Music Genre Classification

Yuvaraj Sriramoju
11710243
YuvarajSriramoju@my.unt.edu

B.N. Narasimha Reddy Machha
11671010
BNNarasimhaReddyMachha@my.unt.edu

Navya Bade
11648430
NavyaBade@my.unt.edu

Sreeja Bommineni
11648419
Sreejabommineni@my.unt.edu

***Abstract*—Music genre classification uses the GTZAN dataset to test sophisticated machine learning methods for music genre classification. The limitations of traditional approaches were their dependence on domain expertise and their use of traditional models for machine learning and manually derived characteristics such as MFCCs. We used Convolutional Neural Networks (CNNs) and feature engineering with Mel-Spectrograms to overcome these restrictions. Critical tonal and rhythmic patterns were caught by converting audio data into time-frequency representations, which increased the classification accuracy. Additionally, we included frequency-based and time-based characteristics to improve the model's capacity to accurately identify genres. For a variety of musical datasets, the suggested methodology shows how deep learning may automate feature extraction and increase accuracy, making it more scalable and effective.**

I. Introduction

Music genre classification aims to categorize songs into different genres such as rock, jazz, classical, and hip-hop using machine learning models. This task helps in enhancing user experience in music streaming platforms by improving recommendation systems and can be helpful in organizing large music libraries. By classifying the music perfectly, music platforms such as Spotify, Amazon Music etc., can offer personalized suggestions, better search functionality, and automated playlist generation, ensuring users discover music aligned with their preferences.

The main challenges in this problem include handling noisy data, class imbalance and genre overlap, where songs exhibit characteristics of multiple genres. Additionally, processing large-scale audio datasets efficiently and extracting relevant features like Spectrograms or chroma from raw audio can be some technical challenges

This project improves the accuracy of classification by using spatial correlations in the domain of time and frequency with the use of a CNN. This makes an improvement when compared to the previous other approaches where those methods lacks the ability to learn features constantly.

II. Literature Survey

Music genre classification has been a big problem and is widely researched in the audio signal processing and machine learning fields. The previous approaches were mostly depend on traditional ml models with audio features like MFCCs, Chroma features and Spectral contrast.

This usage of MFCCS together with the other features for genre classification has been done in the Tzanetakis and cook study in 2002 [7]. This study utilized classifiers like KNN and SVMs. Even though this is effective, the technique frequently require domain expertise to extract features.

When the deep learning features and methods came more into usage the researchers have started working towards the automatic feature extraction using CNNs. In 2014 Schrauwen [2], suggested CNN architectures for spectrogram and raw audio inputs. When compared between the convolutional models and CNNs the CNNs had enhance classification accuracy by identifying local patterns in time and frequency when they are used on spectograms. These methods are further worked on and are improved for transfer learning and pretrained models like VGGish for the analysis of the audio by Kim et al (2018) [4].

The standard approaches rely on statistical descriptors such as MFCCS but Mel Spectrograms are used for this project as an input because they have shown higher performance in deep learning applications. This mel spectrograms will collect both the rhythmic and tonal information which is very important for genre differentiation.

The earlier methods and works are depend on ml models like SVM and KNNs. Where as our project uses a CNN based architecture. The CNNs are able to learn complex characteristics directly from the spectrograms.

III. Objective and Research Questions

A. Objective

The main objective of this project is to classify music into various genres by using neural networks. The goal is to explore various feature extraction techniques and train the model to get better accuracy. This can be useful in music recommendation system.

B. Research Questions

1) Does the model's accuracy is affected if the features are reduced or by using normalization?
2) Will there be a chance to increase the model's accuracy to classify if we use data augmentation method for small datasets?
3) What audio features will be useful in classifying the music genres more accurately?

4) Can the feature selection method be used to improve the readability of the model and help the model reduce overfitting?

