Neural Network for Music Style Recognizing

Keli Zheng, Rex Xing, Roy Huang

zhengkeli2009@126.com

Abstract

We introduced a method for music style (genre) recognition (classification) based on MFCC, CQT and image processing convolutional neural network. We firstly convert the audio into two spectrograms – MFCC-spectrogram and CQT-spectrogram. Then we analyze them as images with an image processing neural network. The neural network receives two spectrograms as input images and returns the probabilities of class as its output. To fight with the overfitting, we used random clipping, noise adding and regularization during the training. From the results, our model can perform the classification task with relatively high accuracy compared with the baseline model.

Key words: music genre, MFCC, CQT, CNN

1 Introduction

Music Genre Classification is a popular problem in machine learning with several applications. It can be used to tag every song in a huge music corpus with genre and sub-genre which can be used to identify similar songs. Another application is for a music recommendation system. By using the features extracted from an intermediate layer in the model, we can cluster similar songs and those can be shared with users based on their history behavior.

The traditionally method is to extract features directly from audio and used them as the starting point for a classification task. Since the features is too simple, usually they cannot describe the music well. The model cannot really "understand" the music. So, the accuracy of the model is limited.

We believe that, there is a better approach. We introduced image processing neural network for audio recognition. The idea is that, we firstly convert the audio into an image. Then we can use the image processing neural network for classification. There are already many methods to perform the audio-to-image conversion. In this project we used 2 kind of such conversion – MFCC and CQT.

The task of our model is to classify. The input of our model should be a piece of audio. The model should analyze the audio and return the probabilities of all classes. We can pick the one with the highest probability as a prediction. Or we can also keep all probabilities to detailly describe the characteristics of this audio.

The main scheme is shown in Figure 1. We firstly load the audio files as an audio wave. Then we convert the wave into 2 kind of spectrograms. Then the spectrograms are analyzed by the neural network. The output of the neural network is the probabilities of classes.

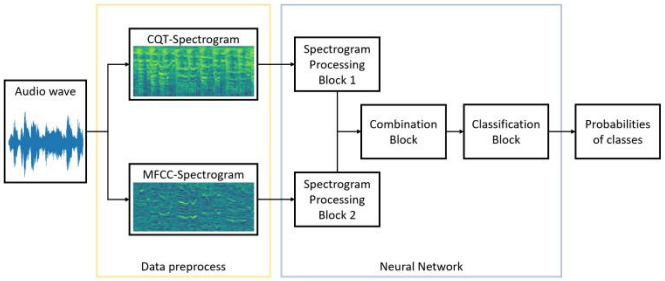

Figure 1. The main scheme

In this project, we are using GTZAN Genre Collection as our dataset, which has 1000 songs of 30 seconds each divided equally into 10 genres, the ten genres are Blues, Classical, Rock, Hip-Pop, Reggae, Country, Disco, Jazz, Pop and Metal. The sample rate of each audio file is 22050 Hz. [1]

2 MFCC-Spectrogram

Mel Frequency Cepstral Coefficients (MFCCs) are widely used feature in speech recognition. We want to convert each audio file into a graph which is a visual representation of MFCCs over time.

In this project, we are using library *librosa* to extract MFCCs [2]. The extraction can be divided into the following steps.

2.1 Framing and Windowing

Since speech signals are time-varying signals, for the sake of processing, assume that it is a stationary system over a short period of time. Therefore, we usually pick frames from the audio signal. The frames have a fixed length and they can

overlap each other. The distance between distance between the head of two frames is called hop length.

In this project the sample rate of audio is 22050 Hz. We use the default configuration of *librosa* for framing. The frame length is set to 2048 so, the frame duration is about 92.9 milliseconds. The hop length is set to 512, so the hop duration is about 23.2 milliseconds.

Windowing is a common operation after the framing. That is, when processing this frame, weight the values of different positions with the window function $w(n)$. Choosing a suitable window function can make the processing results more continuous and avoid glitches.

As a convention, we use the Hamming window in this project.

