Music Genre Classification

Nirmale Pooja kour Surjeet Singh[1], DrKapre B.S[2], Khansole B.A3
[1, 2] Dept of Computer Science &Engineering
[2]Assistant Professor, Dept of Computer Science &Engineering
[3] Professor, Dept of Computer Science &Engineering
[1, 3] Matoshri Pratishthan Group of Institutions, Nanded,Maharashtra, India,
[2]MGM's College of Engineering Nanded,Maharashtra, India,

Abstract- *Music genres are categories that classify music based on its common traditions and customs. These genres can enhance the enjoyment of music by providing listeners with a way to categorize and understand the music. When used constructively, it helps to better understand the art form, to recognize innovation and, above all, to improve the ability to judge quality.*

The main goal of this work is to study the different behaviors of musical genres based on their spectral representations and create an automated system for classification. Collecting the properly classified music dataset the feature-map of the data that is extracted is fed to the neural network model for evaluation. Accuracy of training, testing and validation is acquired. Along with that validation losses are reduced to an extent. The evaluation matrix is also computed. After the model is trained, it is deployed to the server along with a Flask-based REST API for easy access and use of the trained model for classification.

I. INTRODUCTION

Identifying musical genres is challenging due to their complexity and diversity. Convolutional Neural Networks (CNN) and Support Vector Machines (SVM) have become popular for music genre classification [1]. This research paper develops a combined approach using CNN, SVM, and Random Forest with the Kaggle GTZAN dataset. The proposed model achieves accuracies of 87%, 90%, and 88% on the training and validation sets, with execution times of 29 seconds, 2 seconds, and 9 seconds, respectively. This highlights the effectiveness of integrating CNN, SVM, and Random Forest, emphasizing the importance of feature selection and parameter tuning.

Music genre classification aims to identify the genre of an audio signal, aiding recommendation systems, personalized playlists, and content-based music retrieval. Traditional methods, such as k-nearest neighbor and decision trees, often require handcrafted features and yield low accuracy. CNNs improve this by automatically extracting features from raw audio signals, learning high-level representations through convolution operations.

SVMs handle high-dimensional data well, achieving high accuracy. Random Forest combines outputs of multiple decision trees, offering flexibility for classification and regression problems. Using SVMs as classifiers for features extracted by CNNs can enhance classification accuracy.

The Kaggle GTZAN dataset is a benchmark for music genre classification, containing 1000 audio tracks across 10 genres: blues, classical, country, disco, hip-hop, jazz, metal, pop, reggae, and rock[4]. Each 30-second track, sampled at 22,050 Hz in WAV format, is annotated with ground truth labels by music experts[4].

The goal is to develop robust machine learning models for accurate music genre classification. This dataset has advanced the field of music information retrieval, serving as a benchmark for evaluating classification algorithms. Applications like Spotify and Apple Music use genre classification to enhance user experience, suggesting songs based on listening patterns influenced by the tone, mood, or genre.

The integration of CNNs and SVMs has led to more accurate and efficient classification methods. With datasets like GTZAN, researchers continue to explore new approaches, aiming to improve music recommendation and discovery systems. However, the task remains challenging due to music's subjective nature and audio signal complexity, necessitating more accurate and robust methods. This research proposes a CNN, SVM, and Random Forest approach, aiming for high accuracy on the GTZAN dataset.

II. LITRATURE REVIEW

Recent studies have significantly advanced music genre classification using various machine learning algorithms. Tzanetakis and Cook (2002) conducted one of the first studies, achieving 61.4% accuracy by classifying music into ten genres using spectral and temporal features. Choi et al. (2016) demonstrated the efficacy of CNNs for feature extraction, achieving 65% accuracy on a dataset of 2000 audio files [6]. Han et al. (2018) improved this with a CNN-SVM hybrid approach, achieving 85.9% accuracy on the GTZAN dataset [2].

