Politecnico di Milano
School of Industrial and Information Engineering
M.Sc. in High Performance Computing

Final project:
Classify musical genre using audio files

Prof. Edie Miglio
A.Y. 2024-2025

Authors:
Peng Rao (ID 270661)

Contents

1 Introduction

In today's fast-paced world, with millions of new songs released daily, organizing music into genres is essential for improving user experiences on streaming platforms and managing music libraries efficiently. However, manually classifying songs by genre is a time-consuming task. This is where **machine learning** offers a powerful solution.

This project presents multiple **machine learning models** and **Convolutional Neural Networks** to classify music genres using audio files. The dataset used in this project is the **GTZAN dataset**, which contains 1000 audio tracks from 10 different genres. The models are trained on features extracted from the audio files, such as **chroma features**, **MFCCs**, and **spectral features**.

The project is structured as follows:

- **Data Exploration**: I explore the GTZAN dataset and visualize the audio features.
- **Data Preprocessing**: I preprocess the data by normalizing the features and splitting the dataset into training and testing sets.
- **Traditional Machine Learning Models**: I train several traditional machine learning models on the dataset and evaluate their performance.
- **Convolutional Neural Network (CNN)**: I train a CNN model on the dataset and evaluate its performance.
- **Conclusion**: I summarize the results of the project and discuss future work.

2 Data Exploration

2.1 Data Description

The **GTZAN dataset** is the most-used public dataset for evaluation in machine listening research for music genre recognition (MGR). The files were collected in 2000-2001 from a variety of sources including personal CDs, radio, microphone recordings, in order to represent a variety of recording conditions.[1]

The dataset consists of 1000 audio tracks each 30 seconds long. It contains 10 genres, each represented by 100 tracks. The tracks are all 22050Hz Mono 16-bit audio files in **.wav** format. The features of the dataset can be summarized as follows table 1.

Category	Feature	Description
General	filename, length, label	File metadata and classification
Spectral	chroma_stft, rms, spectral_centroid, rolloff, etc.	Frequency-based characteristics
Harmonic	harmony, perceptr, tempo	Harmonic and perceptual content
MFCCs	mfcc1-20 (mean, var)	Captures timbral properties of audio

Table 1: GTZAN Dataset Features

2.2 Waveform Analysis

The **waveform** of an audio signal represents the amplitude of the sound wave over time. Figure 1 visualizes the waveform of a sample track. Genres such as pop, metal, and hip-hop exhibit significant amplitude variations throughout the track, indicating dynamic changes in loudness. In contrast, classical and jazz show more structured amplitude patterns, reflecting their dynamic nature. Blues and reggae have distinct waveform structures that set them apart from other genres.

