PROJECT REPORT FOR DT2470 MUSIC INFORMATICS

Hanqi Yang
TINNM
hanqi@kth.se

Qian Zhou
TINNM
qianzho@kth.se

1. INTRODUCTION

The combination of pitch, timbre and other components of music has given rise to different musical genres. And with such a large music library, both streaming services and users need to tag genres in order to manage their preferred content. As a result, a need for a classification of musical genres emerged.

It is a quite difficult problem because the boundaries between different genres could be fuzzy in nature. Even if the human ear of college students has only about 70% correct rate in classifying musical genres [1]. In recent years, there has been a great deal of research on music genre classification (MGC) using machine learning (ML). In general, the main application direction of traditional neural network architectures such as Convolutional Neural Network (CNN) is image classification, but some studies have demonstrated that using CNN also has good classification results for audio features such as Mel-Spectrum, MFCC, etc. [2]. Hamel *et al* [3] used Support Vector Machine (SVM) based on the output of a neural network for MGC task. T.R. [4] and J.A. *et al* [5], on the other hand, used Gaussian Mixture Model (GMM) as the model of MGC.

In this project, we will build a system to recognize the genre of a specific song snippet. Given a segment of a song, the task is to match the genre of the song to one of six classes. We will extract 10 features in total of each audio file and concatenate them as a music feature image. And then we will use 3 different ML models, CNN, SVM and GMM, and compare their results to find a better model.

2. PROBLEM DESCRIPTION

In this part, we will talk about the dataset and the environment in Section 2.1. As the pipeline of feature extraction shown in Figure 1, the principles and parameter configuration of the 10 features will be introduced in Section 2.2. After extracting the features needed, the procedure of data preprocessing and the model construction will be introduced in Section 2.4 and Section 2.3 respectively.

2.1 Data set & Environment

Our dataset is ISMIR2004Genre, which contains more than 2000 full-song files from 6 different genres or mixed genres: classical, electronic, jazz-blue, metal-punk, rock-pop and world. But the music files in the original dataset vary too much in length, ranging from 6 seconds to 20 minutes. In order to facilitate the feature extraction later, we

.

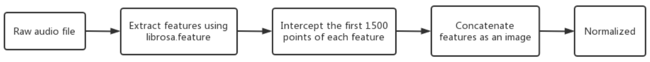

Raw audio file
Extract features using librosa.feature
Intercept the first 1500 points of each feature
Concatenate features as an image
Normalized

Figure 1: Pipeline of feature extraction

remove the songs that are too short and too long. Finally, we obtain a dataset containing 1742 tracks ranging from 30 seconds to 6 minutes in length. The dataset is introduced in [6]. The number of samples of each genre are shown below.

- classical: 731 samples
- electronic: 261 samples
- jazz_blues: 66 samples
- metal_punk: 124 samples
- rock_pop: 291 samples
- world: 269 samples

The whole project is based on *Python 3.9*. To speed up and simplify the code, we use the functions in *librosa.feature*, which is a strong library for audio processing, to extract the features. CNN model is constructed by *TensorFlow*, a software library for numerical computation using data flow graphs. It is widely used in the region of ML and CNN research. SVM and GMM model use the model from *sklearn* library, which can cover most of the mainstream ML algorithms.

2.2 Feature extraction

Raw audio samples cannot be analyzed, so feature extraction is one of the most important steps. We select 10 features that are representative of audio analysis, and this section will introduce and analyze them in detail. The length of each feature is related to the length of the song. As our dataset is based on whole songs as samples, this means that each song has a different length of features extracted. To ensure that the features are of the same length, we extracted the first 1500 points of each song feature, which is approximately the first 30 seconds of the song. Finally, we concatenate these 10 features together to make a feature image of size 81×1500 as shown in Figure 2.

