# MUSIC GENRE CLASSIFICATION: THE REPRODUCTION AND COMPARISON OF TRADITIONAL METHODS AND CNN

GROUP 5

**Cheng Qizhe**
2021533139
chengqzh@shanghaitech.edu.cn

**Gan Hangyi**
2021533153
ganhy@shanghaitech.edu.cn

**He Jiacheng**
2021533170
hejc@shanghaitech.edu.cn

**Zhang Yufei**
2021533169
zhangyf@shanghaitech.edu.cn

January 2024

**ABSTRACT**

Music genre classification is a hot topic in the field of artificial intelligence, and there have been mature algorithms and models developed for this purpose. In order to gain a deeper understanding of various methods and their effectiveness in music genre classification, we aim to start with traditional approaches and reproduce some of the conventional methods used in this area. For example, we will explore classification using methods such as K-nearest neighbors (KNN), decision trees, and support vector machines (SVM) to classify music genres. Additionally, we will further investigate modern methods and analyze their advantages and disadvantages. This report will showcase our learning process and exploration in this field.

***Keywords*** Music genre classification · k-nearest neighbor · decision trees · support vector machines · convolutional Neural networks.

## 1 Introduction

Music genre classification is the task of automatically identifying the genre of a given audio signal. It has many applications in the music industry, including recommendation systems, personalized playlists, and content-based music retrieval. Traditional methods for music genre classification include feature extraction and classification using algorithms k-nearest neighbor, decision trees, and support vector machines are a few examples. For modern methods, deep learning models such as convolutional Neural networks (CNNS), Recurrent neural networks (RNN), and Long Short-Term Memory networks (LSTM) perform well in music genre classification. These models are capable of processing more complex timing information and high dimensional data, adapting to a wider range of musical characteristics.Next, we will introduce our work from the following aspects:

- Data Collection: we choose the GTZAN dataset to get the data.
- Feature Extraction: We have chosen the Mel spectrogram and MFCC methods to extract features from the music data for subsequent classification.
- Implementation of Traditional Methods: We have implemented some traditional machine learning methods such as K-nearest neighbors (KNN), decision trees, and support vector machines (SVM) for music genre classification.
- Exploration of Modern Methods:We have further explored modern approaches for music genre classification such as convolutional Neural networks(CNNS).

- Result Analysis and Comparison: We have analyzed and compared the results obtained from different methods, evaluating their performance in music genre classification tasks.

## 2 Dataset

To facilitate learning and reduce distractions, we chose to process music without human voices.The dataset we choose is GTZAN.

GTZAN dataset is a publicly available dataset widely used for music classification and music information retrieval tasks. The dataset consists of 1000 audio files covering 10 different music genres, with each genre containing 100 tracks. The genres include blues, classical, country, disco, hip-hop, jazz, metal, pop, reggae, and rock. Each music track has a duration of 30 seconds(in WAV format) and a sampling rate of 22050 Hz. In addition, the dataset includes raw spectrum images that have been processed and features in 30 second and 3 second csv formats.

## 3 Feature Extraction

### 3.1 Mel spectrogram

The spectrum provides information about the frequency composition and intensity of a signal. It can be used to analyze the frequency characteristics of audio signals, and extract the features for machine learning models. However, the general spectrum is a linear scale, while human perception of sound frequencies is nonlinear. In the lower frequency range, the human ear has a higher resolution of frequency, but the resolution gradually decreases as the frequency increases. Therefore, the linear-scale spectrum cannot simulate human perception of sound well, causing classification bias. Therefore, we need a nonlinear-scale spectrum to simulate human perception of frequency.

The Mel spectrogram is a commonly used feature representation method in audio signal processing. It simulates the way human ears perceive sound frequencies by using the Mel scale(nonlinear scale).

To extract feature from the audio data,we should first divide the continuous audio signal into shorter time segments called frames. Each frame typically contains a few tens of milliseconds of audio data.Then we need apply the Fourier transform to each frame to convert the audio signal from the time domain to the frequency domain representation.

For the data form we choose power spectrum to represent the music feature. we can calculate the energy or power of each frequency component from the frequency domain representation. The power spectrum can be obtained by squaring the amplitude of each frequency component.

To simulate human perception well, we need apply a bank of Mel filters that a bank of Mel filters consists of a set of triangular filters with center frequencies arranged according to the Mel scale. The Mel scale is a non-linear scale based on the perceptual nature of frequency. Filter the power spectrum through each filter to obtain the output of each filter.