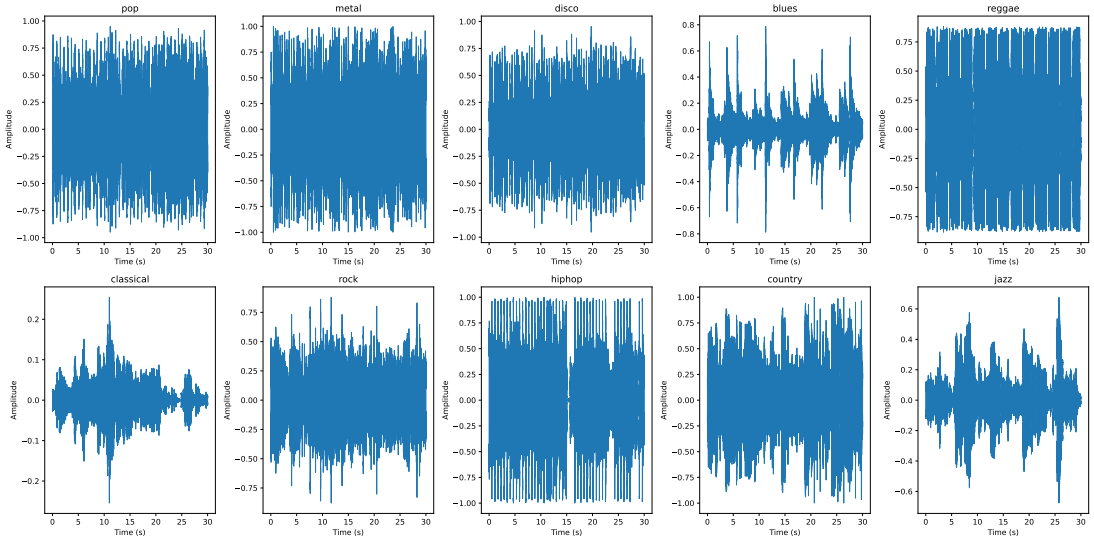

Figure 1: Waveform of various sample audios track over time

2.3 Chroma Analysis

Chroma features (or chroma vectors) are audio features used in music information retrieval to represent the harmonic and tonal content of an audio signal. They capture the distribution of energy across the 12 different pitch classes (e.g., C, C#, D, D#, ..., B) regardless of octave. Figure 2 visualizes the changes in chroma features over time for a sample track. Each row corresponds to one pitch class, showing the energy present in that pitch class over time. Genres like rock, blues, pop, and hip-hop exhibit dense chroma activity, indicating frequent chord changes. In contrast, classical and jazz display more structured patterns, with jazz showing greater harmonic complexity.

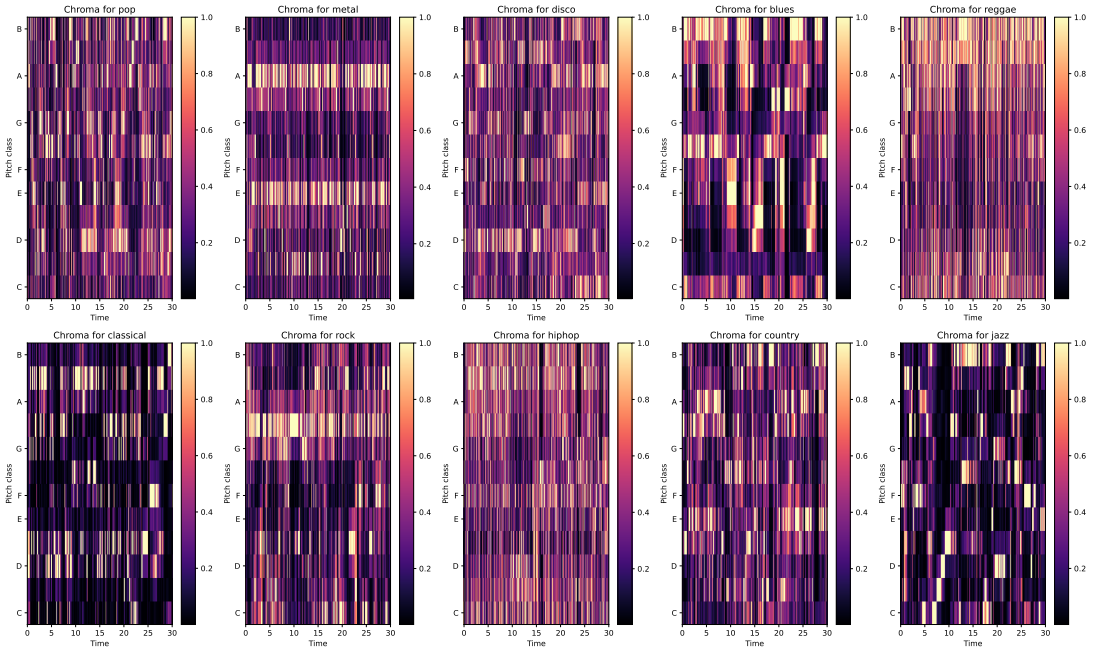

Figure 2: Chroma Features of various pitch classes over time

2.4 MFCC Analysis

MFCCs (Mel-Frequency Cepstral Coefficients) are widely used in audio and speech processing for feature extraction. They convert raw audio signals into a compact, perceptually meaningful representation by simulating the human auditory system. Typically, the first 13-20 coefficients are used as features. Figure 3 visualizes the MFCCs of a sample track. These coefficients capture the timbral properties of the audio signal, illustrating how the spectral content evolves over time. Genres such as rock, blues, and pop show high variability in MFCCs, reflecting their diverse timbral characteristics. In contrast, classical and jazz exhibit more structured patterns, indicating consistent timbral properties.