2.2.1 Mel-Spectrogram

As the basic representation of audio feature, Mel-Spectrogram is one of the most widely used one in Music

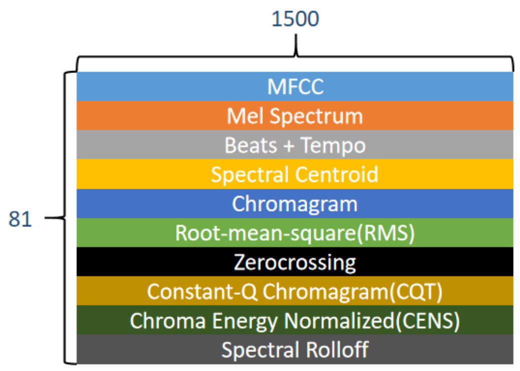


Figure 2: The feature construction as an image

Information Retrieval (MIR). Since the usual spectrogram has its frequencies distributed linearly, this does not match the human ear's perception of frequencies, which is sensitive to changes in the lower frequency bands and sluggish in the higher frequency bands. Thus, based on the STFT, the FFT bins are grouped and smoothed according to Mel Scale after taking the log-amplitude of the magnitude spectrum. The frequencies of the spectrogram transformed by Mel Scale enable the human ear to perceive signals with the same frequency difference almost equally well. Mel Scale is the result of a non-linear transformation.

With *librosa.feature.melspectrogram*, we compute the Mel-Spectrogram using a Hanning window of 2,048-sample long and 512-sample hop length, and n_mel = 20. (The configurations of the parameters mentioned below are all the same as those of Mel-Spectrogram, and will not be repeated afterwards.) And then we get a Mel-Spectrogram of size 20×1500. To analyze this graph, we average each row and perform Principal Component Analysis (PCA) so that each Mel-spectrogram could be a point. Samples with the same label are labeled with the same color and are used to observe the distribution of each class. The result is shown in Figure 3a. It can be seen that the points are very unevenly distributed in the figure, with a portion overlapping on the left side, while the yellow, blue, and pink classes are scattered on the right side. Since pink, blue and yellow correspond to rock and pop, electronic, metal and punk, whose genres are indeed similar. At the meantime, there is a big gap with the remaining genres on the other side, so it can be seen that the Mel-spectrogram feature has a significant effect on genre classification.

2.2.2 *Mel-Frequency Cepstral Coefficients*

Mel-Frequency Cepstral Coefficients (MFCC) is obtained by performing Discrete Cosine Transform (DCT) on the basis of Mel-Spectrogram. Although DCT is just a linear transformation which has little effect on neural network, some research showed that Mel-Spectrogram may reduce the performance in some specific models because of its high correlation [7]. Considering that the three models we choose have different properties, we decide to use both the Mel-Spectrogram and MFCC.

With *librosa.feature.mfcc*, we compute the MFCC sequence using 20 MFCCs. To analyse the MFCC, we do the

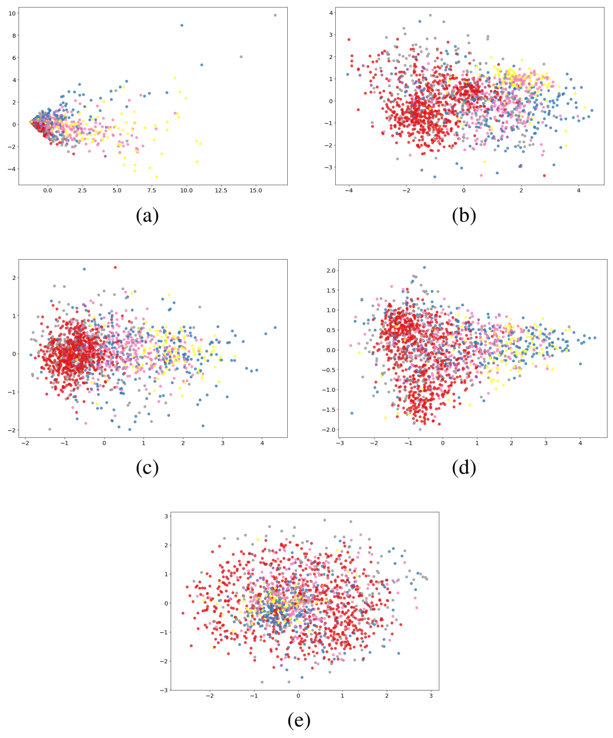


Figure 3: Distribution of features: (a) Mel-Spectrogram; (b) MFCC; (c) Chromagram; (d) Constant-Q Chromagram; (e) Chroma Energy Normalized Statistics. The color of points matches to the labels as: red-classical, blue-electronic, purple-jazz and blue, yellow-metal and punk, pink-rock and pop, grey-world

same operation for MFCC as we do for Mel-Spectrogram, and the result is shown in Figure 3b. Unlike the Mel-Spectrogram distribution, the points are evenly dispersed in the MFCC distribution. The red and gray are clustered on the left while the pink, blue and yellow are clustered on the right. As mentioned before, the scatter plot shows that MFCC is also able to distinguish widely different genres, even better than Mel-spectrogram.