IV. Dataset Description

We are using GTZAN dataset from Kaggle. The audio files were collected in 2000-2001 from a variety of sources including personal CDs, radio, microphone recordings. The dataset has 10 music genres, each containing 100 audio files of 30 seconds each. The total dataset is of size 1.41 GB. The dataset has audios of 10 genres and spectrogram images of respective audio files. The dataset also contains a csv file of extracted features mean and average value of different extracted features from the audio files for 30 seconds and 3 seconds. But due to computational constraints, we plan to use only a part of the dataset. The dataset has multiple corrupted files in multiple music genre folders. We deleted the files which has more noise and are corrupted.

A. Visualizations

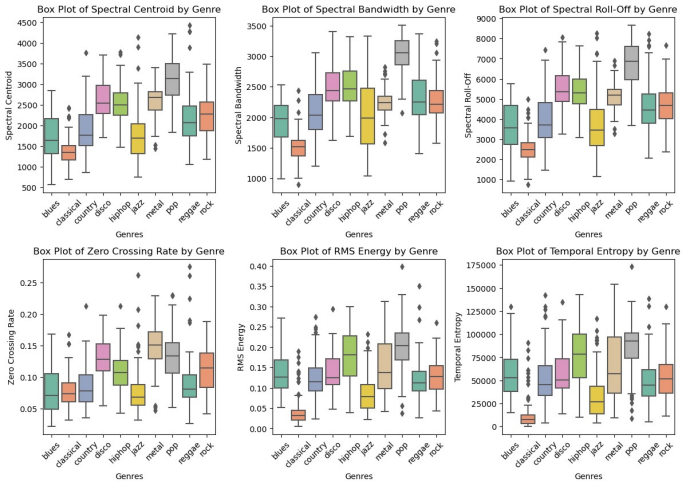

Fig. 1. Box Plot

V. Feature Engineering Methodology

Generally, Data contains many features. Training the model by using all the features can misguide the model as non-relevant features might result in false learning. Feature Engineering is an important step in Machine Learning which can impact classification model performance. In our project, feature engineering mainly refers to transforming, extracting features from raw audio.

A. Extracting Features

The Neural Networks cannot directly take raw audio file as input and train the model. So, we extracted meaningful features which helps in classifying between genres.