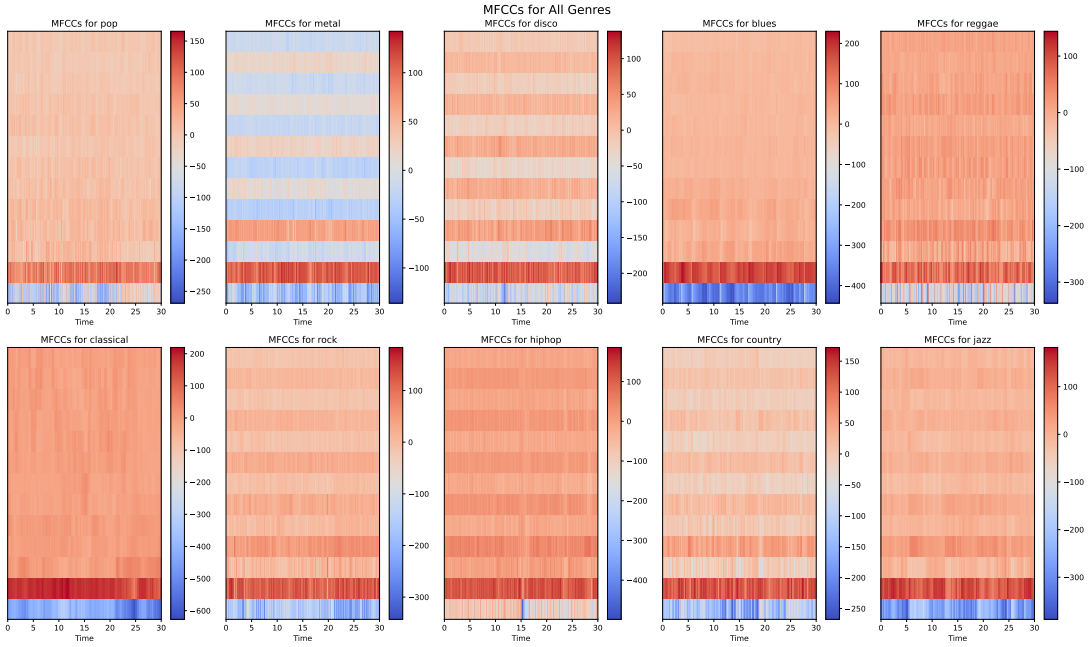

Figure 3: MFCC13 of various sample audios track over time

2.5 Mel Spectrogram Analysis

A **Mel Spectrogram** is a visual representation of the frequency content of an audio signal over time, mapped to the Mel scale. It approximates the human ear's response to different frequencies, emphasizing lower frequencies (where humans are more sensitive) and compressing higher frequencies. Figure 4 visualizes the Mel spectrogram of sample tracks. The spectrogram captures the frequency content of the audio signal over time, showing how the energy is distributed across different frequency bands. Genres like rock, blues, and pop exhibit dense spectrogram activity, indicating a wide range of frequency components. In contrast, classical and jazz show more structured patterns, with jazz displaying greater frequency complexity.

2.6 Principal Component Analysis

Principal Component Analysis (PCA) is a dimensionality reduction technique that transforms data into a lower-dimensional space while preserving as much variance as possible. I applied PCA to the dataset and visualized the first two principal components in figure 5.

