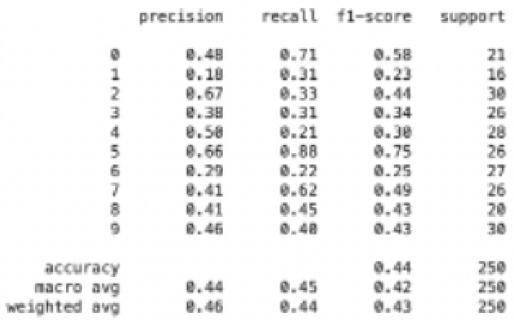
```
              precision    recall  f1-score   support

           0       0.24      0.48      0.32        21
           1       0.05      0.06      0.05        16
           2       0.50      0.40      0.44        30
           3       0.09      0.15      0.11        26
           4       0.23      0.11      0.15        28
           5       0.67      0.54      0.60        26
           6       0.25      0.22      0.24        27
           7       0.40      0.23      0.29        26
           8       0.26      0.30      0.28        20
           9       0.36      0.27      0.31        30

    accuracy                           0.28       250
   macro avg       0.31      0.28      0.28       250
weighted avg       0.32      0.28      0.29       250
```

Fig 3: FFT Featurization - Decision Tree



```
              precision    recall  f1-score   support

           0       0.13      0.14      0.14        21
           1       0.06      0.12      0.08        16
           2       0.43      0.40      0.41        30
           3       0.23      0.27      0.25        26
           4       0.25      0.11      0.15        28
           5       0.62      0.77      0.69        26
           6       0.15      0.11      0.13        27
           7       0.38      0.31      0.34        26
           8       0.20      0.30      0.24        20
           9       0.48      0.33      0.39        30

    accuracy                           0.30       250
   macro avg       0.29      0.29      0.28       250
weighted avg       0.31      0.30      0.30       250
```

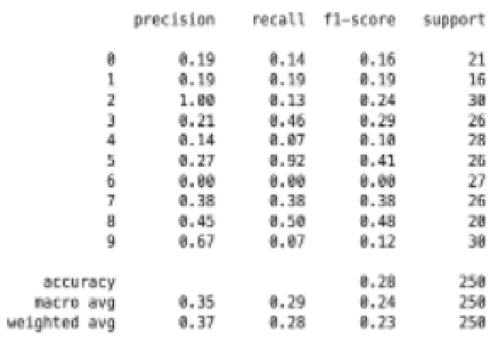
Fig 4: FFT Featurization - Logistic Regression



```
              precision    recall  f1-score   support

           0       0.48      0.71      0.58        21
           1       0.18      0.31      0.23        16
           2       0.67      0.33      0.44        30
           3       0.38      0.31      0.34        26
           4       0.50      0.21      0.30        28
           5       0.66      0.88      0.75        26
           6       0.29      0.22      0.25        27
           7       0.41      0.62      0.49        26
           8       0.41      0.45      0.43        20
           9       0.46      0.40      0.43        30

    accuracy                           0.44       250
   macro avg       0.44      0.45      0.42       250
weighted avg       0.46      0.44      0.43       250
```

Fig 5: FFT Featurization - Random Forest

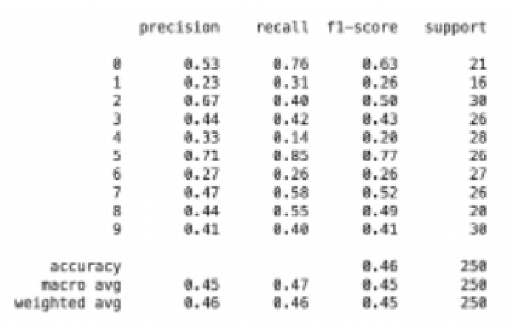
4. Support Vector Classifier: Obtained an average accuracy of 28%. See Figure 6.



```
              precision    recall  f1-score   support

           0       0.19      0.14      0.16        21
           1       0.19      0.19      0.19        16
           2       1.00      0.13      0.24        30
           3       0.21      0.46      0.29        26
           4       0.14      0.07      0.10        28
           5       0.27      0.92      0.41        26
           6       0.00      0.00      0.00        27
           7       0.38      0.38      0.38        26
           8       0.45      0.50      0.48        20
           9       0.67      0.07      0.12        30

    accuracy                           0.28       250
   macro avg       0.35      0.29      0.24       250
weighted avg       0.37      0.28      0.23       250
```

