

MUSIC GENRE CLASSIFICATION USING DEEP LEARNING

by

Team D

Koushik Mannaru;

Sravani Pilla;

Sreeja Komma Reddy;

FINAL PROJECT REPORT

for

DATA 606 Capstone in Data Science

University of Maryland Baltimore County

2024

Copyright ©2024
ALL RIGHTS RESERVED

ABSTRACT

The project aims to address the problem of automatic music genre classification, which is essential in the context of music information retrieval. By using Convolutional Neural Networks and Long Short-Term Memory networks, the GTZAN Dataset allowed us to develop a classification model that performed well. Given the growing need for music categorization in current applications such as music streaming and digital libraries, this work is motivated by its importance. We used Librosa, an audio analysis Python library, to extract features from the audio files that we deemed essential. Mel-frequency cepstral Coefficients, spectral centroid, chroma, and spectral contrast were used to extract timbral and harmonic content. We normalized these features and treated them as input to the entire model. Our model consisted of a CNN model and LSTM model that enabled us to extract spatial hierarchies of the audio features and the temporal relationship with the features, respectively. The role of the CNN layers was to learn the local structure and relationship of the audio, while the LSTM layers learned more soft short-term and long-term structure.

LIST OF ABBREVIATIONS AND SYMBOLS

c CNN: Convolutional Neural Network

l LSTM: Long Short-Term Memory

m MFCC: Mel-Frequency Cepstral Coefficients

sr Sample Rate: The number of samples of audio carried per second, measured in Hz.

cf Chroma Features: Features representing the energy distribution across 12 pitch classes, capturing harmonic and melodic characteristics.

sc Spectral Centroid: A measure indicating where the center of mass of the spectrum is located, related to the perceived brightness of a sound.

spc Spectral Contrast: The difference in amplitude between peaks and valleys in a sound spectrum, useful for identifying different textures in music.

gtzan GTZAN Dataset: A widely used dataset for music genre classification tasks, containing 1000 audio tracks across 10 genres.

db Decibel : A unit used to measure the intensity of a sound or the power level of an electrical signal by comparing it with a given level on a logarithmic scale.

hz Hertz : The unit of frequency, equal to one cycle per second.

ao Adam Optimizer: An optimization algorithm for training deep learning models, known for its efficiency and low memory requirements.

relu Rectified Linear Unit: An activation function used in neural networks, defined as the positive part of its argument, facilitating faster training and mitigating the vanishing gradient problem.

ACKNOWLEDGMENTS

We, Team D, comprising Koushik Mannaru, Sreeja Komma Reddy, and Sravani Pilla, would like to express our deepest gratitude to all those who supported and guided us throughout this music genre classification project. We sincerely thank our professor, Unal Sakoglu, for his valuable guidance, support, and feedback, which is the main reason for shaping our research and its success. Additionally, we thank our colleagues and friends who provided their insights and assistance during meetings. Their contributions were crucial in refining our approach and overcoming the challenges we encountered.

Lastly, we are grateful for the resources and support provided by UMBC which made this project possible.

TEAM MEMBERS' CONTRIBUTIONS

Sravani pilla worked on the literature and background review, provided valuable insights into tuning the project, actively participated in Feature Engineering and Exploratory Data Analysis, helped in preparing and processing new data set and also helped in evaluating the model performance.

Sreeja Komma Reddy provided studies related to the work done on Audio Classification, extracted and processed crucial features required for the project, actively participated in Exploratory Data Analysis and collected and tuned audio files according to the model and helped in building and hyper tuning the model

Koushik Mannaru worked on Exploratory Data Analysis and preprocessing of data, provided studies for comparison and Model architecture, studied the audio files and their characteristics, built and evaluated the model on all collected data.

Everyone participated in Model testing as it needs high computation power. We shared the work among ourselves with different hyperparameters to select the best model

Name	Duties	Achievements
Koushik Mannaru	Model Architecture and Hyper tuning Exploratory Data Analysis Studies for results comparison and Data characteristics	Successfully built Model with best parameters Data Analysis and better understanding of Audio files
Sreeja Komma Reddy	Data Collection and Pre-processing Data study and Exploratory Data Analysis Feature Extraction Model Tuning	Successfully collected and Processing Data Data Analysis and successfully achieved for better model

Sravani Pilla	Literature and Background review Feature Engineering Exploratory Data Analysis and Preprocessing New Dataset Preparation Model Evaluation and Fine Tuning	Successfully captured Essential features, performed Data Analysis,prepared new dataset and worked proactively for achieving better model
---------------	---	--

TABLE OF CONTENTS

ABSTRACT.....	i
LIST OF ABBREVIATIONS AND SYMBOLS	ii
ACKNOWLEDGMENTS (& TEAM MEMBERS' CONTRIB.).....	iii
LIST OF FIGURES	vi
I. INTRODUCTION.....	1
I.1 Problem statement	
I.2 Background & Literature Review/Survey	
I.3 Dataset Overview	
I.4 Specifications	
I.5 Processing steps	
II. METHODS.....	3
II.1 Overview	
II.2 Feature Extraction	
II.3 Description of Data Processing	
II.4 Exploratory Data Analysis	
III. RESULTS	7
IV. CONCLUSIONS AND FUTURE WORK.....	10
V. REFERENCES.....	11
VI. APPENDIX.....	12
VI.1 Programming Codes	

LIST OF FIGURES

2.1 Exploratory Data Analysis Numerical Features	4
2.2 Exploratory Data Analysis- Visualizing Sound.....	4
2.3 Exploratory Data Analysis- Spectrogram Analysis.....	5
2.4 Exploratory Data Analysis- Beyond the Ear.....	5
2.5 Model Parameters and K-Fold.....	6
2.6 Model Architecture.....	6
3.1 Training Accuracy.....	7
3.2 Testing Accuracy.....	7
3.3 Newly Created Dataset Accuracy.....	7
3.4 Classification on Different Dataset with same approach [Ref 3].....	8
3.5 works Done on GTZAN Dataset with Different Evaluation [Ref 8].....	9

I. INTRODUCTION

I.1 Problem statement:

Music genre classification is a notably demanding and vital task in the area of music information retrieval. As digital music collections and streaming platforms continue to expand rapidly, considerable effort is being focused on the development of efficient and reliable methods to classify audio tracks into predefined categories. Music genre classification is a difficult job largely out of the sheer complexity of music forms, which can differ in aspects such as rhythm, melody, harmony, and timbre. The target of this venture is to create a reliable model that can categorize audio tracks into one of ten genres. To handle this issue, the current work employs advanced deep learning techniques known as Convolutional Neural Networks and Long Short-Term Memory networks. This project aims to develop a highly accurate and consistent classifier that may be used to organize and retrieve music files for a variety of applications.