2.2.3 *Beats and Tempo*

Beats and Tempo are fundamental aspects of music. The function *librosa.beat.beat_track* will return a global tempo in beats per minute and beat event locations using dynamic programming beat tracker [8]. The size of beats may not reach 1500, so we fill in zero at the end of the feature.

2.2.4 *Spectral Centroid*

The spectral centroid is defined as the center of the 'gravity' of the magnitude spectrum and can indicate the 'brightness' of a sound. Its value of the t-th frame is

$$C_t = \frac{\sum_{k=0}^{K-1} f(k)X_t[k]}{\sum_{k=0}^{K-1} X_t[k]} \tag{1}$$

where $X_t[k]$ represents the magnitude of bin number k, and $f(k)$ represents the center frequency of that bin. The histogram of spectral centroid for each genre is shown in Figure 4. It can be seen that the distribution of each genre is different from each other and can be fitted with multiple Gaussian distributions. Compared to MFCC, although this feature has a certain distinguishing effect on different genres, it is not so good as MFCC. Especially for classical and eletronic music, the only difference is the length of the right trailing.

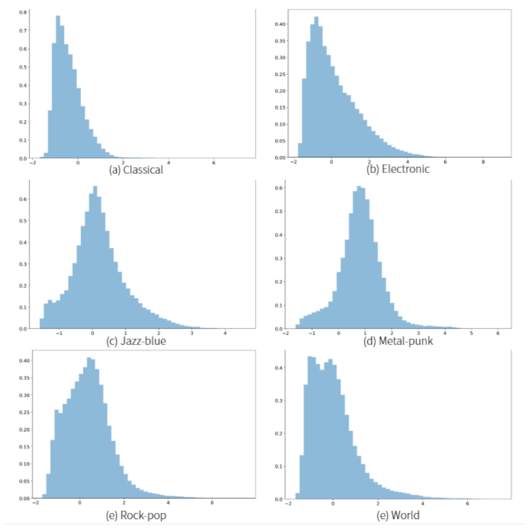


Figure 4: The histogram of Spectral Centroid

2.2.5 *Chromagram*

Chroma feature is a collective term for Chroma Vector and Chromagram. Chroma Vector is a feature vector containing 12 elements indicating how much energy of each pitch class. The function *librosa.feature.chroma_stft* returns the normalized energy for each chroma bin at each frame.

By plotting the distribution of Chromagram in Figure 3c, we can see that its overall distribution is similar to that of MFCC, which proves it a suitable feature for genre classification.

2.2.6 *Root-mean-square Energy*

The root-mean-square energy (RMSE) represents the energy of each frame. With librosa.feature.rms, we can compute RMSE value for each frame. The histogram, which is shown in Figure 5, can also be fitted with Gaussian model. Since pink, blue and yellow correspond to rock and pop, electronic, metal and punk, whose genres are indeed similar. At the meantime, there is a big gap with the remaining genres on the other side, so it can be seen that the Mel spectrogram feature has a significant effect on genre classification.

2.2.7 *Zero-crossing Rate*

The zero-crossing represents the noisiness of a sound. The zero-crossing rate is the fraction of zero-crossing and can be defined as

$$Z_t = \frac{1}{N-1}\sum_{n=0}^{N-1} |sign(x[n]) - sign(x[n-1])| \quad (2)$$

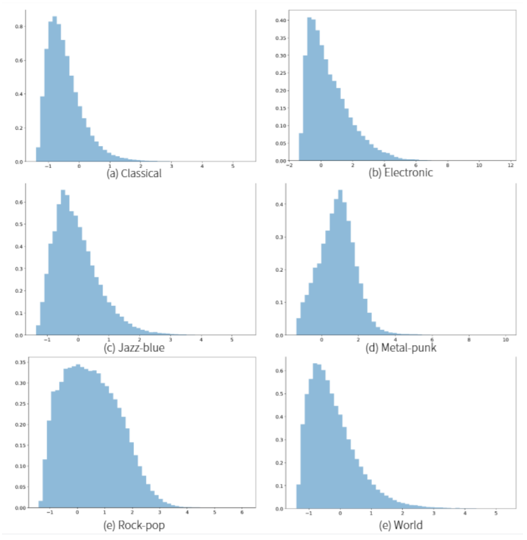


Figure 5: The histogram of Root-mean-square Energy

where the $sign$ function is 1 for positive arguments and 0 for negative arguments and $x[n]$ is the time domain signal of length N.