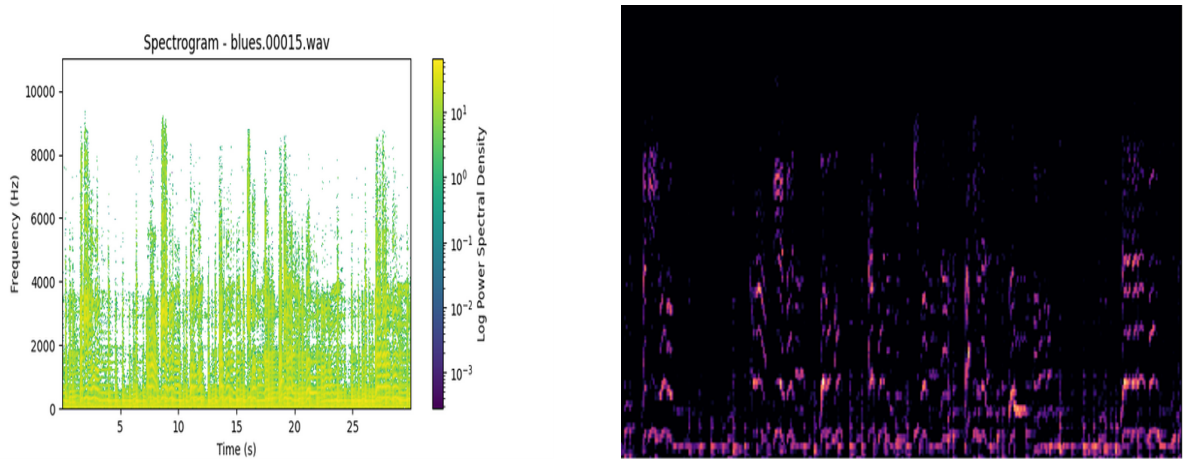
After applying filters,we apply a logarithm operation to the output of each filter to reduce the dynamic range and better represent the features of the audio signal.Then combine the outputs of the filters after the logarithm operation to obtain the Mel spectrogram.

### 3.2 MFCC

Mel-Frequency Cepstral Coefficients (MFCC) is a commonly used audio feature extraction method.MFCC and Mel-frequency spectrum share some common steps in signal transformation, including Fourier transform, Mel filters, and logarithmic operations. However, the difference lies in the fact that after performing the logarithmic operation, MFCC applies Discrete Cosine Transform (DCT) to the energy value sequence and extracts feature coefficients from it. Compared to the Mel-frequency spectrum, MFCC feature coefficients result in more concise feature extraction. Additionally, due to their inclusion of spectral feature information, they also make the data more representative.

### 3.3 K-nearest neighbors (KNN)

The K-Nearest Neighbors (KNN) algorithm is a commonly used supervised learning algorithm, mainly used for classification problems, but can also be used for regression problems. The KNN algorithm classifies based on the similarity between instances, i.e. it assigns a new sample to the class of the K most similar training samples. In our implementation, we first load the data from the pre-processed dataset and divide it into training and testing sets. Then, we save the labels corresponding to the training samples for classification. During the prediction stage, for a new testing sample, we calculate the distance between it and each training sample. Here, we use the Mahalanobis distance to




(a) MFCC (b) GTZAN

Figure 1: spectrum of MFCC and GTZAN dataset

measure the distance between samples:

$$D_M(x, y) = \sqrt{(x - y)^T \Sigma^{-1} (x - y)}$$

Where x and y are the data samples,$\Sigma$ represents the covariance matrix of a multi-dimensional random variable.The Mahalanobis distance considers the correlation between multiple dimensional data and can better reflect the actual differences between samples.

After measuring the distances, we select the k nearest training samples to the test sample. We then perform a statistical analysis on the labels of these k training samples and choose the label that appears most frequently as the predicted result.

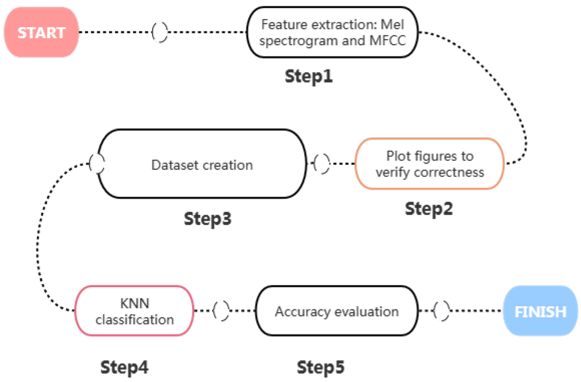



Figure 2: Pipeline of KNN

### 3.4 Support Vector Machine (SVM)

Support Vector Machine (SVM) is a binary classification model that achieves classification by finding an optimal hyperplane in the feature space. The core idea of SVM is to find a hyperplane that maximizes the margin between classes. In a high-dimensional space, the hyperplane is a subspace of dimension (n-1), where n is the dimensionality of the features. Support vectors are the points closest to the hyperplane, and they determine the position and orientation of the hyperplane.