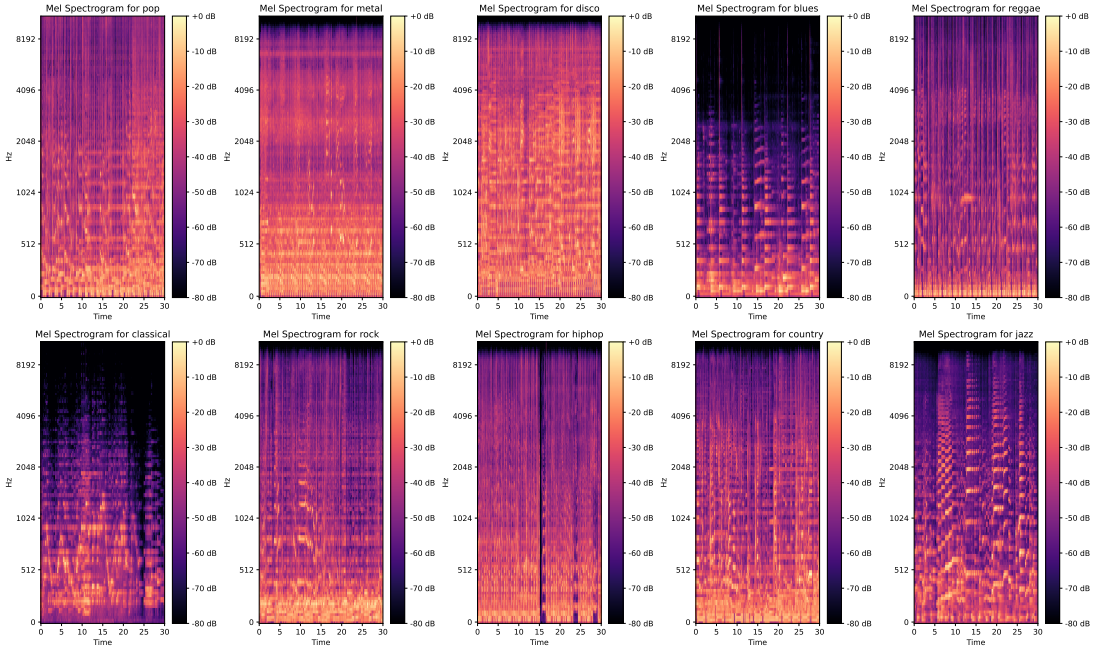

Figure 4: Mel Spectrogram of various sample audios track over time

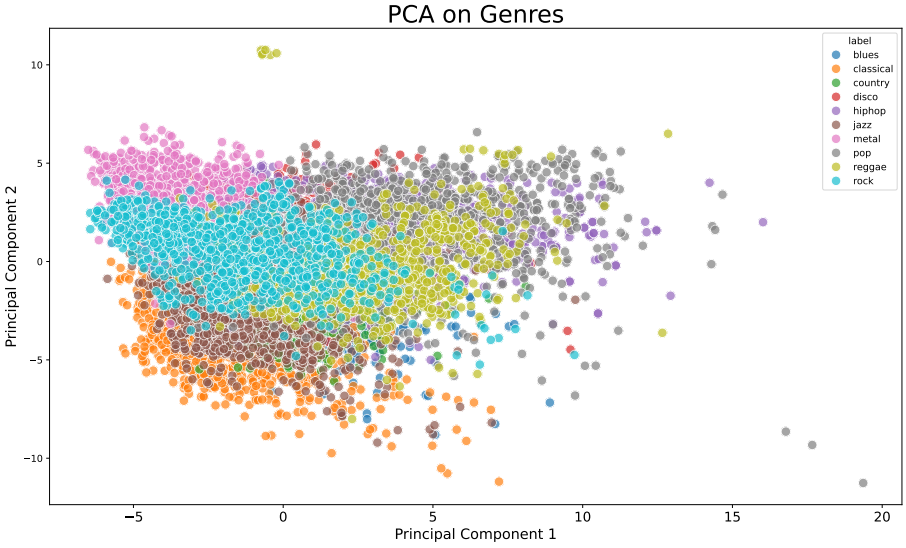

Figure 5: PCA of the GTZAN dataset

3 Data Preprocessing

3.1 Normalization

I normalized the features of the dataset to ensure that all features have the same scale. Normalization is essential for machine learning models that rely on distance-based metrics to prevent features with larger scales from dominating the model. I used min-max scaling to normalize the features. The formula for min-max scaling is:

$$X_{\text{scaled}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}} \tag{1}$$

3.2 Train/Test Split

I split the dataset into training and testing sets using an **8/2** split. I used the training set to train the model and the testing set to evaluate its performance. The validation has been used during the training phase of the classifers to find good values of the hyperparameters.

3.3 Cross-Validation Split

I used a 5-fold cross-validation split to evaluate the model's performance. Cross-validation is a technique used to assess the model's generalization performance by training and testing the model on different subsets of the data, as shown in figure 6.

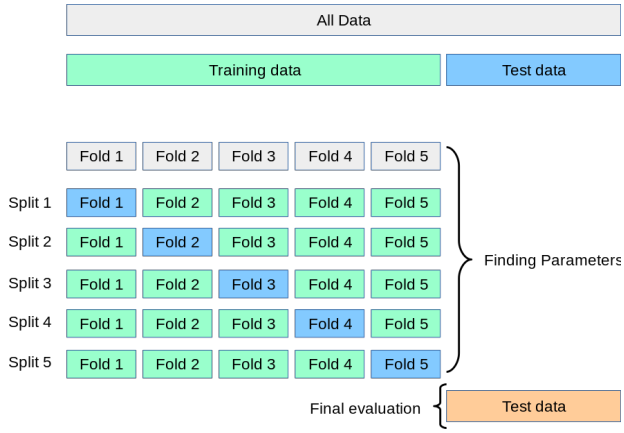

Figure 6: 5-fold Cross-Validation Split[2]

3.4 One-Hot Encoding

For CNN model, I used **one-hot encoding** to convert the target labels into a binary matrix. One-hot encoding is a technique used to convert categorical data into a binary matrix, where each column represents a unique category, and each row represents an instance. For example, the label 'rock' would be encoded as [0, 0, 0, 1, 0, 0, 0, 0, 0, 0] in a 10-class classification problem.