Fig 6: FFT Featurization - Support Vector Classifier

5. XGBoost Classifier: Obtained an average accuracy of 46%. See Figure 7.

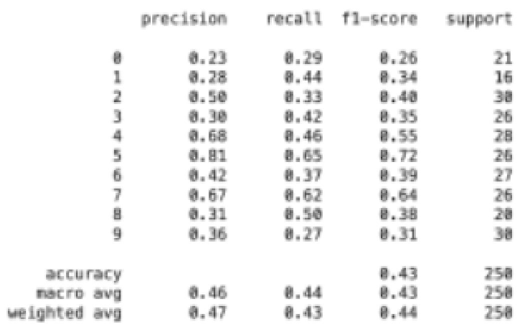


```
              precision    recall  f1-score   support

           0       0.53      0.76      0.63        21
           1       0.23      0.31      0.26        16
           2       0.67      0.40      0.50        30
           3       0.44      0.42      0.43        26
           4       0.33      0.14      0.20        28
           5       0.71      0.85      0.77        26
           6       0.27      0.26      0.26        27
           7       0.47      0.58      0.52        26
           8       0.44      0.55      0.49        20
           9       0.41      0.40      0.41        30

    accuracy                           0.46       250
   macro avg       0.45      0.47      0.45       250
weighted avg       0.46      0.46      0.45       250
```

Fig 7: FFT Featurization - XGBoost Classifier

MFCC Featurizations Results

1. Decision Tree Classifier: Obtained an average accuracy of 43%. See Figure 8.

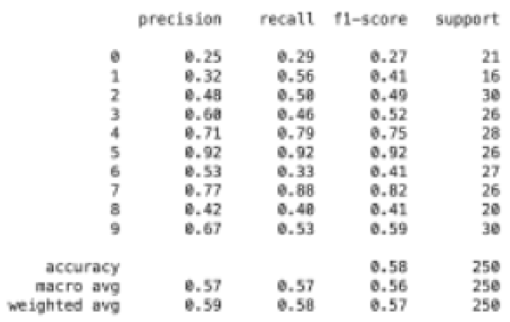


```
              precision    recall  f1-score   support

           0       0.23      0.29      0.26        21
           1       0.28      0.44      0.34        16
           2       0.50      0.33      0.40        30
           3       0.30      0.42      0.35        26
           4       0.68      0.46      0.55        28
           5       0.81      0.65      0.72        26
           6       0.42      0.37      0.39        27
           7       0.67      0.62      0.64        26
           8       0.31      0.50      0.38        20
           9       0.36      0.27      0.31        30

    accuracy                           0.43       250
   macro avg       0.46      0.44      0.43       250
weighted avg       0.47      0.43      0.44       250
```

Fig 8: MFCC Featurization - Decision Tree

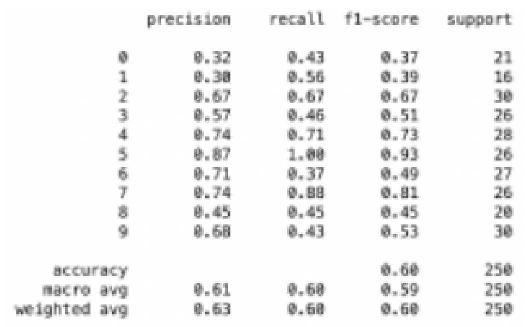
2. Logistics Regression: Obtained an average accuracy of 50%. See Figure 9.

3. Random Forest: Obtained an average accuracy of 60%. See Figure 10.



```
              precision    recall  f1-score   support

           0       0.25      0.29      0.27        21
           1       0.32      0.56      0.41        16
           2       0.48      0.50      0.49        30
           3       0.60      0.46      0.52        26
           4       0.71      0.79      0.75        28
           5       0.92      0.92      0.92        26
           6       0.53      0.33      0.41        27
           7       0.77      0.88      0.82        26
           8       0.42      0.40      0.41        20
           9       0.67      0.53      0.59        30

    accuracy                           0.58       250
   macro avg       0.57      0.57      0.56       250
weighted avg       0.59      0.58      0.57       250
```