In our implementation, we first extract features from audio data, then preprocess them to obtain features and corresponding labels, which are transformed into feature matrices and label vectors. The preprocessed dataset is then divided into training and testing sets.

During the training process, we initialize the weights and biases. We then use stochastic gradient descent to iteratively update them until the desired number of iterations is reached. In each iteration, we iterate through the training samples and update the weights and biases based on the comparison between the predicted results and the true labels. If the prediction is correct, only the weights are updated. If the prediction is incorrect, both the weights and biases are updated, and the total loss is calculated. Here, we use the hinge loss as the loss function:

$$L_{y\cdot(x\cdot w+b)} = [1 - y(x \cdot w + b)]_+$$

$$[z]_+ = \begin{cases} z, & z > 0 \\ 0, & z \leq 0 \end{cases} \tag{1}$$

The characteristic of the hinge loss function is that when there is a large difference between the predicted result and the true label, the value of the loss function increases, thereby encouraging the model to better classify the samples. At the same time, the hinge loss function imposes a greater penalty on support vectors that are far from the hyperplane, which plays a crucial role in determining the decision boundary.

### 3.5 Decision tree

The decision tree algorithm is a machine learning algorithm based on tree structure. It recursively divides the dataset to construct a tree for prediction and decision-making. The decision tree algorithm is suitable for both classification and regression problems.

In the decision tree, each internal node represents a feature or attribute, and each leaf node represents a category or value. Starting from the root node, the dataset is divided into different subsets based on different feature attributes, and then the best feature is selected for further division in each subset until the stop condition is reached. In our implementation, we first preprocess the audio data the same way as other algorithms, and divide the data into training set and test set to obtain the feature matrix and label vector. Then we pass the data into the constructed decision tree and recursively split it from the root node. At each recursion, we first determine whether to stop the recursion. If yes, we return the leaf node; otherwise, we traverse all features, sort them according to feature values, and obtain all possible split points.

For all possible split points, we calculate the corresponding Gini index and return the best split point. After the split, we repeat the process on the subsets and return a decision node. For test set samples, we traverse the decision tree starting from the root node until reaching a leaf node, and obtain the predicted result for the sample.

The Gini index is a metric used to evaluate the purity of a sample set in classification algorithms. In the context of the decision tree algorithm, it is used to select the optimal splitting feature:

$$Gini(p) = \sum_{k=1}^{K} p_k(1 - p_k) = 1 - \sum_{k=1}^{K} p_k^2$$

Here, $p_k$represents the probability of a sample belonging to class k, and K is the number of classes.

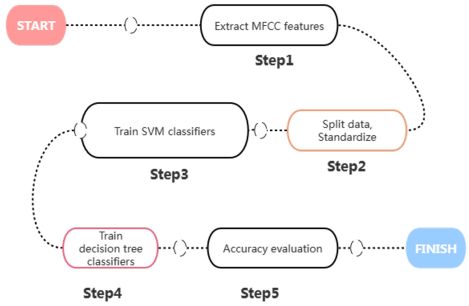



Figure 3: Pipeline of Decision Tree

### 3.6 Naive Bayes Classifier

The Naive Bayes Classifier is a classification algorithm based on probability statistics and the assumption of feature independence. Its "naive" aspect lies in assuming that the features are mutually independent, meaning that given the class, the features are conditionally independent. The Naive Bayes Classifier is based on Bayes' theorem and performs classification by calculating posterior probabilities.

During the training phase, it learns the prior probabilities of each class and the conditional probability distributions of each feature within each class. It preprocesses the dataset and calculates the unique class labels and their frequencies using numpy library functions. It also calculates the prior probabilities for each class. Then, for each class, it computes the mean and variance of the features.

During the prediction phase, it uses the learned probability model to calculate the posterior probabilities of each class given the observed features, and selects the class with the highest posterior probability as the predicted result. In our implementation, we also preprocess the dataset. For the training set, we utilize numpy library functions to obtain the different class labels and their frequencies, and calculate the prior probabilities for each class. Then, for each class, we calculate the mean and variance of its features.