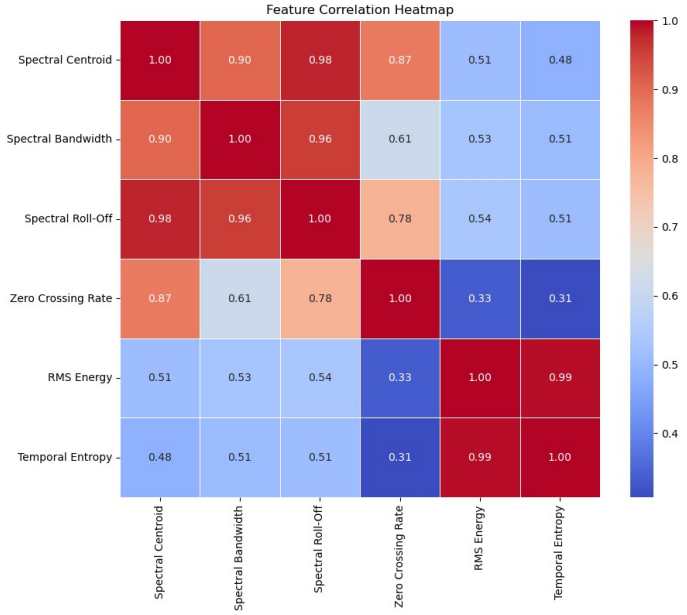

Fig. 2. Heat Map

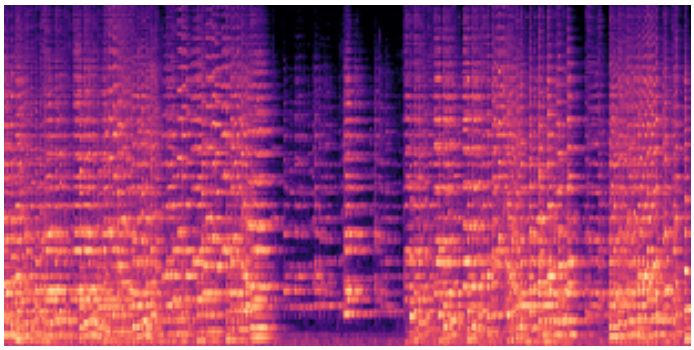

Fig. 3. Mel Spectrogram image of Classical Music

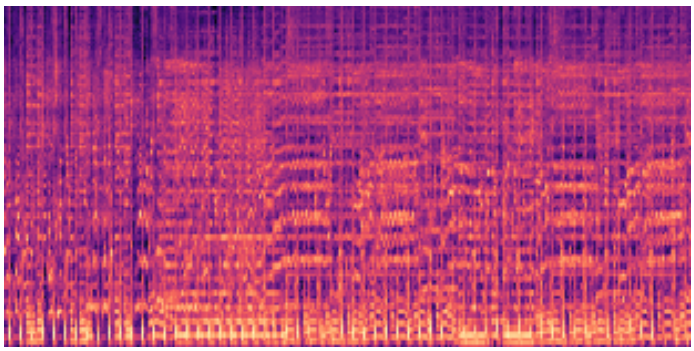

Fig. 4. Mel Spectrogram image of Disco Music

1) Mel-Spectrograms: We converted the raw audio file into waveform. We converted it into spectrogram which used Mel Scale.

$$Spectrogram(t, f) = |X(t, f)|^2 \quad (1)$$

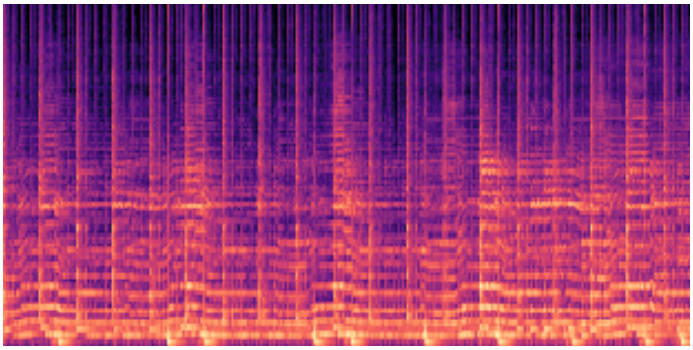

Fig. 5. Mel Spectrogram image of Rock Music

Where $X(t, f)$ is the Fourier transform of the audio signal over time t and frequency f.

2) Frequency-Domain Features: We have extracted the following Frequency Domain features from the raw audio

- Spectral Centroid: Check the audio if it has high for sharp sounds (like cymbals), low for soft sounds (like bass).
- Spectral Bandwidth: It is used for measuring the range of frequencies. It goes wide for complex sounds and narrow for simpler ones.
- Spectral Roll-Off: It focuses on the frequency below of most sound energy. Usually helps to separate low and high-pitched sounds of audio.

3) Time Features: The following time features are extracted from the audio

- Zero Crossing Rate (ZCR): It counts how often the signal changes direction. It goes higher for noisy music and lower for smoother sounds.
- Root Mean Square Energy (RMSE): To measure the loudness of the sound in an audio.
- Temporal Entropy: It is useful for genres like jazz which have dynamic changes in their audio.

B. Transforming Features

After Successfully extracting the features from audio, We transformed the data as we need. We performed various transformations to improve the performance of the model.

1) Logarithmic Mel-Spectrogram: Mel Spectrograms have a wide dynamic range where the lower intensity frequencies can dominate the representation. By a logarithmic transformation to Mel Spectrogram, we can compress the range and extract the important spectral information.

$$\text{Log-Mel}(t, f) = \log(1 + \text{Mel-Spectrogram}(t, f)) \quad (2)$$

The use of $\log(1 + x)$ ensures that all values are positive and helps in managing variations in the signal's amplitude.