2.2 Discrete Fourier Transform (DFT)

This was inspired by the human cochlea (depending on the input sound, it will resonate in different places). And depending on the location of the resonance, different neurons send different signals to the brain to tell it what frequency the sound corresponds to. We use periodic diagrams to estimate the power spectrum to achieve a similar effect.

$$X[k] = \sum_{n=0}^{N-1} w[n] \cdot x[n] \cdot e^{-2\pi i \frac{kn}{N}}, 1 \le k \le N$$

In this formula:

$x[n]$ – the n-th amplitude of the original audio wave.

$X[k]$ – the k-th value of the spectrum after DFT.

$w(n)$ – the window function (In this project we use Hamming window).

N – length of spectrum X after DFT. (In this project $N = 2048$)

As we known, $X[k]$ as the result of DFT, is complex number. For the further processing, we estimate the power spectrum:

$$P[k] = |X[k]|^2$$

2.3 Calculate the Mel Filter Bank

Human's hearing is less sensitive to high frequency. the cochlea does not differentiate between two frequencies very well, especially for high frequency signals.

Using mel-frequency, the MFCCs can better describe the human's hearing. This is the relation between frequency and mel-frequency:

$$Mel(f) = 2595 * lg\left(1 + \frac{f}{700}\right)$$

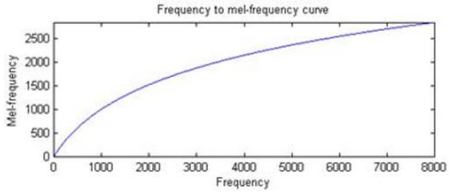

Figure 2. Frequency to mel-frequncy curve

We divide the frequency range into different buckets, and then add up all the energy that falls into the bucket. This is done with the Mel Filter Bank. According to relation between frequency and mel-frequency, the first filter is very narrow, and it collects frequencies close to 0Hz. The further back, the wider the filter gets, the larger the range of the frequencies it collects.

A set of approximately 20~40 (usually 26) triangular filter banks is shown in Figure 3. It will filter the power spectrum estimation of the periodic diagram obtained in the previous step. Our filter bank consists of 26 vectors with a length of 257. Most of the 257 values of each filter are 0, and only for the frequency range to be collected are non-zero. The input signal of 257 points will pass through 26 filters, and we will calculate the energy of the signal passing through each filter.

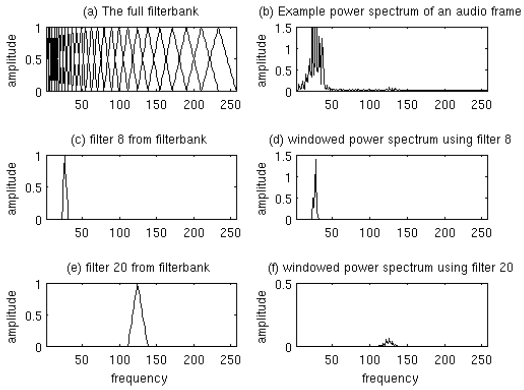

Figure 3. Power Spectrum of Mel Filter Banks and Windows

The calculation process is shown in the figure below. Finally, we will retain the energy of these 26 filters. Figure 3 (a) shows 26 filters; Figure 3 (b) is the filtered signal; Figure 3 (c) is the eighth filter, which only allows signals from a certain frequency range to pass through. Figure 3 (d) the energy of the signal passing through it; Figure 3 (e) is the 20-th filter; Figure 3 (f) is the energy of the signal passing through it.

2.4 Operate the Logarithm on the Energy

And then for the energy of the filter bank, we take the log of it. It is inspired by human hearing: human feelings on loudness is not linear. To double the size of the human perception, we need to increase the energy eightfold. This means that if the sound is loud enough, then a little more energy will not make a significant difference to perception. This compression operation of log makes our features closer to human hearing. And why log instead of cube root? Because log allows us to use a channel normalization technique called cepstral mean subtraction which can normalize the differences between different channels.