With *librosa.feature.zero_crossing_rate*, we compute the zero-crossing rate and plot the histogram as Figure 6. It can be seen that the distribution of zero-crossing rate is relatively similar. There are four genres that have similar graphs, which means that the zero-crossing rate is poorly discriminative relative to the features above.

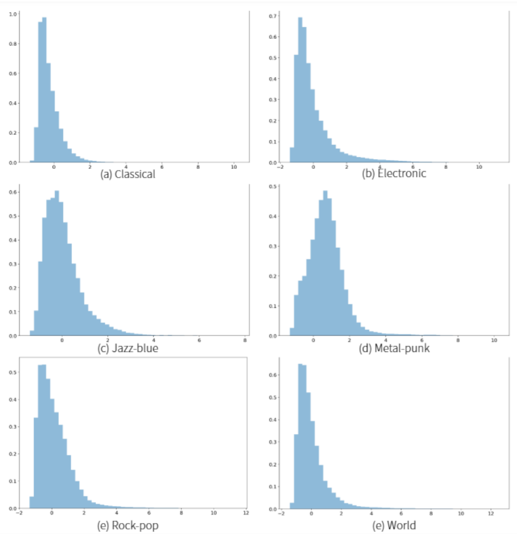


Figure 6: The histogram of Zero-crossing Rate

2.2.8 *Constant-Q Chromagram*

Constant-Q Transform (CQT), which is related to Fourier Transform, is a bank of filters in which the center frequency is distributed exponentially and the filter bandwidth is different, but the center frequency to bandwidth ratio is a constant Q. Q can be defined as

$$Q = \frac{f_k}{\triangle f_k} \tag{3}$$

$$f_k = f_0 \cdot 2^{\frac{k}{b}} (k = 0, ...) \tag{4}$$

where f_k is the center frequency, $\triangle f_k$ is the filter width, f_0 is the minimal center frequency and b is the number of filters per octave.

With *librosa.feature.chroma_cqt*, we compute the chromagram of CQT signal. And its distribution is also changed as the point of red class are spread out in the vertical axis, which is shown in Figure 3d. This suggests that it is also discriminative, but relatively less effective because there is more overlap between the different genres.

2.2.9 *Chroma Energy Normalized Statistics*

Considering short-time statistics over energy distributions within the chroma bands, we can obtain Chroma Energy Normalized Statistics (CENS), which features are robust to dynamics, timbre and articulation [9]. With *librosa.feature.chroma_cens*, we can obtain the CENS-Chromagram. Figure 3e shows the distribution of CENS, which is greatly different from the distribution before. All the points are coincident, very evenly distributed in the center of the plot, showing no difference at all.

2.2.10 *Spectral Rolloff*

The spectral rolloff is another measure of spectral shape and is defined as the frequency below which 85% of the distribution magnitude is concentrated

$$\arg\min_{f_c \in \{1,...,N\}} \sum_{i=1}^{f_c} X_t[k] \geq 0.85 \cdot \sum_{i=1}^{N} X_t[k] \tag{5}$$

where f_c is the rolloff frequency.

With *librosa.feature.spectral_rolloff*, we get rolloff frequency of each frame using Hanning window. The histogram of spectral rolloff is shown in Figure 7. It is obvious that this feature works well for different genres.

2.3 Normalized and data split

After concatenating these features as a image, we standardize them using

$$z = \frac{x - mean}{std} \tag{6}$$

where z is the output feature, x is the input feature, $mean$ is the mean value of all samples and std is the standard deviation of all samples.

We split the data randomly into 90% for training set and 10% for testing set using function *train_test_split* from *sklearn*. Finally, we obtain 1567 features for training and 175 features for testing.

2.4 Model Construction

In this part, we first build a convolutional neural network with default parameter configurations. Then, we train the model by adjusting parameters and setting different model configurations to obtain best performance. The full pipeline is shown in Figure 8.