2) Delta and Delta-Delta Features: The delta and delta-delta features are generally used to capture the first and second-order derivatives of the spectrogram over time. By this, we can know how the frequency changes over time and can help models identify patterns such as rhythmic changes or tonal shifts in the music. The Delta captures the first order rate of change

$$\delta(t, f) = \frac{\text{Mel-Spectrogram}(t + 1, f) - \text{Mel-Spectrogram}(t - 1, f)}{2} \quad (3)$$

The Delta-Delta captures the second order rate of change.

$$\delta\delta(t, f) = \frac{\delta(t + 1, f) - \delta(t - 1, f)}{2}] \quad (4)$$

3) Smoothing: Smoothing is used to reduce high-frequency noise and fine-grain fluctuations in the data. We applied Gaussian smoothing to the spectrogram image to create a smoother representation of the audio content.It averages the pixel values in a local neighborhood with a Gaussian weight and blurs high frequency components. This help in reducing the noise and unnecessary variations.

$$\text{Smoothed Image}(t, f) = \mathcal{G}(t, f, \sigma) \quad (5)$$

4) Edge Detection: Edge detection is an important transformation in computer vision. It is used to identify the regions where there are changes in intensities in images. We applied Canny edge detection algorithm spectrogram images to identify the boundaries of different regions. We converted Mel-spectrogram into an 8-bit image and then used the Canny algorithm to detect edges.

$$\text{Edges}(t, f) = \text{Canny}(\text{Image}(t, f) \times 255) \quad (6)$$

5) Combining Features: Once we extracted all these features, we have combined them into single representation for input into a model. The combination of these features can help the model to learn from rich representation and for a better performance.

$$\text{Combined Features} = \text{Stack}(\text{Log-Mel}, \delta, \delta\delta, \text{Smoothed}, \text{Edges}) \quad (7)$$

C. Feature Selection

When dealing with high dimensional data such as audios, it is important to select features. Feature selection main task is to identify and use the most relevant features that contribute to the classification task and remove unnecessary features which have less to no use. By using optimal set of features, we can improve model performance, reduce over fitting and reducing training time.

1) Correlation-based Feature Selection: Features with high correlation to the target class are retained.

2) Random Forests: By using the importance scores, we ranked the features.

3) Dimensionality of Spectrograms: Mel-spectrograms can have a high number of dimensions. By using Feature selection methods, we reduced the number of dimensions and only used relevant features.

VI. Experimental Setup

A. Dataset

We have used GTZAN audio dataset from Kaggle. The dataset has 10 music genres, each containing 100 audio files of 30 seconds each. The total dataset is of size 1.41 GB.

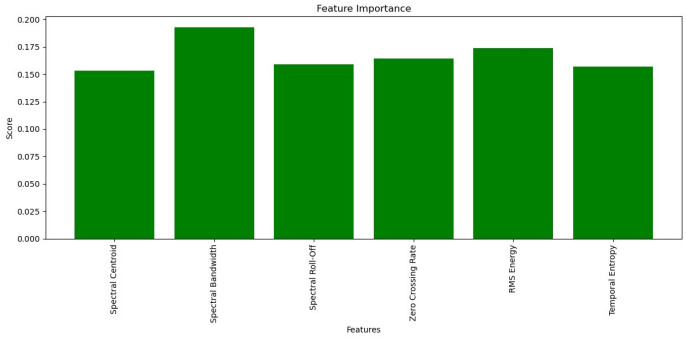

Fig. 6. Feature Importance

1) PreProcessing: After collecting the data, we performed pre-processing steps. We made sure that all the audio files in the dataset are usable by checking if there are any missing or corrupted files. In case of corruption we removed the files. We also handled class imbalance by applying oversampling and under sampling for the classes.