I.2 Background and Literature Review:

The development of music genre classification as a computer science problem can be dated back to the pioneering work by Tzanetakis and Cook, who proposed an approach based on the use of timbre, pitch, and rhythm features jointly with standard classifiers. While their research showcased the potential of automatic music classification based on machine learning techniques, the approach's performance was limited to the inherent simplicity of the used features. Specifically, as early models failed to identify more complex and intricate patterns in audio data for more refined classification, their performance is thus limited by the features and cannot be optimal in terms of accuracy and robustness. Nonetheless, Tzanetakis and Cook's innovative research sparked further exploration of more complex feature extraction and classification methods.

The limitations of classical machine learning models in capturing intricate audio patterns motivated researchers to explore the applicability of deep learning. In particular, the performance of deep learning models, such as Convolutional Neural Networks and Recurrent Neural Networks, namely, Long Short-Term Memory have been reported to be superior to other models. Studies have also revealed that CNNs are able to learn local audio patterns while RNNs can be used to learn temporal dependencies. Particularly, a hybrid approach with both CNN and RNN has shown superior classification performance. For example, Mohsin Ashraf and colleagues introduced a hybrid CNN and RNN in their study, "A Hybrid CNN and RNN Variant Model for Music Classification" the study architecture integrated CNN with an RNN model, including LSTM, Bi-LSTM, GRU, and Bi-GRU. Authors proved that this integrated model could effectively capture the spatial and temporal characteristics of music, respectively, confirming the flexibility and effectiveness of deep learning in music classification.. The model performance was evaluated using Melspectrogram and Mel-Frequency Cepstral Coefficients.

I.3 Dataset Overview:

For this project, the primary dataset utilized is the GTZAN dataset, a well-known collection specifically designed for music genre classification tasks. This dataset is readily available on Kaggle and has been extensively used in the literature as a benchmark for evaluating music classification algorithms.

Link: **<https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification/data>**

I.4 Specifications:

Number of Tracks: 1,000 audio tracks

Duration: Each track is 30 seconds long

Genres: The dataset includes 10 genres, with 100 tracks per genre. The genres are Blues, Classical, Country, Disco, Hip-Hop, Jazz, Metal, Pop, Reggae, and Rock.

Sample Rate: 22,050 Hz

Channels: Mono (1 channel)

Sample Format: 16-bit

In addition to the GTZAN dataset, a supplementary test dataset comprising 100 songs from free music archives was developed to enhance model evaluation. The inclusion of this dataset addresses the concern that the GTZAN dataset, being relatively old, may not fully represent contemporary music trends. The new dataset, collected in 2023, provides a more current evaluation benchmark.

I.5 Processing Steps:

Audio Conversion: The additional dataset was processed to match the format of the GTZAN dataset.

Sample Rate: Adjusted to 22,050 Hz

Channels: Converted to mono (1 channel)

Sample Format: Set to 16-bit

These steps ensure consistency in audio properties between the GTZAN dataset and the additional evaluation dataset, allowing for a fair comparison and comprehensive assessment of the model's performance.

I. METHODS

II.1 Overview:

The methodology for this project involves several key steps: feature extraction from the audio files, data preprocessing, model architecture design, training, and evaluation. The core approach integrates Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks to leverage both spatial and temporal characteristics of the audio data.

II.2 Feature Extraction:

Feature extraction is a critical step in music genre classification, as the quality of the features significantly impacts the performance of the classification model. We utilized Librosa, a powerful Python library for audio and music analysis, to extract essential features from the audio files. Specifically, we extracted MFCC, spectral centroid, chroma feature, and spectral contrast. These features capture various aspects of the audio signal, including its timbral, harmonic, and rhythmic characteristics.

Using Librosa, the features were extracted with a hop length of 512, resulting in 1293 numerical values for each feature. These values were then truncated to 512, producing a uniform shape of (512, 33) for each song. This preprocessing step ensures that the input to the neural network is consistent, facilitating more effective learning. This feature extraction approach is inspired by the work of Li Guo, Zhiwei Gu, and Tianchi Liu, who demonstrated the efficacy of these features in their study. When we tried to extract more than 512 of numerical values, we observed high bias in the model where the training accuracy was 83% but testing accuracy only 38%—we truncated features from 1,293 to 512 and simplified the model architecture, enhancing its generalization capabilities. This preprocessing step ensures that the input to the neural network is consistent, facilitating more effective learning. This feature extraction approach is inspired by the work of Li Guo, Zhiwei Gu, and Tianchi Liu, who demonstrated the efficacy of these features in their study.

II.3 Data Preprocessing

GTZAN Dataset:

As optimal music genre classification model performance and accuracy are contingent on adequate preprocessing, we took appropriate steps to guarantee input data consistency and quality for the GTZAN dataset and the newly collected data. Preprocessing steps are highlighted below: Silence at the beginning and the conclusion of audio tracks brings extra “noise” and variations to the input data. Thus, we employed the librosa effects trim function, which mechanically trims silent node sections from the start and end of each audio track. Audio files that are corrupted include mistakes and irregularities in their contents and should not be used. Files that could not be uploaded or otherwise loaded into the environment were deleted before preprocessing commences.