4 Traditional Machine Learning Models

4.1 Baseline Models

I trained several traditional machine learning models on the dataset to establish a baseline performance. The models I used are:

- **Logistic Regression**
- **Support Vector Machine**
- **Decision Tree**
- **Random Forest**
- **XGBoost**

The models were trained using the default hyperparameters and evaluated using 5-fold cross-validation. The results on the training set are summarized in figure 7, with the x-axis representing accuracy. The **XGBoost** model outperformed the other models, achieving an accuracy of **0.91** on the training set.

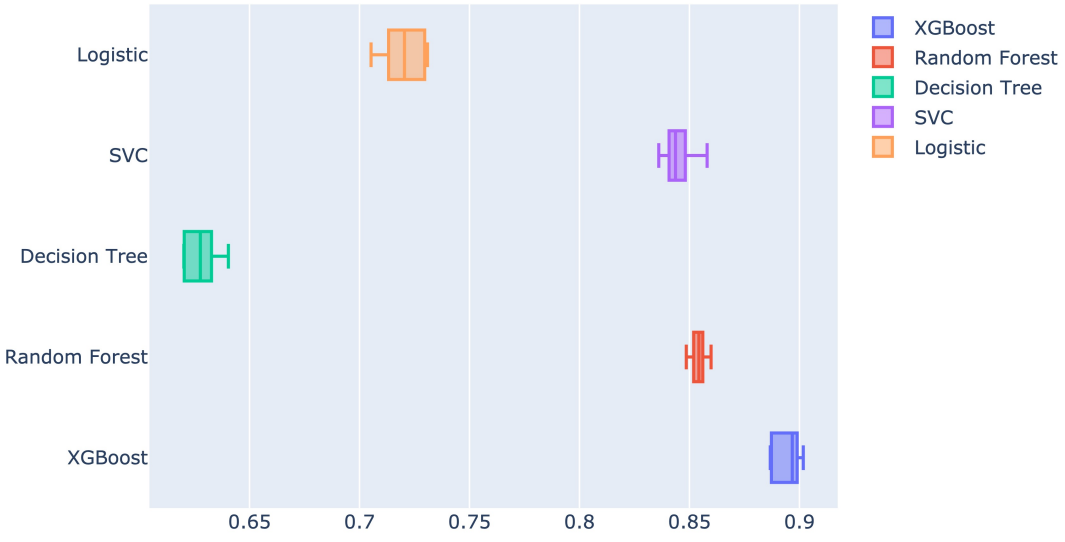

Figure 7: Baseline Models Performance on training set

4.2 Model Evaluation

I evaluated the XGBoost model on the testing set and obtained an accuracy of **0.90**. The confusion matrix in figure 8 shows that the model performs well for some genres (e.g., classical, metal) but struggles with others (e.g., rock, country).