B. Model Architecture

For our project, we developed three models classify music genres and compared based on their performance. Each model has different approaches to feature extraction and machine learning techniques.

1) Random Forest ML Model: We first developed Random Forest Classifier. We chose it as baseline model to evaluate traditional ML algorithms when trained on extracted audio features like Frequency and time based. We split the data into 70-30 and trained the model with 100 estimators

2) CNN Model with Raw Spectrogram Images: Next we trained created a CNN Model for Raw spectrogram images. The architecture consists of the following layers:

- Convolutional layer with 32 filters, kernel size 3x3, and ReLU activation
- Max pooling layer with pool size 2x2
- Convolutional layer with 64 filters, kernel size 3x3, and ReLU activation
- Max pooling layer with pool size 2x2
- Fully connected (dense) layer with 128 units and ReLU activation
- Output layer with 10 units (for 10 genres) and softmax activation

3) CNN Model with Feature Engineering Spectrograms: Later, We created another CNN model with same architecture to train on Feature Engineered images. We made sure that, same model is used for both the types. It has the following layers

- Convolutional layer with 32 filters, kernel size 3x3, and ReLU activation
- Max pooling layer with pool size 2x2
- Convolutional layer with 64 filters, kernel size 3x3, and ReLU activation
- Max pooling layer with pool size 2x2
- Fully connected (dense) layer with 128 units and ReLU activation
- Output layer with 10 units (for 10 genres) and softmax activation

C. Training Details

We trained both the models using Adam optimizer with an initial learning rate of 0.001. The batch size was set to 32, and the model was trained for 50 epochs.

D. Computational Resources

1) Software: We have worked in Jupyter notebook on Python 3.9. We have used different python libraries such as Keras, Tensorflow, librosa, Matplotlib, Pandas, Numpy, Sklearn, OpenCV etc

2) Hardware: We trained our model on intel i7 CPU which has 8GB RAM. We also made use of NVIDIA GTX 1650 GPU with 4GB RAM.

VII. Model Evaluation

Model Evaluation helps to understand how well a model has learned. We used accuracy as a standard metric since, it is commonly used metric for classification. For the Random Forest Classifier we have calculated Precision, Recall and F1-Score.

1) Accuracy: It measures the percentage of correct predictions out of the total predictions made by the model.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{8}$$

2) Precision: It measures the proportion of positive predictions that are actually correct.

$$\text{Precision} = \frac{\text{TP}}{\text{TP + FP}} \tag{9}$$

3) Recall: It measures the proportion of actual positives that are correctly identified by the model.

$$\text{Recall} = \frac{\text{TP}}{\text{TP + FN}} \tag{10}$$

4) F1-score: The harmonic mean of precision and recall, providing a single metric that balances the two.

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision + Recall}} \tag{11}$$

Where:

- TP = True Positives
- TN = True Negatives
- FP = False Positives
- FN = False Negatives

We have also generated confusion matrix for RF CLassifier which provides model's performance by showing the true positive, false positive, true negative, and false negative counts for each class. The Fig 1 shows the consfusion matrix.