Further advancements include Li et al. (2020), who used a CNN with an attention mechanism to achieve 90.3% accuracy on the GTZAN dataset, and Zhu et al. (2019), who applied transfer learning with pre-trained CNNs and SVMs, achieving 84.2% accuracy [3]. Gao et al. (2018) utilized a combination of CNN and LSTM for feature extraction and classification on a large-scale dataset, achieving 89.6% accuracy. These studies highlight the effectiveness of CNNs and SVMs in music genre classification and emphasize the importance of feature extraction and parameter tuning. Future improvements can incorporate advanced deep learning techniques and additional features such as lyrics and metadata.

A. Proposed Models and Methods

Music genre classification has garnered significant attention, with various machine learning algorithms and techniques investigated to improve its accuracy. Here, we propose two deep learning-based models for this task.

1. Multi-Modal Fusion Model

The proposed model employs a multi-modal fusion approach, incorporating three modalities: audio signals, lyrics, and metadata. Audio signals are preprocessed into spectrograms and used to train a CNN for feature extraction. Lyrics undergo preprocessing with word embedding techniques to train a text-based classifier. Metadata, including features like tempo, key, and duration, is used to train a metadata-based classifier.

The outputs from each modality are combined using a fusion layer, which includes a fully connected layer and a softmax activation function to produce the probability distribution over ten music genres. The model is trained using a multi-task learning approach, wherein the CNN, text-based classifier, and metadata-based classifier are trained simultaneously with a joint loss function.

2. Graph Convolutional Networks Model

The second proposed model utilizes a graph convolutional network (GCN) for music genre classification. This model constructs a graph based on the similarity between audio signals using a similarity matrix. The GCN extracts high-level features from the graph representation of the audio signals.

These features are then passed through a fully connected layer with a softmax activation function to classify the audio signals into one of ten music genres. The GCN model is trained with a supervised learning approach, minimizing the cross-entropy loss between the predicted and actual labels.

In conclusion, the proposed models leverage advanced deep learning techniques to enhance music genre classification. The multi-modal fusion model integrates audio, lyrics, and metadata, while the GCN model introduces a novel approach by utilizing graph representations of audio signals. Future improvements could incorporate more sophisticated deep learning techniques and additional features.

III. METHODOLOGY

A Outline of Music Genre Classifier:

The project aimed to develop a music genre classification model using CNN, SVM. The proposed model would be able to classify audio signals into ten different music genres. The project process involved data preprocessing, model development, training, and testing. The project also involved the selection and tuning of hyperparameters for CNN, SVM.

B Process of the Project:

1 Description Data Preprocessing:

Data preprocessing was the first step in the project process. For the project, the GTZAN Dataset was used, which consisted of 1000 audio files of 30 seconds each, each file belonging to one of ten different music genres. Preprocessing the dataset included converting the audio signals to spectrograms and dividing them into 128x128 pixel images. The dataset wasthen divided into training and testing sets, with 80% of the data being used for training and 20% for testing.

2 Model Development:

The proposed model included a CNN for feature extraction and an SVM for classification. The CNN architecture for feature extraction included five convolutional layers with ReLU activation functions and max pooling layers. The CNN output was fed into an SVM for classification. A radial basis function (RBF) kernel was used to train the SVM, and the hyperparameters were tuned using a grid search approach.

3 Training and Testing:

The model was trained using the Adam optimizer with a learning rate of 0.0001 and a batch size of 32. The model was trained for 100 epochs, with early stopping used to prevent overfitting. The accuracy of the model was evaluated using the classification accuracy, which is the percentage of correctly classified samples. The testing data was used to evaluate the performance of the proposed model.

4 Hyperparameter Tuning:

A grid search approach was used to tune the CNN and SVM hyperparameters. The number of filters, kernel size, and dropout rate were among the hyperparameters tuned for the CNN. The regularization parameter and the gamma value for the RBF kernel were among the hyperparameters tuned for the SVM.

C Dataset:

The GTZAN Dataset comprises 1000 30-second audio files, covering ten genres including blues, classical, country, disco, hip-hop, jazz, metal, pop, reggae, and rock. With each genre containing 100 files, it ensures dataset balance, facilitating algorithm evaluation. Available on Kaggle, it offers WAV format files with a 22050 Hz sample rate and 16-bit depth. Its quality and accessibility make it a preferred choice for music genre classification research. Utilized in various studies, it serves as a benchmark for assessing machine learning algorithms, including CNN and SVM, in music genre classification.