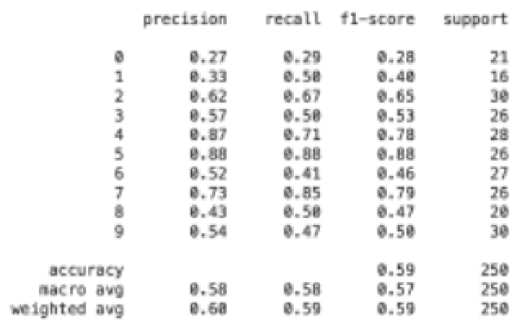
Fig 9: MFCC Featurization - Logistic Regression



```
              precision    recall  f1-score   support

           0       0.32      0.43      0.37        21
           1       0.30      0.56      0.39        16
           2       0.67      0.67      0.67        30
           3       0.57      0.46      0.51        26
           4       0.74      0.71      0.73        28
           5       0.87      1.00      0.93        26
           6       0.71      0.37      0.49        27
           7       0.74      0.88      0.81        26
           8       0.45      0.45      0.45        20
           9       0.68      0.43      0.53        30

    accuracy                           0.60       250
   macro avg       0.61      0.60      0.59       250
weighted avg       0.63      0.60      0.60       250
```

Fig 10: MFCC Featurization - Random Forest

4. Support Vector Classifier: Obtained an average accuracy of 59%. See Figure 11.



```
              precision    recall  f1-score   support

           0       0.27      0.29      0.28        21
           1       0.33      0.50      0.40        16
           2       0.62      0.67      0.65        30
           3       0.57      0.50      0.53        26
           4       0.87      0.71      0.78        28
           5       0.88      0.88      0.88        26
           6       0.52      0.41      0.46        27
           7       0.73      0.85      0.79        26
           8       0.43      0.50      0.47        20
           9       0.54      0.47      0.50        30

    accuracy                           0.59       250
   macro avg       0.58      0.58      0.57       250
weighted avg       0.60      0.59      0.59       250
```

Fig 11: MFCC Featurization - Support Vector Classifier

5. XGBoost Classifier: Obtained an average accuracy of 46%. See Figure 12.

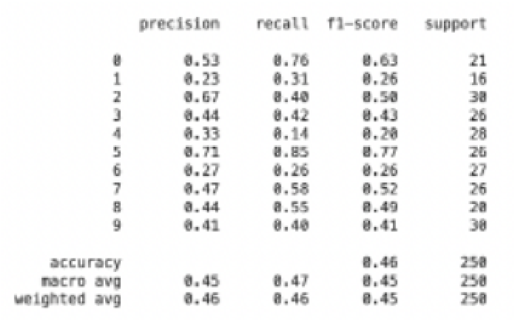
Deep Learning Models

1. CNN1: Obtained an average accuracy of 80%. See Figure 13.

2. CNN2: Obtained an average accuracy of 81%. See Figure 14.

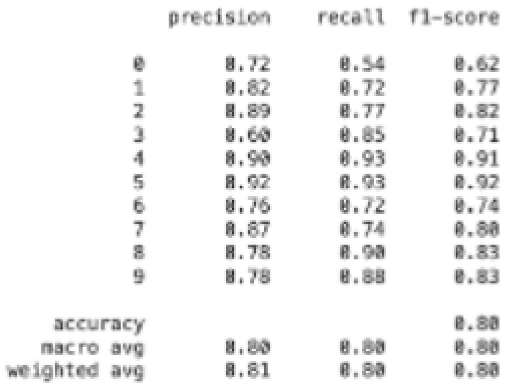
3. CNN3: Obtained an average accuracy of 81%. See Figure 15.

4. GRU: Obtained an average accuracy of 74%. See Figure 16.



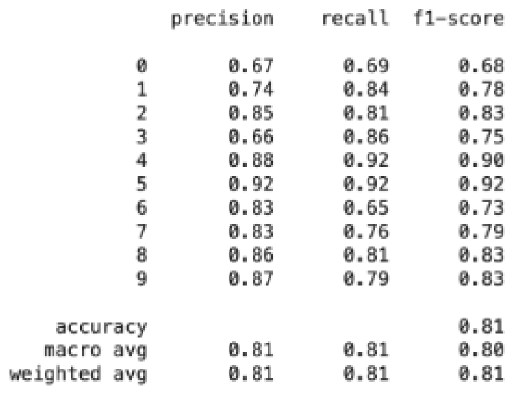
```
              precision    recall  f1-score   support

           0       0.53      0.76      0.63        21
           1       0.23      0.31      0.26        16
           2       0.67      0.40      0.50        30
           3       0.44      0.42      0.43        26
           4       0.33      0.14      0.20        28
           5       0.71      0.85      0.77        26
           6       0.27      0.26      0.26        27
           7       0.47      0.58      0.52        26
           8       0.44      0.55      0.49        20
           9       0.41      0.40      0.41        30

    accuracy                           0.46       250
   macro avg       0.45      0.47      0.45       250
weighted avg       0.46      0.46      0.45       250
```