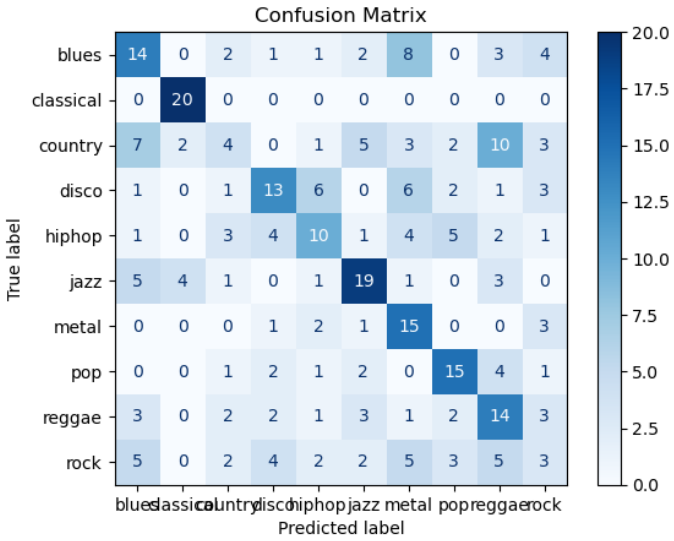

Fig. 7. Confusion Matrix

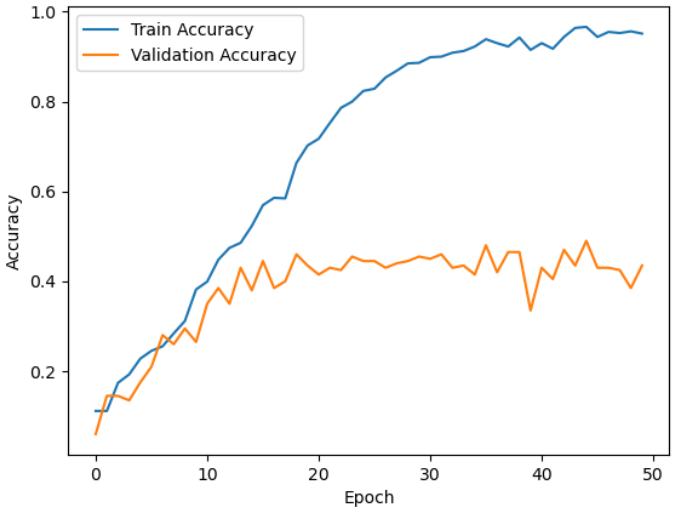

Fig. 9. Accuracy vs Epoch

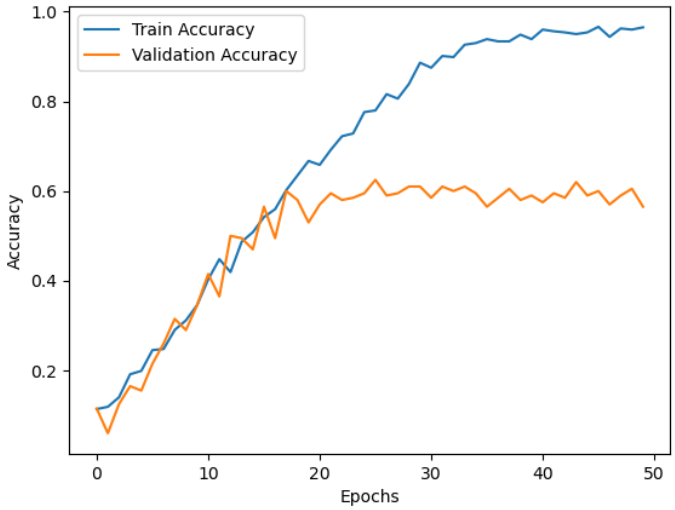

Fig. 8. Accuracy vs Epoch

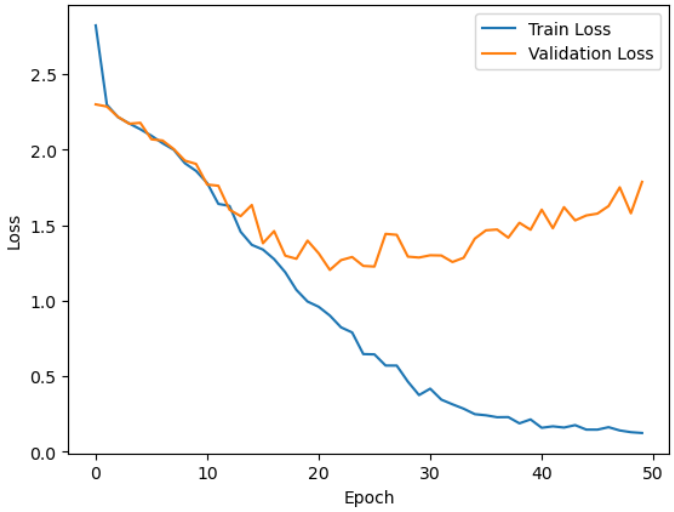

Fig. 10. Train Loss and Validation Loss

A. *Train and Validation Accuracy vs Epochs*

We have plotted train accuracy and validation accuracy over epochs. This helps to know how well the model is performing on both the training and validation datasets. The Fig 2 displays the Training and validation accuracies over epochs for Normal CNN model with normal spectrogram images. The Fig 3 displays the Training and validation accuracies over epochs for CNN model with Featured spectrogram images.

B. *Train Loss vs Validation loss*

We have also generated Train Loss vs. Validation Loss graph. This helps to assess how well the model is minimizing the loss function during training. This helps in detecting the overfitting and underfitting.

The Fig 3 shows the comparison between Training loss and Validation for Normal CNN model with normal spectrograms

The Fig 4 shows the comparison between Training loss and Validation for CNN model with Featured spectrogram images.

VIII. Deployment

We have deployed our project using GradioAPI. To make our music genre classification model accessible to users and ensure ease of interaction, we deployed the model using GradioAPI. It is a powerful ML library for building Interfaces.