During the prediction phase, for each sample in the test set, we calculate the log likelihood of the observed features given each class, add the log prior probability, and obtain the scores for each class. Finally, the predicted class is the one with the highest score.

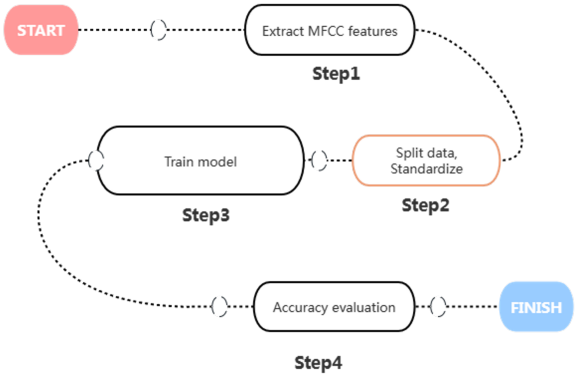



Figure 4: Pipeline of Naive Bayes Classifier

## 4 Exploration of Modern Methods

### 4.1 Convolutional Neural network

Convolutional Neural Network(CNN) is a type of deep learning model that extracts features by applying convolution operations on input data. It consists of multiple convolution layers, pooling layers, and fully connected layers.

The basic CNN has three convolutional layers with ReLU activation function and max pooling layers and three fully connected layers. The first part includes two convolutional layers (conv1 and conv2), both of which have a 3x3 kernel, a stride of 1, and padding of 1. The first convolutional layer, conv1, takes an input with 3 channels and produces 16 output channels, while the second, conv2, produces the 32 channels and the third, conv3, produces the 64 channels. A 2x2 max-pooling layer (pool) that decreases the spatial dimensions. Lastly, the part consists of fully connected (FC) layers (fc1, fc2, and fc3) that transform the high-level features extracted by the convolutional layers into final class scores. These FC layers are interspersed with dropout layers that have a dropout rate of 0.5. The model ends with an output layer (fc3) that produces 10 outputs, presumably corresponding to 10 classes.

After completing the pre-processing of the data,the dataset was divided into two sets: training and testing, with $80\%$ of the data used for training and $20\%$ for testing. The SGD optimizer was used to train the model, with a learning rate of 0.001, momentum of 0.9. And early stopping was used when the model was trained to avoid overfitting, The dropout CNN is based on the basic CNN and add two dropout layer among the fully connected layers with dropout of 0.5.

VGG16 is finally used to classify the music genres. The images was first convert into 224x224 pixel images to adapt to the VGG16. VGG16 has a high amount of parameters, approximately 138 million. Due to its simplicity and depth, VGG16 has become a popular choice in the community and has been adapted and modified for various tasks beyond just image classification. The model is known for its simplicity, utilizing only 3x3 convolutional layers stacked on top of each other in increasing depth. Deeper layers in the network have more filters. After several convolutional layers, a max-pooling layer is used to reduce the spatial dimensions. There are three fully linked layers after these convolutional blocks. The first two layers each contain 4096 channels, whereas layer three has 1000 channels, or 1000 classes in the ImageNet dataset. ReLU activation function is used after each convolutional and fully connected layer. It's used to produce a probability distribution over the 1000 classes. This project has been modified in the final fully connected layers to accommodate the number of categories in the dataset. These layers output a score for each of the 10 image categories instead of the original 1000 categories.

The codes are based on the pytorch to load the data , divide the data set, build the CNN, and train the model. And finally got an accuracy of the test data set.

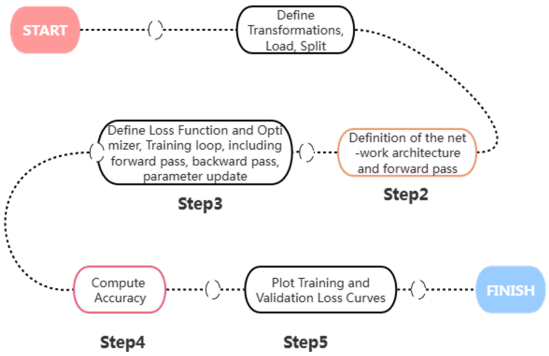



Figure 5: Pipeline of CNN

## 5 Result Analysis and Comparison

### 5.1 K-nearest neighbors (KNN)

Due to the scarcity of the labeled data set, we chose to use the GTZAN data set as both the training set and the test set. Like our reference website, the KNN algorithm for traditional music genre analysis generally extracts features through MFCC. However, since the GTZAN dataset we selected is a dataset without voice (MFCC is more sensitive to voice feature extraction while mel spectrum retains more audio features), two feature extraction methods (mel spectrum and MFCC) are considered to be used simultaneously to predict the category of data in the test set.(Use dat to storage features, distinct from csv for given data sets) In addition, the GTZAN dataset has a small number of samples, which makes our KNN method have a performance on this dataset that is similar to that of convolutional neural network – the KNN method using the two feature extraction has an accuracy of 73.25%.