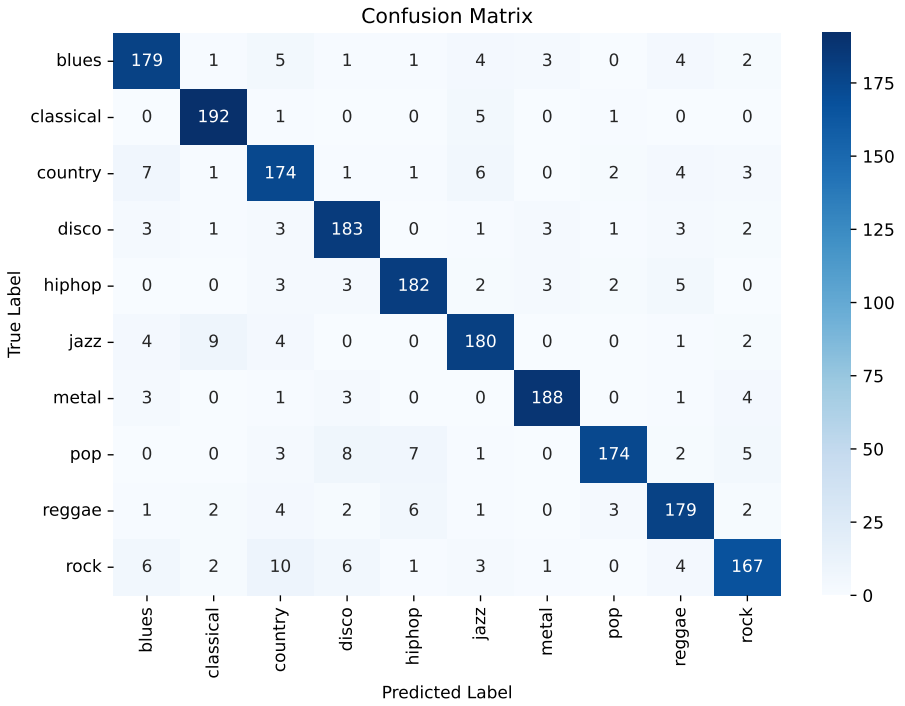

Figure 8: XGBoost Model Evaluation on testing set

To understand the importance of different features in the XGBoost model, I used the feature importance scores provided by the model. Feature importance scores indicate how valuable each feature is in constructing the boosted decision trees within the model. Figure 9 shows the feature importance scores for the top 10 features.

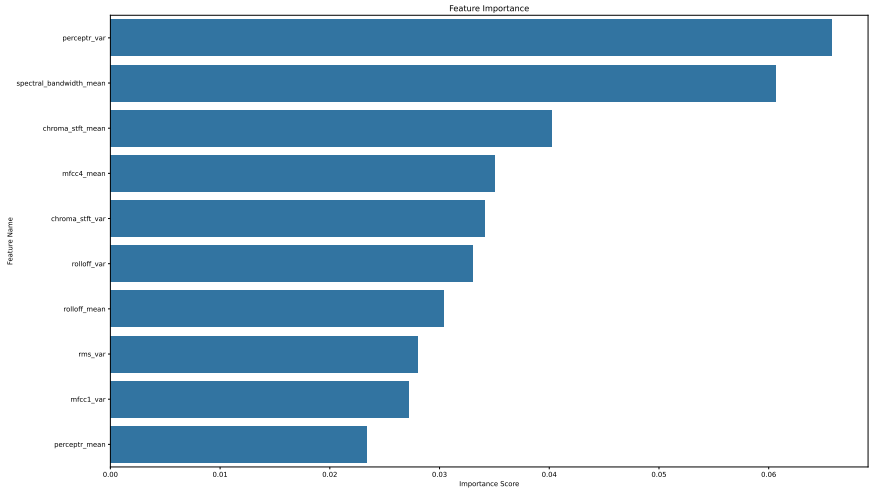

Figure 9: Feature Importance Scores for Top 10 Features

The most important features are **perceptr** and **spectral_bandwidth**, which capture the harmonic

and spectral content of the audio signals, respectively. The two features might capture characteristics that are more prominent in classical and metal music.

4.3 Hyperparameter Tuning

I used **RandomizedSearchCV** to tune the hyperparameters of the XGBoost model. I performed a randomized search over a predefined hyperparameter grid and selected the best hyperparameters based on the model's performance. The hyperparameters I tuned were:

- **max_depth**: Maximum depth of the tree
- **learning_rate**: Step size shrinkage used in update to prevent overfitting
- **n_estimators**: Number of boosting rounds
- **subsample**: Subsample ratio of the training instances
- **reg_alpha**: L1 regularization term on weights

The best hyperparameters found by RandomizedSearchCV were:

- **max_depth**: 5
- **learning_rate**: 0.2
- **n_estimators**: 500
- **subsample**: 0.8
- **reg_alpha**: 0.1

5 Convolutional Neural Network (CNN)

Deep learning enables music genre classification without relying on manually crafted features. In this study, audio signals are transformed into spectrograms, which are then treated as images for classification.