Gradio reduces the need for heavy frontend development by allowing us to rapidly design simple web applications that enable smooth interaction with machine learning models.We utilized Gradio's straightforward interface to create a web application that allows users to upload audio files and predict music genre. The Interface includes:

- Input: Users can upload an audio file (e.g., MP3 or WAV format) via an upload button.

- Output: The predicted genre is displayed after the model processes the input file.

Gradio is simple to use and simplifies the process of building interactive interfaces for machine learning models. Fig 5 shows

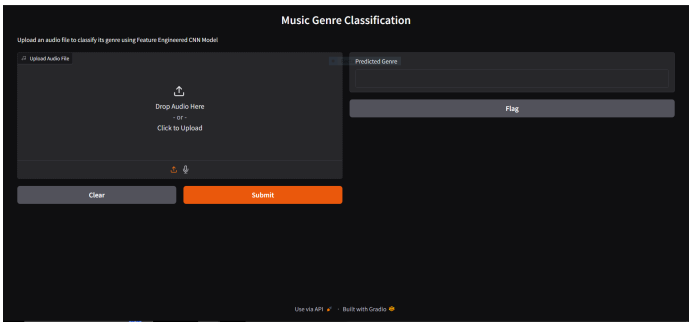

Fig. 11. Gradio Interface

the Gradio Interface. It has a block to upload audio file. After submitting the audio, the background will predict the genre and display it.

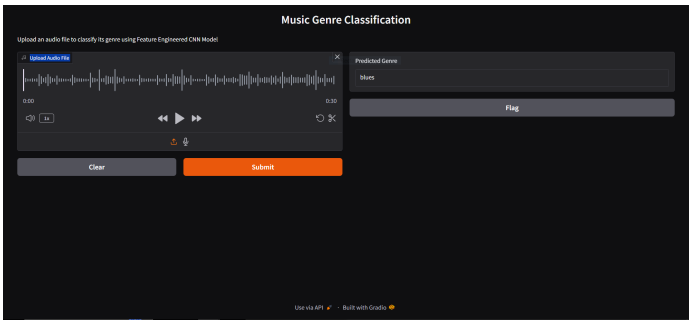

Fig. 12. Gradio Interface with Prediction

IX. Results and Discussions

A. Results

Model/Accuracy	Acc	Validation Acc	Loss	Validation Loss
Random Forest	42%	-	-	-
CNN Model	95%	43%	0.14	2.4
CNN (Featured)	96.5%	56%	0.12	1.7

TABLE I
Model Performance Comparison

The Above Table 1 gives the result comparison over all the models. We can see that Normal Machine Learning Random Forest Model has less accuracy 42%. CNN Model which is trained using Feature Engineered images has edge over CNN model which is trained using normal spectrogram images. The feature engineering had a positive impact on the CNN model. By selecting or creating more relevant features the model was able to learn more effectively.