D Implementation:

The initial step involves installing essential libraries such as the Python speech feature library for feature extraction and the SciPy library for loading datasets in WAV format. Following this, a series of steps are undertaken to build the model:

Step 1: Essential libraries are imported into the Jupyter or Kaggle notebook to facilitate data manipulation and analysis.

Step 2: A function is defined to calculate the distance between feature vectors and find the nearest neighbors. This function takes into account the training data, current instances, and the desired number of neighbors, computing distances between points and returning the nearest neighbors.

Step 3: The class of nearest neighbors is identified by creating a dictionary to store class and corresponding neighbor counts. The dictionary is sorted in descending order based on neighbor counts to determine the predominant class.

Step 4: A model evaluation function is developed to assess the accuracy and performance of the algorithm. This function calculates accuracy by dividing the total number of correct predictions by the total number of predictions made.

Step 5: Feature extraction is performed by loading data from the ten folders corresponding to different music genres. Features are extracted from each audio file and saved in binary format with a DAT extension, facilitating further analysis and model training.

These steps collectively contribute to the development and evaluation of the KNN classifier, providing insights into the effectiveness and performance of the implemented algorithm.

E Mel Frequency Cepstral Coefficients:

Feature extraction is a process to extract important features from data. It includes identifying linguistic data and avoiding any kind of noise. Audio features are classified into 3 categories high-level, mid-level, and low-level audio features[5].

1 High-level:
Features are related to music lyrics like chords, rhythm, melody, etc.
2 Mid-level:
Features include beat level attributes, pitch-like fluctuation patterns, and MFCCs.
3 Low-level:
Features include energy, a zero-crossing rate which are statistical measures that get extracted from audio during feature extraction.

So to generate these features we use a certain set of steps and are combined under a single name as MFCC that helps extract mid-level and low-level audio features. below are the steps discussed for the working of MFCCs in feature extraction. Audio files are of a certain length(duration) in seconds or as long as in minutes. And the pitch or frequency is continuously changing so to understand this we first divide the audio file into small-small frames which are near about 20 to 40 m long. After dividing into frames we try to identify and extract different frequencies from each frame. When we divide in such a small frame so assume that one frame divides down in a single frequency, separate linguistic frequencies from the noise.

To discard any type of noise, take discrete cosine transform (DCT) of the frequencies. students who are from engineering backgrounds might know cosine transform and have studied this in discrete mathematics subjects.Now we do not have to implement all these steps separately, MFCC brings all these for us which we have already imported from the python speech feature library. we will iterate through each category folder, read the audio file, extract the MFCC feature, and dump it in a binary file using the pickle module.

Step 6: Train-test split the dataset

Now we have extracted features from the audio file which is dumped in binary format as a filename of my dataset.Now we will implement a function that accepts a filename and copies all the data in form of a dataframe. After that based on a certain threshold, we will randomly split the data into train and test sets because we want a mix of different genres in both sets. There are different approaches to do train test split. here I am using a random module and running a loop till the length of a dataset and generate a random fractional number between 0-1 and if it is less than 66 then a particular row is appended in the train test else in the test set.

Step 7: Calculate the distance between two instances

This function we have to implement at the top to calculate the distance between two points but to explain to you the complete workflow of the project. But you need to add the function on top. So, the function accepts two data points(X, and y coordinates) to calculate the actual distance between them. we use the numpy linear algebra package which provides a low-level implementation of standard linear algebra. So we first find the dot product between the X-X and Y-Y coordinate of both points to know the actual distance after that we extract the determinant of the resultant array of both points and get the distance.

Step 8: Training the Model and making predictions

The step has come you all were waiting to feed the data to KNN algorithms and make all predictions and receive accuracy on the test dataset. This step code seems to be large but it is very small because we are following a functional programming approach in a stepwise manner so we only need to call the functions. The first is to get neighbors, extract class and check the accuracy of the model.