Fig 12: MFCC Featurization - XGBoost Classifier



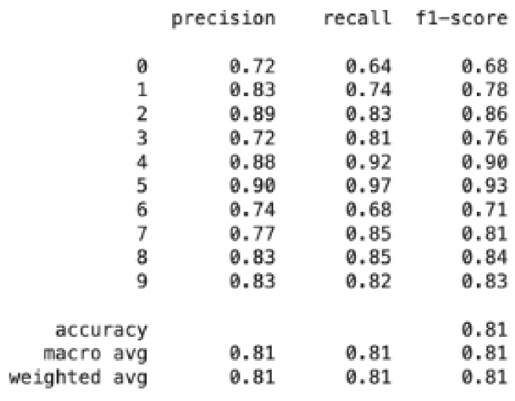
```
              precision    recall  f1-score

           0       0.72      0.54      0.62
           1       0.82      0.72      0.77
           2       0.89      0.77      0.82
           3       0.60      0.85      0.71
           4       0.90      0.93      0.91
           5       0.92      0.93      0.92
           6       0.76      0.72      0.74
           7       0.87      0.74      0.80
           8       0.78      0.90      0.83
           9       0.78      0.88      0.83

    accuracy                           0.80
   macro avg       0.80      0.80      0.80
weighted avg       0.81      0.80      0.80
```

Fig 13: CNN1



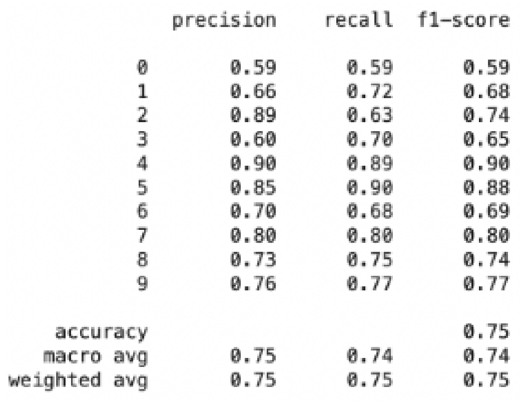
```
              precision    recall  f1-score

           0       0.67      0.69      0.68
           1       0.74      0.84      0.78
           2       0.85      0.81      0.83
           3       0.66      0.86      0.75
           4       0.88      0.92      0.90
           5       0.92      0.92      0.92
           6       0.83      0.65      0.73
           7       0.83      0.76      0.79
           8       0.86      0.81      0.83
           9       0.87      0.79      0.83

    accuracy                           0.81
   macro avg       0.81      0.81      0.80
weighted avg       0.81      0.81      0.81
```

Fig 14: CNN2



```
              precision    recall  f1-score

           0       0.72      0.64      0.68
           1       0.83      0.74      0.78
           2       0.89      0.83      0.86
           3       0.72      0.81      0.76
           4       0.88      0.92      0.90
           5       0.90      0.97      0.93
           6       0.74      0.68      0.71
           7       0.77      0.85      0.81
           8       0.83      0.85      0.84
           9       0.83      0.82      0.83

    accuracy                           0.81
   macro avg       0.81      0.81      0.81
weighted avg       0.81      0.81      0.81
```