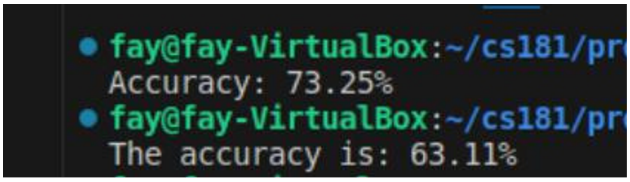



Figure 6: Result of KNN

### 5.2 Support Vector Machine (SVM)

The performance of SVM depends on the size and content of the dataset, feature extraction, and parameter tuning. Due to the small size of the GTZAN dataset, the training set is not large enough, resulting in overfitting and a decrease in

classification accuracy. Additionally, the unbalanced distribution of training samples among different music genres causes the hyperplane to be biased towards labels with larger sample sizes, resulting in an unreasonable division. Although the data features extracted by MFCC are representative, they are not comprehensive enough, resulting in ambiguous distinctions between different samples or similar features in some dimensions. The SVM algorithm is sensitive to feature selection, resulting in poor classification performance. By tuning different parameters such as kernel functions and penalty factors, we achieved an accuracy rate of approximately $55\%$ using the SVM algorithm.

### 5.3 Decision Tree

The advantages of the decision tree algorithm are that it provides an intuitive classification and has a relatively fast training speed, with fewer restrictions on the data content. The effectiveness of the decision tree depends on the control of the tree depth and whether the data contains exceptional values. However, due to the complexity of the classification process and the large number of categories, decision trees with deeper depths and more branches may result in overfitting during the training process. Furthermore, since the data features themselves have different dimensions, and the structure of the decision tree is relatively simple, the minor changes in the data can result in completely different decision processes, leading to poor performance on the test set. In addition, the structure of the decision tree ignores the correlation between features, which can cause multi-dimensional features to not only fail to enhance classification but also disrupt it. Finally, after attempting different parameters, the decision tree algorithm achieved an accuracy rate of about $40\%$.

### 5.4 Naive Bayes Classifier

The Naive Bayes classifier, as a structurally simple classification algorithm, has fast training speed and strong robustness, making it suitable for various types of datasets and accomplishing classification tasks. However, its assumption of independence between features is not well-suited for music features obtained through MFCC transformation, completely ignoring the correlations and influences among features, resulting in a deviation from reality in its classification performance. Additionally, the imbalanced dataset, insufficient training samples, and excessive reliance on the training set contribute to the poor generalization ability of the Naive Bayes classifier when the test set has a different data distribution than the training set, leading to significant deviations between predicted and actual classification results. Based on the GTZAN dataset training, we achieved an accuracy of around $45\%$ using the Naive Bayes classifier.

### 5.5 Convolutional Neural network

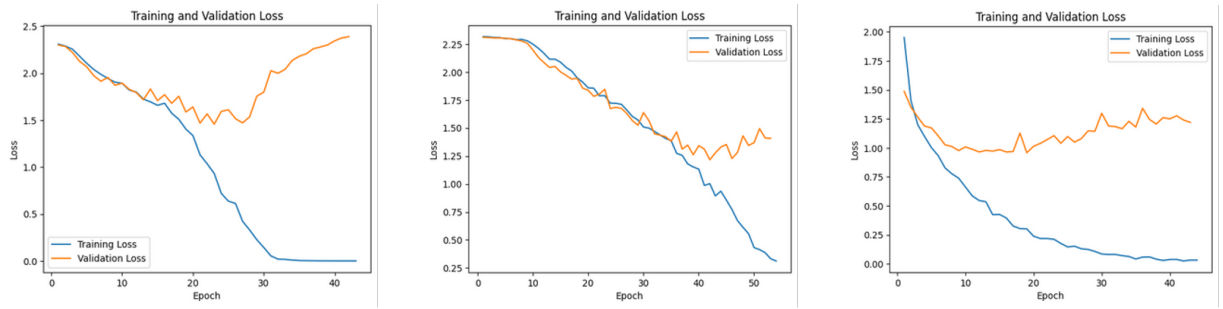
CNN, as an important algorithm in deep learning today, is widely used in various machine learning tasks. Because CNN can automatically extract features, and its convolutional layer parameters can be shared, it reduces the bias in feature extraction and the risk of overfitting during the training process. Additionally, local connections and pooling enable the extraction of local features from the data, simplifying the computational complexity of the model.