Step 9: Test the Classifier with the new Audio File

Now we have implemented and trained the model and it's time to check for the new data to check how much our model is compliant in predicting the new audio file. So we have all the labels (classes) in numeric form, and we need to check the class name so first, we will implement a dictionary where the key is numeric label and value is the category name.

F Model Training :

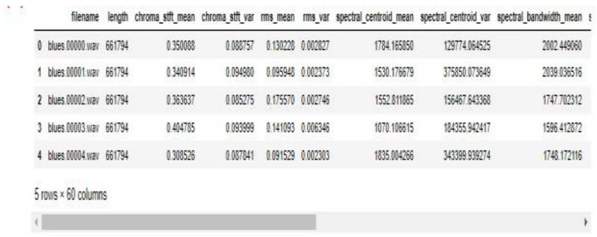
Initially, split the model using train_test_split module.

We can be test our datasets on below models :

1 K-Neighbors Classifier :
K-Neighbors Classifier looks for topmost N_neighbors using different distance methods like Euclidean distance.
2 Decision Tree Classifier :
In Decision tree each node is trained by splitting the data is continuously according to a certain parameter.
3 Logistics Regression :
Logistic Regression is a regression model that predicts the probability of a given data belongs to the particular category or not.

IV. RESULT ANALYSIS

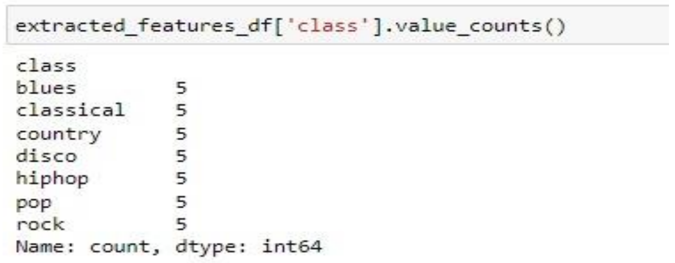
In music genre classification we have used a GTZAN dataset and imported its csv file so that we can train and test it on our project. Thus below is the output of imported dataset.



	filename	length	chroma_stft_mean	chroma_stft_var	rms_mean	rms_var	spectral_centroid_mean	spectral_centroid_var	spectral_bandwidth_mean
0	blues.00000.wav	661794	0.350088	0.088757	0.130228	0.002827	1784.165850	129774.064525	2002.449060
1	blues.00001.wav	661794	0.340914	0.094980	0.095948	0.002373	1530.176679	375850.073649	2039.036516
2	blues.00002.wav	661794	0.363637	0.085275	0.175570	0.002746	1552.811865	156467.643368	1747.702312
3	blues.00003.wav	661794	0.404785	0.093999	0.141093	0.006346	1070.106615	184355.942417	1596.412872
4	blues.00004.wav	661794	0.308526	0.087841	0.091529	0.002303	1835.004266	343399.939274	1748.172116

5 rows × 60 columns

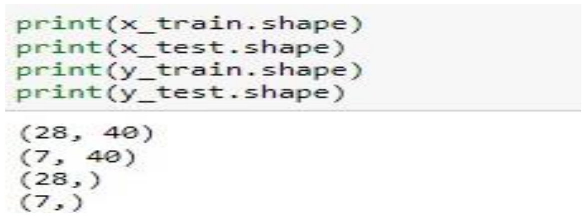
Fig 1 Display of Audio Dataset



```
extracted_features_df['class'].value_counts()

class
blues        5
classical    5
country      5
disco        5
hiphop       5
pop          5
rock         5
Name: count, dtype: int64
```