New Dataset:

In this study, the sample rate, audio channels, and bit sample size differ between the newly collected audio files and the GTZAN dataset. The characteristics of the new dataset relative to the GTZAN files were converted into a 22050 Hz sample rate, a mono channel, and a 16-bit sample format. All the data files were preprocessed with the same process in the new dataset. Preprocessing steps such as the removal of silence were applied and deleted the corrupted audio files. It is essential to conduct preprocessing steps to ensure that all the datasets are accurate, clean, and the features can be obtained and modeled accurately.

II.4 Exploratory Data Analysis:

Before diving into model training, dataset is explored to understand the characteristics and patterns within the music files. Librosa library, a powerful tool in Python was used for analyzing and extracting features from music and audio signals. It is also widely used in fields like music information retrieval sound classification and signal processing. Insights Gained: By visually inspecting the audio, patterns in waveforms and spectrograms that could correlate with genre-specific traits were discovered.

```
y, sr = librosa.load('blues.00000.wav')

print('\nNumerical Features :', y)
print('\n shape of the converted Audio files :', np.shape(y))
print('\n Sample Rate (KHz):', sr)

print('Length of the Audio File in seconds:', 661794/22050)
```

```
Numerical Features : [ 0.00732422  0.01660156  0.00762939 ... -0.05560303 -0.06106567
-0.06417847]

 shape of the converted Audio files : (661794,)

 Sample Rate (KHz): 22050
 Length of the Audio File in seconds: 30.013333333333332
```

Fig.2.1 Numerical Feature for EDA

Visualizing Sound:

Sound waves are plotted to analyze the amplitude variations over time, which gives insight into the loudness and energy of the tracks.

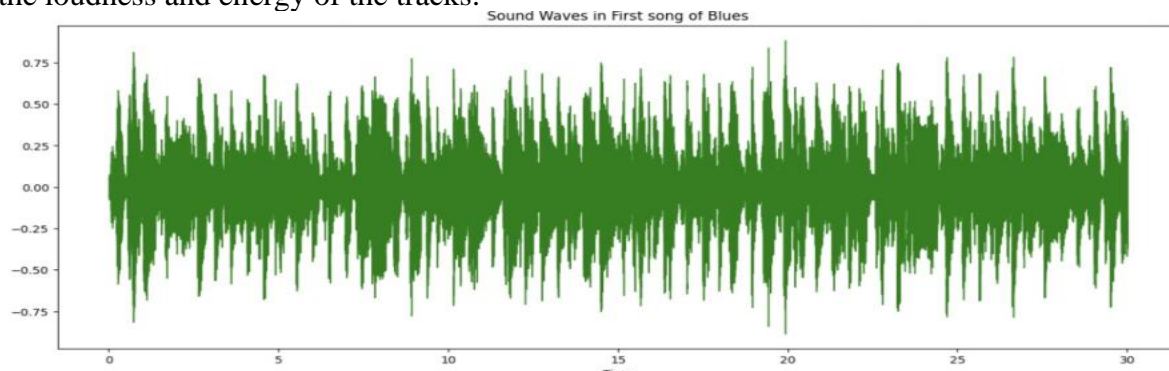


Fig. 2.2 Exploratory Data Analysis – Visualizing Sound

Spectrogram Analysis:

Using Mel Spectrograms, sound is translated into a visual landscape, revealing how frequencies vary with time which is an essential feature for genre classification. Tempo and the speed of track are examined which along with rhythmic patterns helps in distinguishing between genres.

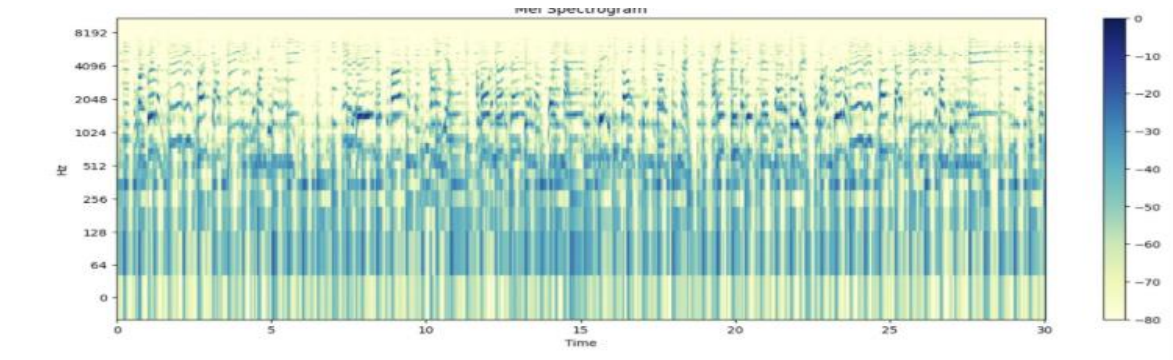


Fig. 2.3 Exploratory Data Analysis – Spectrogram Analysis

Beyond the Ear:

Harmonics vs. Percussive Elements: Our analysis separates harmonics from percussive sounds, which are often challenging to differentiate by ear, yet critical for understanding the music's texture.

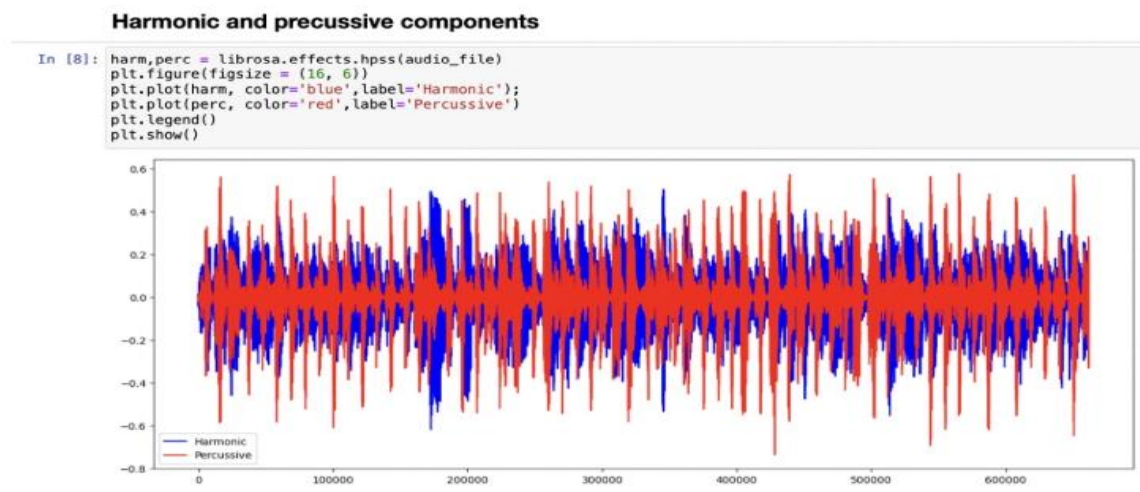


Fig. 2.4 Exploratory Data Analysis- Beyond the ear

II.5 Model Overview:

The architecture of the model combines Convolutional Neural Networks and Long Short-Term Memory networks, and includes 141,386 trainable parameters. The parameters are represented in the convolutional layers, LSTM layers, and dense layers and allow the model to learn spatial and temporal features in audio data. More precisely, hyperparameters such as the kernel size, the number of filters, and the dropout rate decision were adjusted to avoid overfitting and provide the basis for appropriate performance. In this way, the hyperparameters ensured that the model's complexity did not suppress its generalization. Moreover, the model was trained through rigorous optimization using the Adam optimizer, and the loss function of categorical cross-entropy was used to calculate the gap between the predicted genre and its real label. In other words, the loss function and hyperparameters parameters guided the model during the training process to update the weights of the model in each iteration, ensuring that the loss is minimized and classification accuracy is maximized.

<pre>model.summary()</pre>		
Model: "sequential"		
Layer (type)	Output Shape	Param #
conv1d (Conv1D)	(None, 510, 256)	25600
max_pooling1d (MaxPooling1D)	(None, 255, 256)	0
conv1d_1 (Conv1D)	(None, 253, 128)	98432
max_pooling1d_1 (MaxPooling1D)	(None, 126, 128)	0
lstm (LSTM)	(None, 126, 64)	49408
lstm_1 (LSTM)	(None, 32)	12416
dropout (Dropout)	(None, 32)	0
dense (Dense)	(None, 10)	330
<hr/> Total params: 186186 (727.29 KB) Trainable params: 186186 (727.29 KB) Non-trainable params: 0 (0.00 Byte)		

<pre>model.add(Dropout(0.5)) model.add(Dense(units=10, activation='softmax')) # Compile the model model.compile(loss='categorical_crossentropy', optimizer=Adam(), metrics=['accuracy']) # Train the model history = model.fit(x_train_fold, y_train_fold, batch_size=32, epochs=400, verbose=0) # Evaluate the model on the validation set scores = model.evaluate(x_val_fold, y_val_fold, verbose=0) # Append evaluation metrics to lists acc_per_fold.append(scores[1] * 100) # Accuracy loss_per_fold.append(scores[0]) # Loss print(f"Validation Accuracy: {scores[1] * 100:.2f}%") print(f"Validation Loss: {scores[0]:.4f}") # Print average metrics across all folds print(f"Average Validation Accuracy: {sum(acc_per_fold) / len(acc_per_fold):.2f}% (+/- {np.std(acc_per_fold):.2f})") print(f"Average Validation Loss: {sum(loss_per_fold) / len(loss_per_fold):.4f}") > Training on Fold 1 Validation Accuracy: 62.00% Validation Loss: 1.2887 Training on Fold 2</pre>		
--	--	--

Fig.2.5 Model Parameters and K-Fold

Additionally, since the model's generalization is critical, k-fold cross-validation with k=5 was employed. The statement means that the dataset was split into five sections, with four of them sending for the training and the remaining one for evaluation. The process was repeated five times to identify a consistent accuracy level. The average accuracy of 65% across the GTZAN dataset indicates that model can recognize the patterns of genres.

Additionally, the segmentation of the architecture provides insights into the model's functionality. Thus, convolutional layers identify spatial features, the LSTM layers relate to the temporal dependencies, the dropout regularizes to avoid overfitting, while the dense classification layer links the results with output classes.

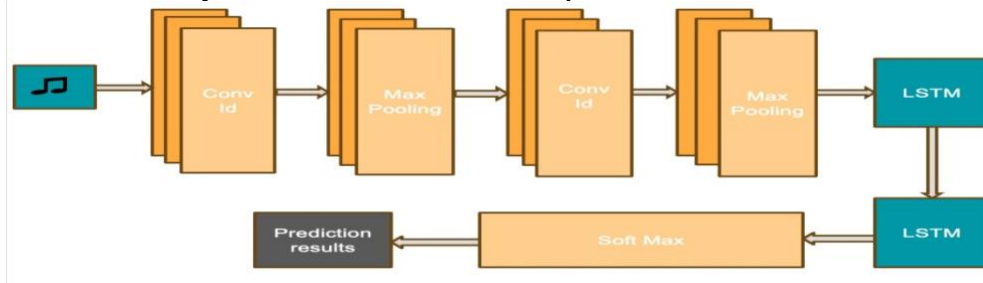
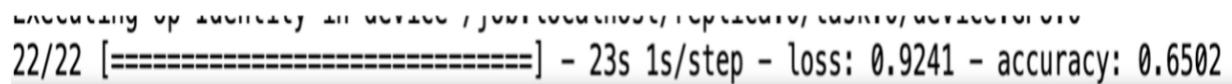


Fig.2.6 Model Architecture

This model provides sufficient evidence of automation of the classification process that may be exploited across various applications from device design to music content creation.

III. RESULTS


The results obtained from the music genre classification model show its potential in identifying genre-specific patterns when working with the GTZAN dataset. More specifically, the model generally performs well in this regard with the following major observations. Initially, following extensive training and validation, the model shows a relatively high training accuracy of almost 65% on the dataset. This implies that the model was able to understand the characteristics of the audio data very well. For this reason, the model has shown potential to achieve a good generalization effect when used to classify genres in a real situation.