This step is quite simple, just operate the logarithm on 26 energy:

$$s(m) = ln\left(\sum_{k=0}^{N-1} P[k]\, H_m(k)\right), 0 \le m \le M$$

In this formula:

N – the number of sample points in DFT.

$H_m(k)$ – the response frequency of the filter.

M – the number of filters.

2.5 Discrete Cosine Transform

The last step is to apply DCT transform to these energies. Since different Mel filters have intersection, they are correlated, and we can use DCT transform to remove these correlations, which can be utilized in subsequent modeling.

DCT was performed on the signals of these 26 points, and 26 cepstral coefficients are obtained. And at last, we keep the 12 numbers from 2~13, and these 12 numbers are called the MFCC features. The purpose of redoing DCT on the power spectrum is to extract the envelope of the signal. We only take the 2~13 coefficients because the latter energies represent high-frequency signals that change rapidly, they have been found in experiment to make the recognition worse.

2.6 Normalization

After the above processing for all frames. We put all MFCCs frame by frame along the x-axis. So, each row is the changing values of one MFCC over time. Then we get a spectrogram of MFCCs (Figure 4).

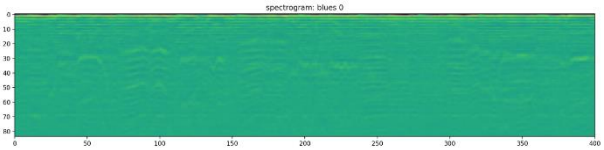

Figure 4. Original MFCC-spectrogram (Blues 0)

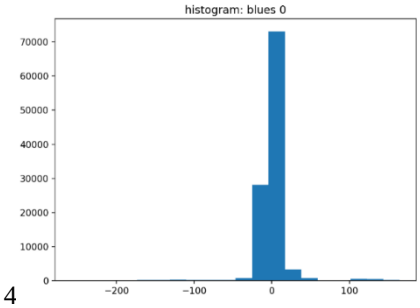

Figure 5. Histogram of original MFCC-spectrogram

As we can see in the histogram (Figure 5), the value varies widely. The shapes in the spectrogram is not obvious. Such spectrogram cannot be well identified by neural network, so we need to do the normalization.

It is worth noting that the normalization is done for every MFCC (every row) separately, not for the whole spectrogram. We do this because the values of different MFCC have too different scales.

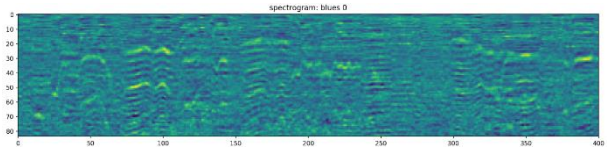

Figure 6. Graph After Normalization (Blues 0)

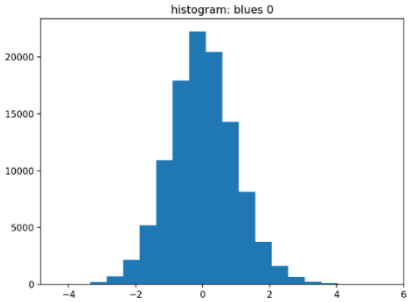

Figure 7. Histogram After Normalization

After the normalization, the shapes in the spectrogram is clearer (Figure 6). The values are mostly between -3 and 3 (Figure 7). This can help our neural network learn and identify better.

Some representative processed MFCC-spectrograms are shown below. We can already see that spectrograms of different genres are quite different.