5.1 Spectrogram Generation

A **spectrogram** is a 2D representation of a signal, having time on the x-axis, **frequency** on the y-axis. Each audio signal was converted into a MEL spectrogram. The parameters used to generate the power spectrogram using STFT are listed below:

- **Window size**: 2048 samples
- **Hop length**: 512 samples
- **Sampling rate**: 22050 Hz

5.2 Cost Function

The cost function used in this study is the categorical cross-entropy loss function, which is commonly used in multi-class classification problems. The categorical cross-entropy loss function is defined as:

$$L(y, \hat{y}) = -\sum_{i} y_i \log(\hat{y}_i) \tag{2}$$

where y is the true label, $\hat{y}$ is the predicted label, and i is the class index.

5.3 Metrics

The model's performance was evaluated using the following metrics:

- **Accuracy**: The ratio of correctly classified instances to the total instances
- **Precision**: The ratio of true positive instances to the total predicted positive instances
- **Recall**: The ratio of true positive instances to the total actual positive instances
- **F1-score**: The harmonic mean of precision and recall

5.4 Model Architecture

Mel spectrograms can be treated as images and used as input to a CNN, which excels in image classification tasks. Each block in the CNN consists of the following layers:

- **Convolutional 2D layer**: Extracts features from the input image using filters. Each filter slides over the image and performs a convolution operation, producing a feature map that highlights different aspects of the image, such as edges, textures, or patterns.
- **Max pooling layer**: Reduces the spatial dimensions of the feature map by sliding a window over the feature map and taking the maximum value within the window. This operation helps in reducing the computational load and makes the features more robust to spatial variations.
- **Dropout layer**: Regularizes the model to prevent overfitting by randomly setting a fraction of the input units to zero at each update during training. This encourages the model to learn more robust features.

The architecture of the CNN model [3] used in this study is shown in figure 10.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 150, 150, 32)	320
conv2d_1 (Conv2D)	(None, 148, 148, 32)	9,248
max_pooling2d (MaxPooling2D)	(None, 74, 74, 32)	0
conv2d_2 (Conv2D)	(None, 72, 72, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 36, 36, 64)	0
conv2d_3 (Conv2D)	(None, 34, 34, 128)	73,856
max_pooling2d_2 (MaxPooling2D)	(None, 17, 17, 128)	0
dropout (Dropout)	(None, 17, 17, 128)	0
conv2d_4 (Conv2D)	(None, 15, 15, 256)	295,168
max_pooling2d_3 (MaxPooling2D)	(None, 7, 7, 256)	0
dropout_1 (Dropout)	(None, 7, 7, 256)	0
conv2d_5 (Conv2D)	(None, 5, 5, 512)	1,180,160
max_pooling2d_4 (MaxPooling2D)	(None, 2, 2, 512)	0
dropout_2 (Dropout)	(None, 2, 2, 512)	0
flatten (Flatten)	(None, 2048)	0
dense (Dense)	(None, 1200)	2,458,800
dropout_3 (Dropout)	(None, 1200)	0
dense_1 (Dense)	(None, 10)	12,010

Figure 10: CNN Model Architecture

5.5 Model Training

The CNN model was trained using the training set and validated using the validation set. The model was trained for 50 epochs with a batch size of 32. The model was compiled using the **Adam optimizer**(with learning rate = 0.0001) and the categorical cross-entropy loss function.

The training phase results are depicted in figure 11, showing the loss and accuracy curves. The figure illustrates a decreasing loss and increasing accuracy over time, indicating effective learning. However, as the number of epochs increases, there is a risk of **overfitting**.

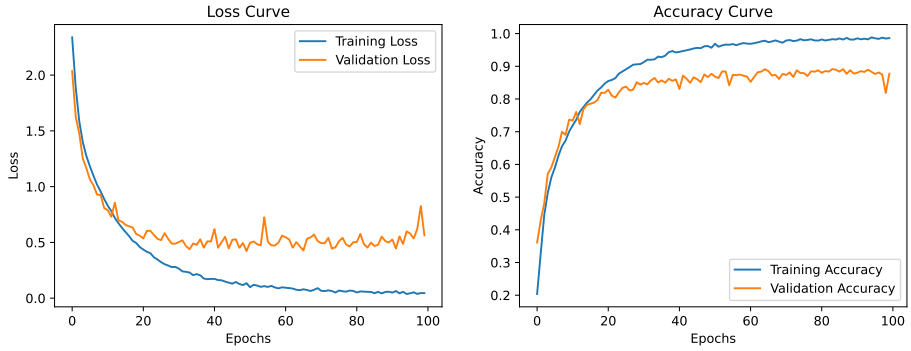

Figure 11: Loss and Accuracy Curves during training

5.6 Model Evaluation

The CNN model was trained on the training set and evaluated on the testing set. It achieved an accuracy of **0.87** on the testing set. The confusion matrix in figure 12 indicates that the CNN model performs well for most genres. However, it encounters difficulties with genres like jazz and reggae, which exhibit similar spectral characteristics.