22/22 [=====] - 23s 1s/step - loss: 0.9241 - accuracy: 0.6502

Fig. 3.1 Training Accuracy

As well, the testing accuracy on the unseen GTZAN dataset consumed slightly lowers at about 53%. Even though this accuracy is still high, the notable gap between the training and testing accuracies can be seen as an indicator of the presence of overfitting. Therefore, more regularization and maybe other model optimization mechanisms should be applied to reduce overfitting and thus enhance the model's generalization trend during use.



Mean Test Accuracy: 0.3142857290804386

Fig. 3.2 Testing Accuracy

Additionally, on the new test dataset, the model generalization power is less convincing, and the obtained accuracy is about 31%, further showing the need for more generalization before using the model in real-life instances. Evidently, the model, as it stands now, only has a promising performance on GTZAN data. However, more work needs to be done in terms of generalization and robustness checks to determine if this is sufficient in other scenarios. Therefore, further parameter and hyperparameter tweaking assisted by adding more regularization mechanisms is required to improve performance in this music genre classification model.



Mean Test Accuracy: 0.5346939010279519

Fig. 3.3 Newly Created Dataset Accuracy

Comparison with Similar works:

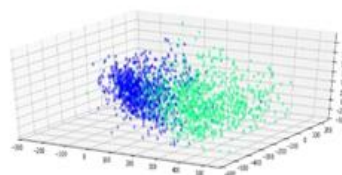
In comparison with similar works in the field of music genre classification, our model demonstrates competitive performance while addressing the challenges posed by different datasets and classification scenarios. In the study conducted by Li Guo, Zhiwei Gu, and Tianchi Liu, a Linear Support Vector Machine (SVM) [Ref 3] classifier achieved a training accuracy of 80.21% and a testing accuracy of 59.08%, whereas a Softmax classifier attained a training accuracy of 52.87% and a testing accuracy of 51.03%. This study initially encountered high bias, later, when they attempted to mitigate bias resulting in reduced accuracy to approximately 50%. However, it's noteworthy that the model used in their works employed a different dataset, other than the GTZAN dataset, with a larger size and 16 distinct classes for classification, potentially introducing different challenges and opportunities compared to the GTZAN dataset

B. Data Preprocessing

25000 of all 106,574 tracks were used in this project for computational efficiency and information integrity. These 25000 tracks were splitted into training set, validation set and test set with sizes of 19922, 2505, 2573, respectively, and all training data was shuffled randomly. Therefore, training examples were represented as a large matrix of 19922 rows and 519 columns, with 518 features and a label of the genre.

Since most of the algorithms that we used usually treat vectors as inputs, we applied either PCA to our matrix or flattened the matrix to an extremely large vector, and then used this structure as a training example. Below is a visualization of a subset of our dataset with only two genres (Instrumental, Hip-Hop) after applying PCA to reduce the input to three dimensions:

(Instrumental, Hip-Hop) after applying PCA to reduce the input to three dimensions:



IV. METHODS

We built two baseline models: Support vector machine with linear kernel and softmax regression, with the following performance:

Classifier	Train accuracy	Test accuracy
SVM Linear	0.8021	0.5908
Softmax	0.5287	0.5103

Fig.3.4 Classification on Different Dataset with same approach [Ref 3]

Furthermore, in the research conducted by Tao Feng on the GTZAN dataset was utilized, but the evaluation focused on 2-class, 3-class, and 4-class classification scenarios, unlike our study, which evaluated accuracy across all 10 music genres. Despite differing evaluation methodologies, our model showcased robust performance, achieving a notable testing accuracy of approximately 53% on the GTZAN dataset. This highlights the versatility and adaptability of our model in handling multi-class classification tasks, underscoring its efficacy in accurately categorizing music into diverse genre categories.

Classic VS. Metal	NN	DBN
	CG 3 line search	CG with 3 line search
Accuracy on train set	100% (120 out of 120)	100% (120 out of 120)
Accuracy on test set	97.5% (78 out of 80)	98.75% (79 out of 80)
More stats:	H ~ (500 500 1000)	H ~ (1000 1500 2000)
H hidden nodes		
T running time	T ~ 18 s	T ~ 7 min

1. DBN needs much larger network and more iterations (>50k) in the BP stage to get better performance. Much more time/computation consuming!
2. NN use smaller size network and converge very fast (after 6k).

(a) 2 class classification

Classic, Metal and Blues	NN	DBN
	CG 3 line search	CG with 3 line search
Accuracy on train set	90% (162 out of 180)	91% (164 out of 180)
Accuracy on test set	70.8% (85 out of 120)	69.16% (83 out of 120)
More stats:	H ~ (500 500 1000)	H ~ (1000 1000 2500)
H hidden nodes		
T running time	T ~ 26 s	T ~ 20+ min

(b) 3-class classification

Classic, Metal, Blues and Dixie	NN	DBN
	CG 3 line search	CG with 3 line search
Accuracy on train set	90.83% (218 out of 240)	89.17% (166 out of 240)
Accuracy on test set	63.75% (102 out of 160)	51.88% (83 out of 160)
More stats:	H ~ (500 500 1000)	H ~ (1000 1000 2500)
H hidden nodes		
T running time	T ~ 26 s	T ~ 20+ min

(c) 4-class classification

Figure 5: Initial experiment result

iterations than NN to achieve high performance. For the 3 class classification, the NN and DBN is also competitive with each other, while DBN has the worse problem of over-fitting (higher accuracy on training set and less accuracy on testing set). the experiment on 4 class classification shows that NN outperform DBN to a noticeable scale.