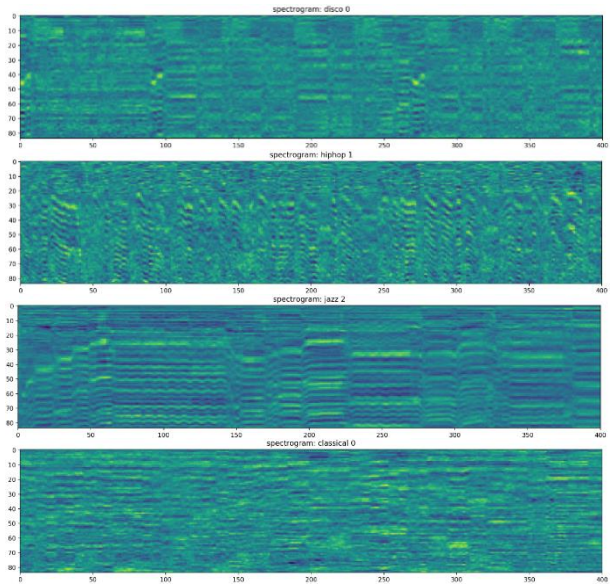

Figure 8. Example MFCC-spectrograms of disco 0, hiphop 1, jazz 2 and classical 0

3 CQT-Spectrogram

Constant quality transform (CQT) is a time-frequency transformation algorithm with the same exponential distribution law. In music, the sound is exponentially distributed, but the audio spectrum obtained from our Fourier transform is linearly distributed, and the frequency points of the two cannot correspond one by one, which will lead to errors in the estimation of the frequency of some scales. Therefore, CQT is generally used in the modern analysis of music sound.

In this project, we are using library *librosa* to perform the CQT [2]. The extraction of CQT-spectrogram can be divided into the following steps.

3.1 Framing and Windowing

To match the time axis of two spectrograms, we use the same configuration for framing as the previously mentioned MFCC-spectrogram (see section 2.1).

3.2 Constant quality transform

As mentioned in section 2.2, the traditional discrete Fourier transform is:

$$X[k] = \sum_{n=0}^{N-1} w[n] \cdot x[n] \cdot e^{-2\pi i \frac{kn}{N}}, 1 \le k \le N$$

We can write the formula like this:

$$X[k] = \sum_{n=0}^{N-1} w[n] \cdot x[n] \cdot e^{-2\pi i \frac{nf_k}{f_s}}, 1 \le k \le N$$

$f_s = \frac{N}{T}$ – the sampling frequency of the audio.

$f_k = \frac{k}{T}$ – the k-th frequency of the Fourier transform.

T – the duration of the audio frame.

Then the resolution of frequency is

$$\Delta f = f_k - f_{k-1} = \frac{n}{T}$$

As we can see, the resolution of frequency is constant, that is, the frequencies are evenly picked.

However, after the Fourier transform, we will find that the characteristics are dense in the low frequency region and sparse in the high frequency region. Such spectrogram cannot clearly reflect the information of music. This is not good for audio recognition.

So, instead of Fourier transform, we use constant quality transform (CQT). The essence of CQT is variable resolution of frequency. We select more points for low frequency and less points for high frequency.

The "constant quality" mean that the quality factor Q is constant:

$$Q = \frac{f_k}{\Delta f} = \frac{f_k}{f_k - f_{k-1}}$$

Then the frequencies are exponentially distributed

$$f_k = \left(\frac{Q}{Q-1}\right)^k f_0$$

According to Nyquist sampling law, if you want to identify the corresponding amplitude value in frequency domain of f_k, the selected number of points should be

$$N_k = \frac{f_s}{f_k - f_{k-1}} = Q\frac{f_s}{f_k}$$

That is, for different frequency f_k we can use different frame size. When doing the framing, we only need to keep the largest N_k as the frame size. When doing the CQT we can use different frame sizes to reduce the amount of calculation.

Then the formula of CQT transform can be written as:

$$X[k] = \sum_{n=0}^{N_k-1} w[n] \cdot x[n] \cdot e^{-2\pi i \frac{nf_k}{f_s}}$$

$$= \sum_{n=0}^{N_k-1} w[n] \cdot x[n] \cdot e^{-2\pi i \frac{nQ}{N_k}}$$

3.3 Normalization

Like the Fourier transform, the result we get after the CQT is complex numbers, which contain not only the information about the amplitude, but also the phrase. To draw a spectrogram, we only need the amplitude. So, we firstly we need to get the absolute value. Then we get a spectrogram like this:

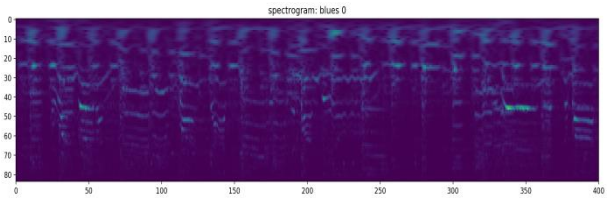

Figure 9. Original spectrogram (blues 0)

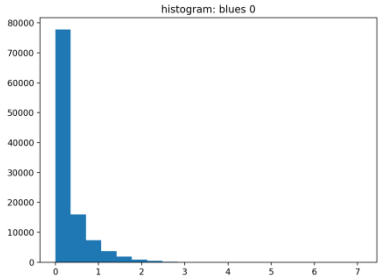

Figure 10. Histogram of the amplitudes

As we can see in these two graphs, the spectrogram is not clear enough to discern the image transformation.

The relation between loudness and humans' hearing is not linear. The decibel is a better unit to describe loudness than the power. So, we convert the values to decibels. The conversion formula is:

$$b = 10 \lg \frac{a^2}{\max a^2}$$

a – the amplitude got after the CQT.

a^2 – the power.

$\max a^2$ – the max power in the whole spectrogram, used as a reference value.

After the conversion, we got a spectrogram like this:

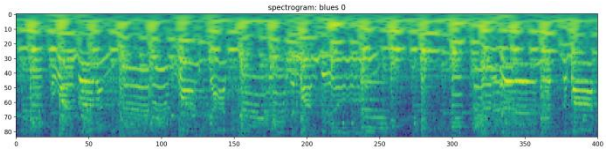

Figure 11. Spectrogram in decibels (blues 0)

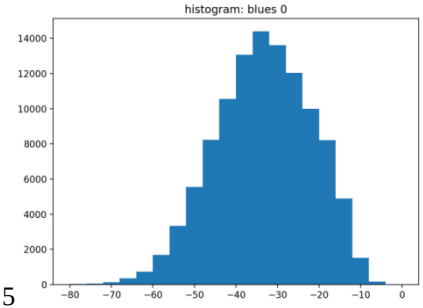

Figure 12.Histogram of amplitudes in decibels.

As we can see in these two graphs, the spectrogram is clearer. The values are basically normally distributed. To make the data more suitable as an input to the neural network we still need to do the normalization, which makes the mean value to be 0 and the variance to be 1.

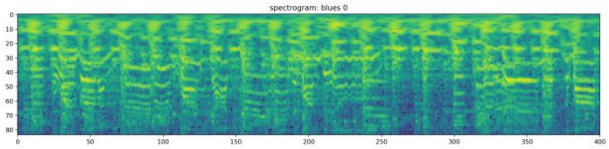

Figure 13. Spectrogram after normalization (Blues 0)

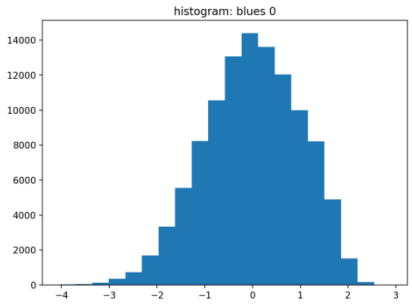

Figure 14. Histogram after normalization

After the normalization, the values are mostly between -3 and 3, which is suitable as an input to a neural network.

Some representative processed CQT-spectrograms are shown below. We can already see many structures in the spectrograms. Each shape in the spectrogram corresponds to a sound in the audio.