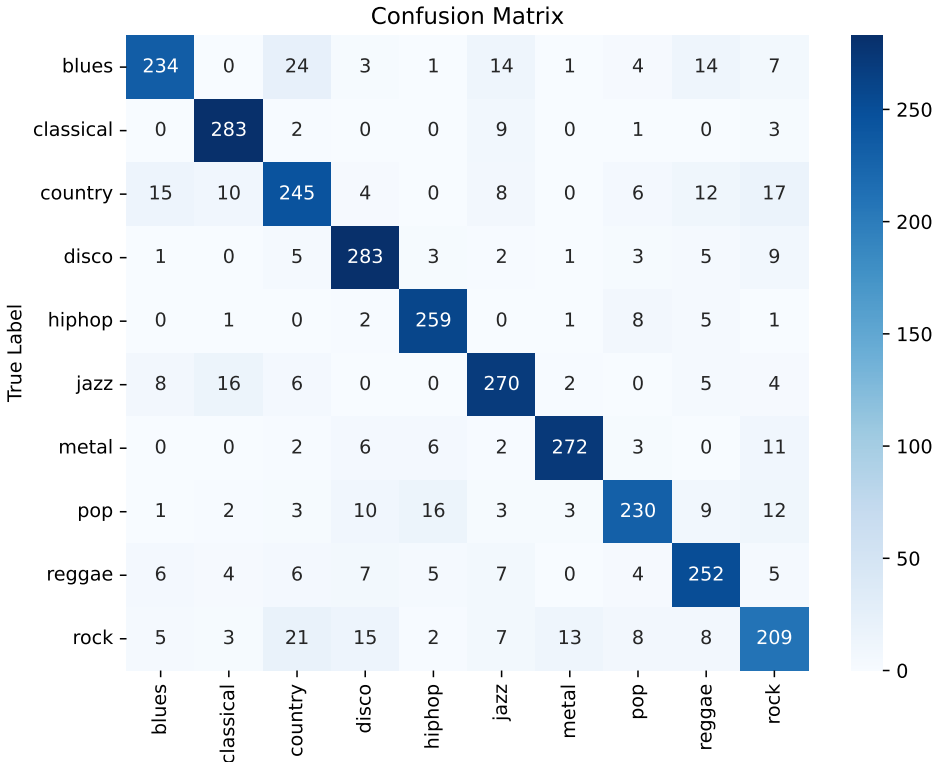

Figure 12: CNN Model Evaluation on testing set

6 Conclusion

In this study, I explored the GTZAN dataset, preprocessed the data, and trained both traditional machine learning models and a CNN model for music genre classification. The CNN model underperformed compared to traditional models because CNNs are optimized for spatial data, and when applied to extracted audio features that lack spatial relationships, they struggle. On the other hand, traditional models like SVC and XGBoost utilize manually extracted features (such as MFCC, Chroma, Spectral Contrast, Zero-Crossing Rate, etc.) that are specifically designed to capture audio patterns effectively. SVC and XGBoost classify based on these high-dimensional features efficiently and are less prone to overfitting.

Based on the performance table 2, it is evident that **XGBoost** is the most suitable model for this classification task.

Model	**Accuracy**	**Precision**	**Recall**	**F1score**
SVC	0.85	0.85	0.85	0.85
Random Forest	0.87	0.87	0.87	0.87
CNN	0.87	0.87	0.87	0.87
XGBoost	0.91	0.91	0.91	0.91

Table 2: Model Performance on testing set

References

[1] *GTZAN Dataset*. URL: https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification.

[2] *Cross-validation: Evaluating Estimator Performance*. scikit-learn. URL: https://scikit-learn.org/stable/modules/cross_validation.html.

[3] A. Ghildiyal, K. Singh, and S. Sharma. "Music Genre Classification Using Machine Learning". In: 2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA). Coimbatore, India: IEEE, Nov. 5, 2020, pp. 1368–1372.