4.5 Second experiment results

Classic, Metal and Blues	NN	DBN
	CG 3 line search	CG with 3 line search
Accuracy on train set	92.89% (2508 out of 2700)	94.5% (2521 out of 2700)
Accuracy on test set	77.17% (1389 out of 1800)	77.94% (1403 out of 1800)
More stats:	H ~ (500 500 1000)	H ~ (1000 1500 2500)
H hidden nodes		
T running time	T ~ 50+ min	T ~ 120+ min

(a) 3-genre classification with larger dataset

Classic, Metal, Dixie and Blues	NN	DBN
	CG 3 line search	CG with 3 line search
Accuracy on train set	94.69% (3406 out of 3600)	75.3% (2712 out of 3600)
Accuracy on test set	60.46% (1451 out of 2400)	61.15% (1467 out of 2400)
More stats:	H ~ (500 500 1000)	H ~ (1000 1500 2500)
H hidden nodes		
T running time	T ~ 60 min	T ~ 160+ min

(b) 4-genre classification with larger dataset

Fig.3.5 works Done on GTZAN Dataset with Different Evaluation [Ref 8]

Overall, while our model's performance may vary from previous works due to differences in datasets, classification scenarios, and evaluation methodologies, it stands as a promising contribution to the field of music genre classification.

IV. CONCLUSION AND FUTURE WORK

we have successfully leveraged deep learning methodologies to attain good accuracies in music genre classification, achieving up to 65% training and 53% testing accuracy. By developing a hybrid architecture with Convolutional Neural Networks (CNNs) for feature extraction and Long Short-Term Memory (LSTM) networks for pattern recognition, we pushed the boundaries of audio data classification techniques. This approach has facilitated the identification of in-depth patterns within music, leading to enhanced classification accuracy and performance.

Furthermore, we have extended beyond regular datasets by incorporating a diverse test dataset sourced from free online music archives. Initial evaluations showed an accuracy of 31%, creating a path for future improvement and refinement. By Overcoming significant challenges such as data corruption, computational constraints, and manual data verification, we ensured high data integrity and model reliability throughout the project lifecycle. These achievements highlight our commitment to advancing the field of music genre classification and its applications.

In future , our vision extends to the r music recommendation systems. By leveraging the foundation laid through our classification model, we aim to explore and develop innovative approaches to music recommendation. By incorporating user preferences, listening habits, and contextual information, we plan to create personalized music recommendation systems that enhance user experience on streaming platforms. With continued research and collaboration, we will shape the future of music discovery, enriching the lives of music enthusiasts.

V. REFERENCES

1. Tzanetakis, G., & Cook, P. (2002). Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5), 293-302. <https://ieeexplore.ieee.org/document/1021072>
2. A Hybrid CNN and RNN Variant Model for Music Classification <https://www.mdpi.com/2076-3417/13/3/1476>
3. Stanford CS229. (2017). Music Genre Classification. <https://cs229.stanford.edu/proj2017/final-reports/5244969.pdf>
4. Manwani, A. (2020). Song Genre Classification from Audio Data using ML. *Kaggle*. <https://www.kaggle.com/code/ajaymanwani/song-genre-classification-from-audio-data-using-ml>
5. Udupa, C. (2020). Music Genre Detection using Machine Learning. *Kaggle*. <https://www.kaggle.com/code/chinmayaudupa/music-genre-detection-using-machine-learning>
6. Analytics Vidhya. (2022). Music Genre Classification Project Using Machine Learning Techniques. <https://www.analyticsvidhya.com/blog/2022/03/music-genre-classification-project-using-machine-learning-techniques/>
7. Olteanu, A. (n.d.). GTZAN Dataset - Music Genre Classification. *Kaggle*. <https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification/data>
8. Feng, T. (2014). *Deep learning for music genre classification*. University of Illinois. Retrieved from https://courses.engr.illinois.edu/ece544na/fa2014/Tao_Feng.pdf
9. Olteanu, A. (n.d.). *GTZAN Dataset - Music Genre Classification*. *Kaggle*. Retrieved from <https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification/data>

VI. APPENDICES

VI. Code:

DATA_606_Project.ipynb:

```
from google.colab import drive
drive.mount('/content/drive')
# Usual Libraries
import pandas as pd
import numpy as np
import seaborn as sns
import os
import matplotlib.pyplot as plt
%matplotlib inline
import sklearn

# Librosa (the mother of audio files)
import librosa
import librosa.display
import IPython.display as ipd
import warnings
warnings.filterwarnings('ignore')

## Initial Audio Analysis
#loading audio file
y, sr = librosa.load('/content/drive/MyDrive/genres_original/blues/blues.00000.wav')

print("\nNumerical Features :", y)
print("\n shape of the converted Audio files :", np.shape(y))
print("\n Sample Rate (Hz):", sr)

print('Length of the Audio File in seconds:', len(y)/sr)

#Trimming silence
audio_file, _ = librosa.effects.trim(y)

print('Audio File:', audio_file, '\n')
print('Audio File shape:', np.shape(audio_file))

## Visual representation of Audio File
filelocation='/content/drive/MyDrive/genres_original'
hop_length=512
genre_names = ["blues", "classical", "country", "disco", "hiphop", "jazz", "metal", "pop",
"reggae", "rock"]
```

```

plt.figure(figsize = (16, 6))
librosa.display.waveshow(y = audio_file, sr = sr, color = "green");
plt.title("Sound Waves in First song of Blues");

#Mel spectrogram
S = librosa.feature.melspectrogram(y=y, sr=sr)
S_DB = librosa.amplitude_to_db(S, ref=np.max)
plt.figure(figsize=(16, 6))
librosa.display.specshow(S_DB, sr=sr, hop_length=512, x_axis='time',
y_axis='log',cmap='YlGnBu')
plt.colorbar()
plt.title("Mel Spectrogram",)
plt.show()

## Harmonic and percussive components
harm,perc = librosa.effects.hpss(audio_file)
plt.figure(figsize = (16, 6))
plt.plot(harm, color='blue',label='Harmonic');
plt.plot(perc, color='red',label='Percussive')
plt.legend()
plt.show()

#Tempo analysis
tempo, _ = librosa.beat.beat_track(y=y, sr=sr)
print('Beats per Minute is ',tempo)

## Feature Engineering
#Extracting MFCCs
mfcc = librosa.feature.mfcc(y=y, sr=sr, hop_length=512, n_mfcc=13)
mfcc.shape
mfcc.T[0:1200,:].shape

mfcc

spectral_center = librosa.feature.spectral_centroid(y=y, sr=sr, hop_length=512)
spectral_center.shape

chroma = librosa.feature.chroma_stft(y=y, sr=sr, hop_length=512)
chroma.shape

spectral_contrast = librosa.feature.spectral_contrast(y=y, sr=sr, hop_length=512)
spectral_contrast.shape

## Batch processing of audio files for feature extraction
data=np.zeros((50, 128, 33), dtype=np.float64)
data.shape