Feature Engineering steps provided the model with high quality features. The might have helped the model to learn more. The CNN (Featured) model showed a reduction in validation loss from 2.4 to 1.7 compared to the CNN Model. This decrease in loss indicates improved generalization and the model can handle unseen data more likely.

B. Challenges and Limitations

We have faced a lot of challenges and limitations in developing our model

1) Overfitting: Overfitting is one of the common challenges in deep learning. Balancing the training accuracy and validation accuracy requires careful regularization dropout etc. Both our CNN models have high training accuracy and less validation accuracy.

2) Feature Engineering and Selection: One of the major challenges we had is feature selection and transformation. We have chose log-mel transformations, delta-delta and edging by trail and error methods over some of the techniques.

3) Hyperparameter tuning: It is a common challenge for CNNs. We have tried fine tuning learning rates, batch sizes, and network architecture to reduce overfitting. We also trained our model using numerous number of epochs to achieve a good accuracy.

4) Computational Constraint: Our main limitation is lack of enough resources. Due to less hardware resources, we constrained ourselves to less layers and basic CNN models. If wish to try high models such as RNNs and transformers.

X. Conclusion

To Conclude [6], we created a model using Artificial Intelligence to detect the type of the music. We aim to improve the model performance by mainly focusing on feature engineering of the data. The results clearly shows the importance of feature selection and transformation. We converted the raw audio files into Mel-spectrograms and trained them using CNN's. We also extracted numerical features such as Frequency and time domain features for ML model.

The CNN (Featured) model showed a good performance over the basic CNN Model and the Random Forest model. The Random Forest Classifier showed a poor performance with only 42% accuracy. On the other side the CNN model achieved good results but resulted in overfitting. By introducting feature engineering techniques, the validation accuracy increased to 56%, and validation loss decreased. Our major challenges during project are overfitting, data quality and computational requirements.

While the feature engineering methods we used in our project contributed significantly, there can be many improvements done for future research and improvements. We can use advance engineering techniques such as spectral contrast, spectral roll-off and tonnetz, which can give important features. Also data augmentation techniques like pitch shifting, time-stretching, or noise addition can help the model to become more robust and generalize better when dealing with smaller and imbalances datasets.

On the other hand, we can also explore more advances models such as Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM) etc., which can capture the temporal dependencies in music. Also, by using transfer learning, it can be easy to train the model and achieve good results.

REFERENCES

[1] Nitin Choudhury, Deepjyoti Deka, Satyajit Sarmah, and Parismita Sarma. Music genre classification using convolutional neural network. In *2023 4th International Conference on Computing and Communication Systems (I3CS)*, pages 1–5, 2023.

[2] Sander Dieleman and Benjamin Schrauwen. End-to-end learning for music audio. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6964–6968, 2014.

[3] Anirudh Ghildiyal, Komal Singh, and Sachin Sharma. Music genre classification using machine learning. In *2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, pages 1368–1372, 2020.

[4] Jaehun Kim, Minz Won, Xavier Serra, and Cynthia Liem. Transfer learning of artist group factors to musical genre classification. pages 1929–1934, 04 2018.

[5] K S Mounika, S Deyaradevi, K Swetha, and V Vanitha. Music genre classification using deep learning. In *2021 International Conference on Advancements in Electrical, Electronics, Communication, Computing and Automation (ICAECA)*, pages 1–7, 2021.

[6] OpenAI. Chatgpt (version 4), 2024. Accessed: December 4, 2024.

[7] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, 2002.