Due to the strong adaptability of convolutional neural networks, the CNN algorithm can be improved through various methods. For example, in the training process, we can add dropout operations, which appropriately weaken the dependencies between neurons, making the trained model more likely to generalize well and improve its accuracy on the test set.

VGG16 is a widely used convolutional neural network that has achieved excellent results in classification competitions. It captures features in the data more effectively by using smaller convolution kernels, max pooling, multiple repeated convolutional blocks, and deeper layers. This results in a more accurate and generalized model. However, due to the strong dependency of CNN on data, and the limited complexity and adequacy of the GTZAN dataset we used, the deeper layers of the convolutional neural network did not have a significant impact on the results, and the model's accuracy did not reach the theoretical optimum. Therefore, our convolutional neural network performed poorly on the GTZAN dataset: the basic CNN achieved an accuracy of $56\%$, the accuracy improved to $60.5\%$ with the addition of dropout, and VGG16 achieved an accuracy of $73\%$.

## 6 Conclusion

In our project, we explored two methods for extracting features from audio data: mel spectrogram and MFCC. We reproduced some traditional methods for music genre classification and also implemented modern approaches. Traditional methods often suffer from limitations such as simple structure or lack of flexibility in the models. This means that they do not perform well when the training data is limited or imbalanced. Traditional methods like decision trees and naive Bayes classifiers did not yield satisfactory results in such cases. However, the KNN algorithm showed




(a) Basic CNN (b) Basic CNN+Drop out (c) VGG16

Figure 7: Training process:Basic CNN,Basic CNN+Drop out, VGG16


(a) Basic CNN (b) Basic CNN+Drop out (c) VGG16

Figure 8: Result:Basic CNN,Basic CNN+Drop out, VGG16

significantly better results than other traditional methods when applied to feature extraction using MFCC and mel spectrogram, thanks to its comprehensive and representative features. Nevertheless, due to its low model complexity, it was difficult to achieve significant improvements with the KNN algorithm.

In the modern approaches, we implemented the CNN algorithm and improved its performance by increasing the depth of the neural network, adding dropout layers, using smaller pooling layers and convolution kernels. These enhancements made the model more stable and improved its generalization, leading to good results. However, due to the limited size and complexity of the training dataset, the optimization of deep neural networks could not be fully demonstrated, resulting in similar results to the improved KNN algorithm. By using larger and more complex datasets, deepening the neural network, and employing other techniques, we can further leverage the performance of convolutional neural networks, optimize the classification model, and achieve higher accuracy and better classification results.

## References

Reference website:
1.https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification
2.https://www.kaggle.com/code/troymazerolle/using-vgg16-to-classify-spectrograms
3.https://data-flair.training/blogs/python-project-music-genre-classification
4.A. Karatana and O. Yildiz, "Music genre classification with machine learning techniques," 2017 25th Signal Processing and Communications Applications Conference (SIU), Antalya, Turkey, 2017, pp. 1-4, doi: 10.1109/SIU.2017.7960694.
5.https://scholarworks.calstate.edu/concern/theses/j6731b078

External library:
1.[python_speech_features: python_speech_features.mfcc] - To extract MFCC (Mel-frequency cepstral coefficients) features from audio signals.
2.[scipy.io.wavfile: wav.read] - To read the audio signals and their sampling rate from WAV files.
3.[scipy.stats: multivariate_normal] - To multivariate normal distributions.
4.[numpy] - To compute the covariance matrix, mean(Do numerical operations)
5.[os] - To check for file existence, create directories, and join file paths.
6.[pickle] - To save the extracted MFCC features into a binary file (mfcc_dataset.dat).
7.[matplotlib] - To generate spectrogram plots.
8.[sklearn] - Data preprocessing, model evaluation, and metrix calculation.
9:[torch] - To build and train neural networks.
10:[torchvision: transforms] - To convert the images to tensors and normalize them.
11:[torch.utils.data: DataLoader] - To load data in batches during training.
12:[torchvision.datasets: ImageFolder] - To load the image data.