```

```

x=librosa.feature.melspectrogram(y=y, sr=sr, n_mels=128)
x.shape

q=librosa.feature.melspectrogram(y=y, sr=sr)
q.shape

genres_dir = "/content/drive/MyDrive/genres_original"
data = np.zeros((999,512,33), dtype=np.float64)
target=[]
# List of genre names
genre_names = ["blues", "classical", "country", "disco", "hiphop", "jazz", "metal", "pop",
"reggae", "rock"]
i=0
for genre in genre_names:
    genre_path = os.path.join(genres_dir, genre)

    # Loop through each file in the genre folder
    for filename in os.listdir(genre_path):
        file_path = os.path.join(genre_path, filename)
        y, sr = librosa.load(file_path)
        y, _ = librosa.effects.trim(y)
        mfcc = librosa.feature.mfcc(y=y, sr=sr, hop_length=512, n_mfcc=13)
        spectral_center = librosa.feature.spectral_centroid(y=y, sr=sr, hop_length=512)
        chroma = librosa.feature.chroma_stft(y=y, sr=sr, hop_length=512)
        spectral_contrast = librosa.feature.spectral_contrast(y=y, sr=sr, hop_length=512)
        target.append(genre)
        data[i, :, 0:13] = mfcc.T[0:512,:]
        data[i, :, 13:14] = spectral_center.T[0:512, :]
        data[i, :, 14:26] = chroma.T[0:512, :]
        data[i, :, 26:33] = spectral_contrast.T[0:512, :]
        print("Numerical features extracted from audio file %i of %i." % (i + 1, 999))
        i+=1
y=np.zeros((999,10))
for i,genre in enumerate(target):
    ind=genre_names.index(genre)
    y[i,ind]=1
y

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv1D, MaxPooling1D, LSTM, Dense, Flatten,
Reshape, Dropout
from tensorflow.keras.optimizers import Adam

```

```

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv1D, MaxPooling1D, LSTM, Dense, Flatten,
Reshape, Dropout
from tensorflow.keras.optimizers import Adam

# Define the model
model = Sequential()

# Add Convolutional layers
model.add(Conv1D(filters=256, kernel_size=3, activation='relu', input_shape=(512, 33)))
model.add(MaxPooling1D(pool_size=2))
model.add(Conv1D(filters=128, kernel_size=3, activation='relu'))
model.add(MaxPooling1D(pool_size=2))

# Flatten the output for LSTM
#model.add(Flatten())

# Reshape for LSTM input
#model.add(Reshape((32, 120))) # Reshape to (timesteps, features)

# Add LSTM layers
model.add(LSTM(units=64, dropout=0.2, recurrent_dropout=0.2, return_sequences=True))
model.add(LSTM(units=32, dropout=0.2, recurrent_dropout=0.2))

# Add Dropout for regularization
model.add(Dropout(0.5))

# Output layer
model.add(Dense(units=10, activation='softmax'))

# Compile the model with a lower learning rate
opt = Adam()
model.compile(loss='categorical_crossentropy', optimizer=opt, metrics=['accuracy'])

# Print model summary
model.summary()

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(data, y, test_size=0.25, random_state=42)

genre_names = ["blues", "classical", "country", "disco", "hiphop", "jazz", "metal", "pop",
"reggae", "rock"]
batch_size = 35 # num of training examples per minibatch
num_epochs = 400
history = model.fit(x_train, y_train, batch_size=batch_size, epochs=num_epochs, verbose=0)

```

```

# Calculate the mean training accuracy
mean_training_accuracy = np.mean(history.history['accuracy'])
print("Mean Training Accuracy:", mean_training_accuracy)

import math
# score, accuracy = model.evaluate(
#     x_test, y_test, batch_size=batch_size, verbose=1
# )

num_batches = len(x_test) // batch_size
accuracies = []

for i in range(num_batches):
    start = i * batch_size
    end = (i + 1) * batch_size
    batch_x = x_test[start:end]
    batch_y = y_test[start:end]
    _, batch_accuracy = model.evaluate(batch_x, batch_y, verbose=0)
    accuracies.append(batch_accuracy)

mean_test_accuracy = np.mean(accuracies)
print("Mean Test Accuracy:", mean_test_accuracy)

model.save('m01.h5')

```

Testing_and_evaluation.ipynb:

```

from google.colab import drive
drive.mount('/content/drive')

from keras.models import load_model
model = load_model('/content/m01.h5')

# Usual Libraries
import pandas as pd
import numpy as np
import seaborn as sns
import os
import matplotlib.pyplot as plt
%matplotlib inline
import sklearn

# Librosa (the mother of audio files)
import librosa
import librosa.display

```



```

import IPython.display as ipd
import warnings
warnings.filterwarnings('ignore')

model.summary()

from keras.models import load_model
model = load_model('/content/m01.h5')

genres_dir = "/content/drive/MyDrive/genres_original"
data = np.zeros((999,512,33), dtype=np.float64)
target=[]
# List of genre names
genre_names = ["blues", "classical", "country", "disco", "hiphop", "jazz", "metal", "pop",
"reggae", "rock"]
i=0
for genre in genre_names:
    genre_path = os.path.join(genres_dir, genre)

    # Loop through each file in the genre folder
    for filename in os.listdir(genre_path):
        file_path = os.path.join(genre_path, filename)
        y, sr = librosa.load(file_path)
        y, _ = librosa.effects.trim(y)
        mfcc = librosa.feature.mfcc(y=y, sr=sr, hop_length=512, n_mfcc=13)
        spectral_center = librosa.feature.spectral_centroid(y=y, sr=sr, hop_length=512)
        chroma = librosa.feature.chroma_stft(y=y, sr=sr, hop_length=512)
        spectral_contrast = librosa.feature.spectral_contrast(y=y, sr=sr, hop_length=512)
        target.append(genre)
        data[i, :, 0:13] = mfcc.T[0:512,:]
        data[i, :, 13:14] = spectral_center.T[0:512, :]
        data[i, :, 14:26] = chroma.T[0:512, :]
        data[i, :, 26:33] = spectral_contrast.T[0:512, :]
        print("Numerical features extracted from audio file %i of %i." % (i + 1, 999))
        i+=1

y=np.zeros((999,10))
for i,genre in enumerate(target):
    ind=genre_names.index(genre)
    y[i,ind]=1

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(data, y, test_size=0.25, random_state=42)
import math
# score, accuracy = model.evaluate(
#     x_test, y_test, batch_size=batch_size, verbose=1