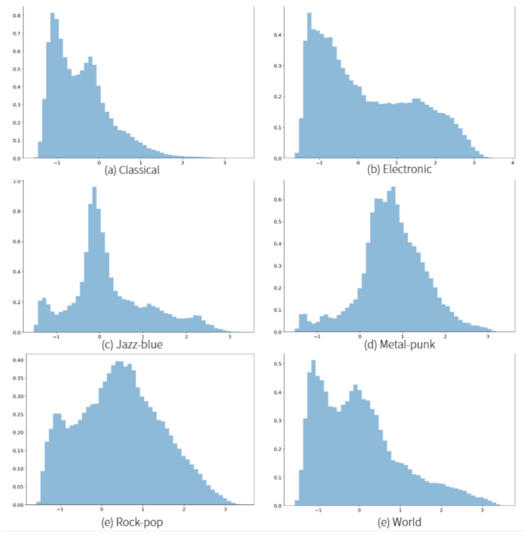
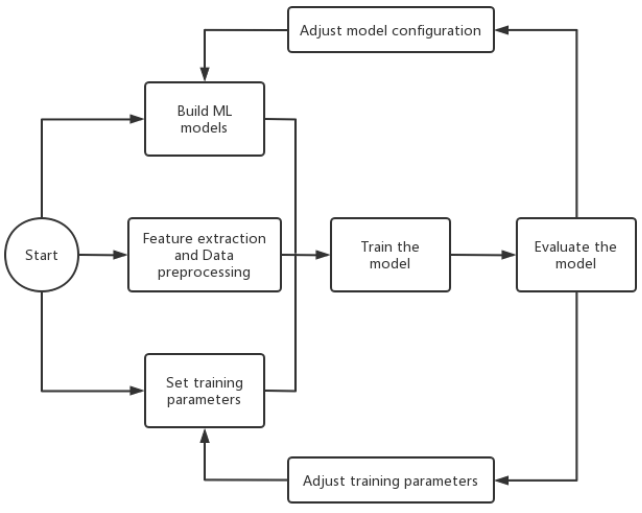


Figure 7: The histogram of Spectral Rolloff



Adjust model configuration
Build ML models
Start
Feature extraction and Data preprocessing
Train the model
Evaluate the model
Set training parameters
Adjust training parameters

Figure 8: Pipeline of building and training a ML model

2.4.1 *CNN*

The default configuration is as below:

- Convolutional layer 1: Filter size: 7×7, stride: 1, zero-padding: 'same', number of filters: 64
- Pooling layer 1: Max pooling, filter size: 2×2, stride: 2, zero-padding: 'same'
- Batch normalization layer
- Leaky ReLU
- Convolutional layer 2: Filter size: 5×5, Stride: 1, zero-padding: 'same', number of filters: 128
- Pooling layer 2: Max pooling, filter size: 2×2, stride: 2, zero-padding: 'same'
- Batch normalization layer
- Leaky ReLU

- Convolutional layer 3: Filter size: 5×5, Stride: 1, zero-padding: 'same', number of filters: 256
- Pooling layer 3: Max pooling, filter size: 2×2, stride: 2, zero-padding: 'same'
- Fully connected layer 1: Output size: 512
- Batch normalization layer
- Leaky ReLU
- Dropout regulation: Rate: 0.5
- Fully connected layer 2: Output size: 128
- Batch normalization layer
- Leaky ReLU
- Dropout regulation: Rate: 0.5
- Fully connected layer 3: Output size: 6
- Softmax classifier

Among them, each convolutional layer consists of several convolutional units, and the parameters of each convolutional unit are optimized by the backpropagation algorithm. The purpose of the convolution operation is to extract different features of the input.

As for the pooling layer, it divides the input image into several rectangular areas, and outputs the maximum value for each sub-area. The pooling layer will continuously reduce the size of the data space, so the number of parameters and the amount of computation will also decrease, which also controls overfitting to a certain extent.

We choose Leaky ReLU as our activation function. The part of the Leaky ReLU input less than 0 has a negative value and a small gradient. Since the derivative is always non-zero, this reduces the occurrence of silent neurons, allowing gradient-based learning (albeit slow), so the Leaky ReLU function works better than the ReLU function.

The fully connected layer is to expand the feature map (matrix) obtained by the last layer of convolution into a one-dimensional vector and provide input to the classifier.