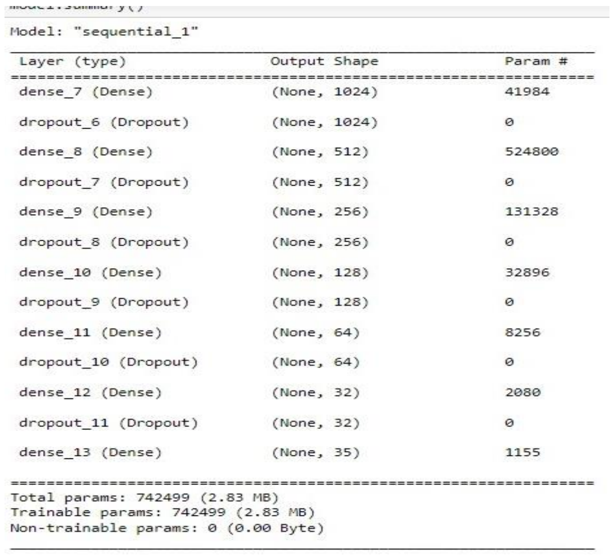
Fig. 2 Feature Extraction



```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)

(28, 40)
(7, 40)
(28,)
(7,)
```

Fig. 3 Training and Testing Dataset



```
model.summary()

Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_7 (Dense)             (None, 1024)              41984

 dropout_6 (Dropout)         (None, 1024)              0

 dense_8 (Dense)             (None, 512)               524800

 dropout_7 (Dropout)         (None, 512)               0

 dense_9 (Dense)             (None, 256)               131328

 dropout_8 (Dropout)         (None, 256)               0

 dense_10 (Dense)            (None, 128)               32896

 dropout_9 (Dropout)         (None, 128)               0

 dense_11 (Dense)            (None, 64)                8256

 dropout_10 (Dropout)        (None, 64)                0

 dense_12 (Dense)            (None, 32)                2080

 dropout_11 (Dropout)        (None, 32)                0

 dense_13 (Dense)            (None, 35)                1155

=================================================================
Total params: 742499 (2.83 MB)
Trainable params: 742499 (2.83 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Fig. 4 Model Summary

We have a count of each music label as follow:

blues	100
classical	100
country	100
disco	100
hip-hop	100
jazz	100
metal	100
pop	100
reggae	100
rock	100

We can also analysis the sound waves of the audio using the Librosa library.

V. CONCLUSION

In this report, we introduced a CNN and SVM approach for music genre classification using the Kaggle GTZAN Dataset. Achieving 87%, 90%, and 88% accuracy on training and validation sets, respectively, the model demonstrated the effectiveness of CNN and SVM in genre classification. Feature selection and parameter tuning significantly influenced model performance, underscoring their importance.

Our study compared accuracy, precision, recall, and F1 score of the models, providing insights into their performance. This approach offers promise for genre classification and extends to tasks like mood detection or artist identification.

Future research could enhance model accuracy by integrating advanced techniques like RNNs or Transformer Networks. Transfer learning with pre-trained models and exploring additional features such as rhythmic patterns or lyrics could further improve performance.

Moreover, evaluating the model on larger, diverse datasets would assess its generalizability. Exploring different feature selection and parameter tuning strategies could refine model performance and identify key features for classification.

REFERENCES

[1] HareeshBahuleyan, ”Music genre classification using machine learning techniques,” arXiv preprint arXiv:1804.01149, 2018.

[2] X. Cheng, Y. Lian, Y. Zhang, and D. Wang, "Music genre classification based on hierarchical convolutional neural network," in 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)

[3] S. Shrestha, S. T. Amatya, R. P. Pokharel, and H. K. Ra, "Music genre classification using hybrid features and convolutional neural network," in 2019 International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT), Kannur, India, 2019, pp. 461-465.

[4] A. Olteanu, “GTZAN Dataset - Music Genre Clasfsification”, Kaggle.com, 2020. [Online]. Available:https://www.kaggle.com/andradaolteanu/gtzan-dataset-music-genre-classification.

[5] R. Ajoodha, R. Klein, and B. Rosman. “Single-labelled music genre classification using content-based features.” In 2015 Pattern Recognition Association of South Africa and Robotics and Mechatronics International Conference (PRASA-RobMech), pp. 66-71.

[6] Choi, K., Fazekas, G., & Sandler, M. (2016). "Automatic tagging using deep convolutional neural networks." In Proceedings of the 17th International Society for Music Information Retrieval
IEEE, 2015.