```

```

# )
batch_size = 35 # num of training examples per minibatch
num_epochs = 400
num_batches = math.ceil(len(x_test) // batch_size)
accuracies = []

for i in range(num_batches):
    start = i * batch_size
    end = (i + 1) * batch_size
    batch_x = x_test[start:end]
    batch_y = y_test[start:end]
    _, batch_accuracy = model.evaluate(batch_x, batch_y, verbose=0)
    accuracies.append(batch_accuracy)

mean_test_accuracy = round(np.mean(accuracies),4)
print("Accuracy of GTZAN Test Dataset:", mean_test_accuracy)

genres_dir = "/content/drive/MyDrive/songs_final"
val_data = np.zeros((100,512,33), dtype=np.float64)
target=[]
# List of genre names
genre_names = ["blues", "classical", "country", "disco", "hiphop", "jazz", "metal", "pop",
"reggae", "rock"]
i=0
for genre in genre_names:
    genre_path = os.path.join(genres_dir, genre)

    # Loop through each file in the genre folder
    for filename in os.listdir(genre_path):
        file_path = os.path.join(genre_path, filename)
        y, sr = librosa.load(file_path)
        y, _ = librosa.effects.trim(y)
        mfcc = librosa.feature.mfcc(y=y, sr=sr, hop_length=512, n_mfcc=13)
        spectral_center = librosa.feature.spectral_centroid(y=y, sr=sr, hop_length=512)
        chroma = librosa.feature.chroma_stft(y=y, sr=sr, hop_length=512)
        spectral_contrast = librosa.feature.spectral_contrast(y=y, sr=sr, hop_length=512)
        target.append(genre)
        print(file_path)
        val_data[i, :, 0:13] = mfcc.T[0:512,:]
        val_data[i, :, 13:14] = spectral_center.T[0:512, :]
        val_data[i, :, 14:26] = chroma.T[0:512, :]
        val_data[i, :, 26:33] = spectral_contrast.T[0:512, :]
        print("Numerical features extracted from audio file %i of %i." % (i + 1, 100))
        i+=1

val_y=np.zeros((100,10))

```

```

for i,genre in enumerate(target):
    ind=genre_names.index(genre)
    val_y[i,ind]=1

import math
# score, accuracy = model.evaluate(
#     x_test, y_test, batch_size=batch_size, verbose=1
# )

num_batches = math.ceil(len(val_data) // batch_size)
accuracies = []

for i in range(num_batches):
    start = i * batch_size
    end = (i + 1) * batch_size
    batch_x = val_data[start:end]
    batch_y = val_y[start:end]
    _, batch_accuracy = model.evaluate(batch_x, batch_y, verbose=0)
    accuracies.append(batch_accuracy)

mean_test_accuracy = round(np.mean(accuracies),4)
print("Accuracy on newly collected data:", mean_test_accuracy)

```

Songs_genre_classification_dataset.ipynb:

```

from pydub import AudioSegment
import os
input_main_folder_path = "/Users/sravanipilla/Desktop/songs" #input folder path

output_main_folder_path = "/Users/sravanipilla/Desktop/songs_final" #output folder path

# Creating the output folder if it doesn't exist in that location
if not os.path.exists(output_main_folder_path):
    os.mkdir(output_main_folder_path)

# Getting the list of genre subfolders
genre_folders = [f for f in os.listdir(input_main_folder_path) if
os.path.isdir(os.path.join(input_main_folder_path, f))]
# Looping through each genre folder

for genre in genre_folders:

    # Path to the current genre input and output folders
    genre_input_folder_path = os.path.join(input_main_folder_path, genre)
    genre_output_folder_path = os.path.join(output_main_folder_path, genre)

```

```

# Creating the genre-specific output folder if it doesn't exist

if not os.path.exists(genre_output_folder_path):
    os.mkdir(genre_output_folder_path)

# Listing all the files in the input genre folder

file_list = os.listdir(genre_input_folder_path)

# Filter out non-MP3 files if necessary
file_list = [file for file in file_list if file.endswith('.mp3')]

# Loop through each file in the genre folder

for i, file_name in enumerate(file_list, start=1):

    # Full path to the current input file

    input_file_path = os.path.join(genre_input_folder_path, file_name)

    # Loading the audio file
    audio = AudioSegment.from_file(input_file_path)

    # Define start and end times for the 30-second segment i.e first 30 seconds
    start_time = 0 * 1000
    end_time = 30 * 1000 # 30 seconds in milliseconds

    # Trimming the audio to the 30-second segment

    trimmed_audio = audio[start_time:end_time]

    # Setting properties to match GTZAN data set specifications

    trimmed_audio = trimmed_audio.set_frame_rate(22050) # Setting sample rate to 22050
    Hz
    trimmed_audio = trimmed_audio.set_channels(1) # Setting audio to mono

    # Defining the output file name based on genre and number

    output_file_name = f"{genre.lower()} {i}.wav"

    # Full path to the output file
    output_file_path = os.path.join(genre_output_folder_path, output_file_name)

    # Exporting the trimmed audio to a new file

```

```
trimmed_audio.export(output_file_path, format="wav", parameters=["-ar", "22050", "-ac", "1", "-sample_fmt", "s16"])

print(f"Processed file saved to: {output_file_path}")
```