Dropout traverses each layer of the network and sets the probability of eliminating nodes in the neural network. Assuming that at each layer in the network, the probability of each node being retained and eliminated is 0.5, we will eliminate some nodes and end up with a network with fewer nodes and a smaller scale.

The softmax function can normalize the output value and convert all the output values into probabilities, where all the probability values add up to 1. When doing classification, the category with high probability value is the predicted category.

The default configuration is shown in Figure 9.

For the default training parameters, exponential decay learning rate is implemented. First we choose 0.01 as our initial learning rate to quickly get a better solution. Then we gradually reduce the learning rate through iteration to make the model more stable in the later stage of training.

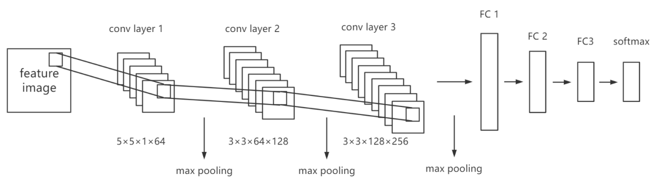


Figure 9: Default CNN configuration

2.4.2 SVM

First, we need to vectorize the 81×1500 array of each sample and run PCA to reduce data to 1567 elements. PCA is the most commonly used linear dimensionality reduction method. Its goal is to map high-dimensional data into a low-dimensional space through some kind of linear projection, and expect the data to have the largest amount of information (maximum variance) in the projected dimension. In this way, fewer data dimensions are used, while retaining the characteristics of more original data points. We choose 1567 elements because we need to consider the trade-off between accuracy and computation speed.

When doing SVM classifier, we use SVC because it is for nonlinear data. Then, to train the model, we use *model.fit*, passing in the feature value matrix and the class labels. In this way, the model will automatically train the classifier. We can use *model.predict* to predict the result, pass in the sample feature matrix, and get the predicted classification result prediction of the test set.

2.4.3 GMM

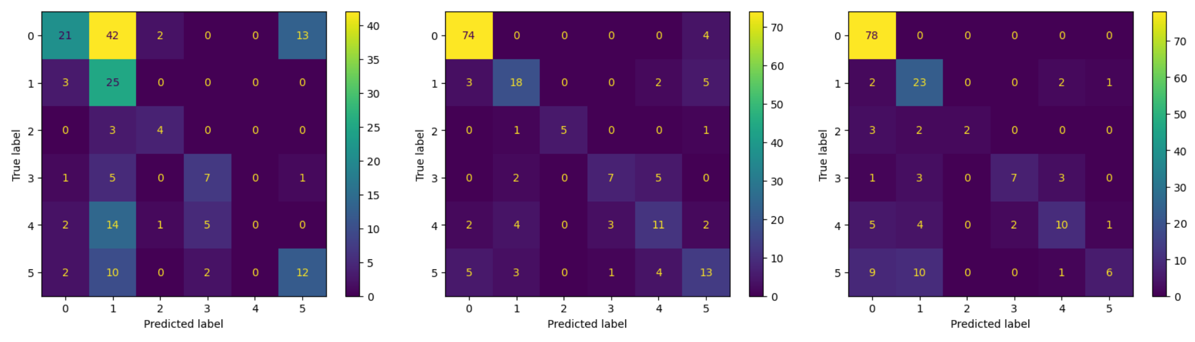
First, we do the PCA as SVM.

For GMM, Its essence is to fuse several single Gaussian models to make the model more complex and generate more complex samples. Theoretically, if there are enough Gaussian models fused by a Gaussian mixture model, and the weights between them are set reasonably enough, this mixture model can fit samples of any distribution. For this training dataset, we use *GaussianMixture* and set n_components = 4 for each class so that four Gaussian distributions are used to approximate the feature distribution of each class. Then, we use model.score_samples to compute the weighted log probabilities for each sample on the six models respectively. The model corresponding to the maximum probability shows the predicted class.

3. RESULTS

We first calculate testing accuracy of the three models based on the above default network structure and training parameters, which are shown in Table 1. It is obvious that SVM is the best, followed by GMM, and the worst is CNN. Among them, the accuracy of GMM and SVM are almost the same with more than 70%, while the accuracy of CNN is only 40%, which is relatively low.