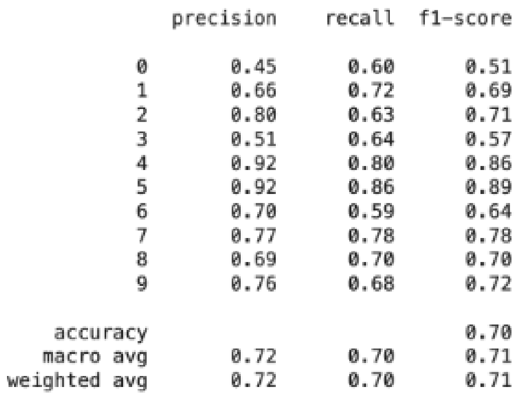
Fig 15: CNN3



```
              precision    recall  f1-score

           0       0.59      0.59      0.59
           1       0.66      0.72      0.68
           2       0.89      0.63      0.74
           3       0.60      0.70      0.65
           4       0.90      0.89      0.90
           5       0.85      0.90      0.88
           6       0.70      0.68      0.69
           7       0.80      0.80      0.80
           8       0.73      0.75      0.74
           9       0.76      0.77      0.77

    accuracy                           0.75
   macro avg       0.75      0.74      0.74
weighted avg       0.75      0.75      0.75
```

Fig 16: GRU

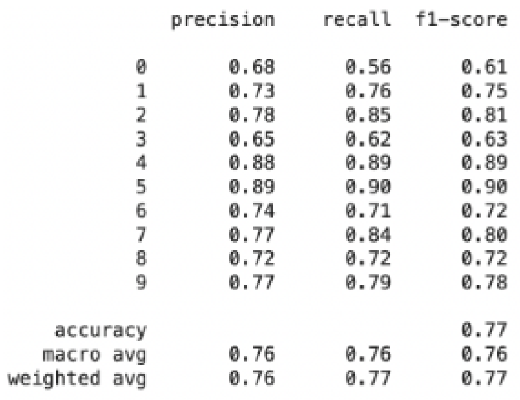
5. BirDirectional GRU: Obtained an average accuracy of 70%. See Figure 17.



```
              precision    recall  f1-score

           0       0.45      0.60      0.51
           1       0.66      0.72      0.69
           2       0.80      0.63      0.71
           3       0.51      0.64      0.57
           4       0.92      0.80      0.86
           5       0.92      0.86      0.89
           6       0.70      0.59      0.64
           7       0.77      0.78      0.78
           8       0.69      0.70      0.70
           9       0.76      0.68      0.72

    accuracy                           0.70
   macro avg       0.72      0.70      0.71
weighted avg       0.72      0.70      0.71
```

Fig 17: BiDirectional GRU

6. LSTM: Obtained an average accuracy of 76%. See Figure 18.



```
              precision    recall  f1-score

           0       0.68      0.56      0.61
           1       0.73      0.76      0.75
           2       0.78      0.85      0.81
           3       0.65      0.62      0.63
           4       0.88      0.89      0.89
           5       0.89      0.90      0.90
           6       0.74      0.71      0.72
           7       0.77      0.84      0.80
           8       0.72      0.72      0.72
           9       0.77      0.79      0.78

    accuracy                           0.77
   macro avg       0.76      0.76      0.76
weighted avg       0.76      0.77      0.77
```

Fig 18: LSTM

Best Results of each type(Model)	
TYPE	**Macro F1**
FFT-XgBoost	0.45
MFCC-RF	0.59
RNN-LSTM	0.75
CNN 3 (overall best)	0.81

Conclusion: As you can see the results from the table, the CNN3 model outperforms all other models.

8 Individual Contributions

1. Amuru Hareesh: ML Models and FFT Based Data Featurization and Visualization.

2. Meenal Jain: MFCC Based Featurization and Training of ML Models.

3. Prakhar Kumar: Converting MFCC Features into time based data and as image data, Training DL Models.

Apart from that, all of the project components have been discussed and everyone has contributed equally to all of them.

References

[1] J. K. Bhatia, R. D. Singh, and S. Kumar, "Music genre classification," in *2021 5th International Conference on Information Systems and Computer Networks (ISCON)*, pp. 1–4, 2021.

[2] Y.-H. Cheng, P.-C. Chang, and C.-N. Kuo, "Convolutional neural networks approach for music genre classification," in *2020 International Symposium on Computer, Consumer and Control (IS3C)*, pp. 399–403, 2020.

[3] S. Allamy and A. L. Koerich, "1d CNN architectures for music genre classification," *CoRR*, vol. abs/2105.07302, 2021.

[4] L. Feng, S. Liu, and J. Yao, "Music genre classification with paralleling recurrent convolutional neural network," *CoRR*, vol. abs/1712.08370, 2017.

[5] A. KM *et al.*, "Deep learning based music genre classification using spectrogram," 2021.

[6] T. Feng, "Deep learning for music genre classification," *private document*, 2014.