Figure 10 shows the confusion matrix of the three models. It can be observed that for CNN a large part of the testing dataset is identified as the second type: electronic. Most of classical music are recognized as electronic probably due to several features like RMSE, zero-crossing



(a) (b) (c)

Figure 10: Confusion matrix of the three models

Table 1: Testing accuracy of the three models

	CNN	SVM	GMM
Testing accuracy	39.43%	73.14%	72.00%

rate, CQT, CENS. As mentioned above, for these features, these two genres are not well differentiated. For SVM and GMM, the recognition is relatively accurate with a few errors.

Since the accuracy of CNN model is unsatisfactory, we make some parameter modifications. We change the batch size from 16 to 32 and also change the filter size of convolutional layers to 7×7, 5×5 and 5×5 respectively. Then we get the accuracy of 54.86% with nearly 15% improvement. The improved confusion matrix is shown in Figure 11, we can see an obvious improvement, although there are still quite a few songs identified as the second type, whose reason is mentioned above.

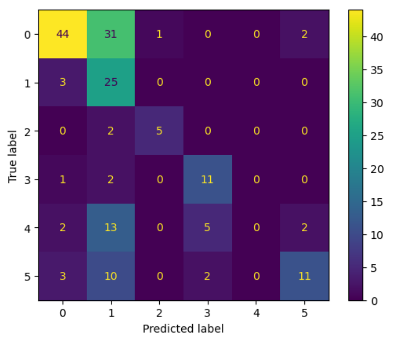


Figure 11: Confusion matrix of improved CNN

4. DISCUSSION AND CONCLUSION

In this project, we analyze the main features of music and use them to implement music genre classification. We choose three different methods to recognize the genre of a specific song snippet. They are evaluated and compared through testing accuracy and confusion matrix. Based on the results, all the three training methods don't seem to achieve very high accuracy. Especially for CNN, it performs the worst on test set. This is due to several features like RMSE, zero-crossing rate, CQT, CENS. As mentioned above, for these features, these two genres are not well differentiated. Maybe in the next step, we can remove some features to see whether it has improved.

In addition, there are a couple of things that may increase accuracy:

- Use SVM on convolutional neural network output (the output of the convolutional layers)
- Use GMM on convolutional neural network output (the output of the convolutional layers)
- Try more complex model, like CNN + Bi-RNN [10], which indicates in contrast to utilizing CNNs alone, all of the CNNs with paralleling RNN can improve the performance of music genre classification.

5. REFERENCES

[1] George Tzanetakis and Perry Cook. Musical genre classification of audio signals. IEEE Transactions on speech and audio processing, 10(5):293–302, 2002.

[2] S. Dieleman and B. Schrauwen, “End-to-end learning for music audio,” in Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on, pp. 6964–6968, IEEE, 2014.

[3] Hamel, Philippe and Eck, Douglas. (2010). Learning Features from Music Audio with Deep Belief Networks.. Proceedings of the 11th International Society for Music Information Retrieval Conference, ISMIR 2010. 339-344.

[4] Thiruvengatanadhan, R.. “Music Genre Classification using Chromagram and GMM.” (2019).

[5] Jean-Julien Aucouturier and François Pachet. Improving timbre similarity: How high’s the sky? Journal of Negative Results in Speech and Audio Sciences, May 2004.

[6] Berenzweig, Adam, Daniel Ellis, Beth Logan, Brian Whitman. ”A Large Scale Evaluation of Acoustic and Subjective Music Similarity Measures.” In Proceedings

of the 2003 International Symposium on Music Information Retrieval. 26-30 October 2003, Baltimore, MD.

[7] K. -H. N. Bui, H. Oh and H. Yi, ”Traffic Density Classification Using Sound Datasets: An Empirical Study on Traffic Flow at Asymmetric Roads,” in IEEE Access, vol. 8, pp. 125671-125679, 2020, doi: 10.1109/ACCESS.2020.3007917.

[8] Ellis, Daniel PW. “Beat tracking by dynamic programming.” Journal of New Music Research 36.1 (2007): 51-60. http://labrosa.ee.columbia.edu/projects/beattrack/

[9] Meinard Müller and Sebastian Ewert “Chroma Toolbox: MATLAB implementations for extracting variants of chroma-based audio features” In Proceedings of the International Conference on Music Information Retrieval (ISMIR), 2011.

[10] Feng, Lin and Liu, Shenlan and Yao, Jianing. (2017). Music Genre Classification with Paralleling Recurrent Convolutional Neural Network.