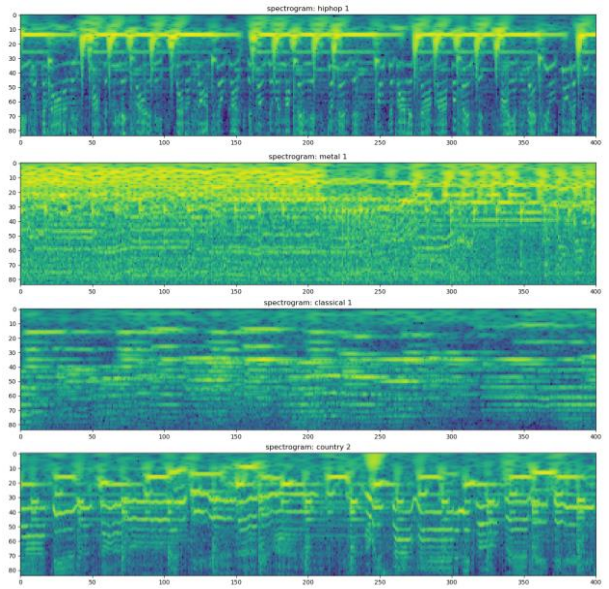

Figure 15. Example MFCC-spectrograms of hiphop 1, metal 1, classical 1 and country 2.

5

4 Neural Network

As shown in the main scheme in Figure 1, the neural network is mainly composed of 4 parts – 2 spectrogram-processing blocks, combination block and classification block. In this project those spectrogram-processing blocks have the same structure. The spectrogram processing block is actually some kind of image processing block, which analyzes the spectrogram as an image. The combination block combines the results from those spectrogram-processing blocks. And finally, the classification block organizes the results into probabilities of classes.

4.1 Spectrogram-processing Block

The structure of the spectrogram-processing block is shown in the following Figure 16.

As we can see, it draws on the neural network model of image processing in the design of this block. There are multiple layers of convolutional neural networks. The activation function is ReLU. Each convolution layer is followed by a max pooling layer and batch normalization layer. There are 5 such structures in total.

In the last layer is a reshaping. The axis of y and axis of channels are combined. The axis of x (axis of time) is remained as is. So that the spectrogram is compressed from a 2-dimensional data into a 1-dimensional data (with a channels).

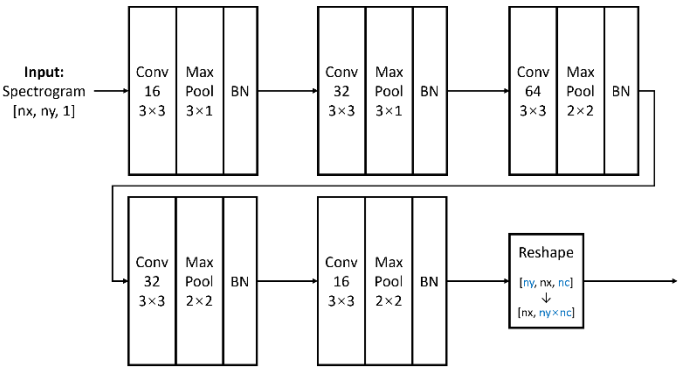

Figure 16. The structure of spectrogram-processing block

The reason why we use an image-processing structure is that, the preprocessed spectrograms have a translation symmetry. The x-axis is the time axis. The x axis is the time axis, the same sound, no matter when it appears, people will recognize it as the same. Correspondingly, in the spectrogram the same shape (corresponding to some sound) can be translated along the x-axis, keeping its nature (the sound) unchanged. The y-axis is the frequency axis. In the data preprocessing the frequencies are arranged according to human hearing. So that the translation along the y axis corresponds a translation of tone. In such translation the melody is still the same, but the tone becomes higher or lower. Therefore, the spectrograms have a translation symmetry along both x-axis and y-axis. And that is why we use an image-processing structure to recognize the sounds.

The reason why we place a reshaping layer at the end is that the spectrogram is too long. Unlike images, because the x axis corresponds to time, and music can have a long duration, the spectrogram is usually long. This brings some difficulties if when using an image-processing structure. Our solution is that we firstly use a shallow convolutional network to recognize local shapes, so that we can make full use of translation symmetry. Then we convert the 2-dimensional spectrogram into a 1-dimensional sequence (along time axis). So that we can handle the timing relationships further.

4.2 Combination Block and Classification Block

The structure of the combination block is shown in the following Figure 17.

The two spectrograms are obtained in different ways, but they correspond in time. So, after obtaining the 1-dimensional sequences, we can combine the results from both spectrogram processing blocks. In order not to lose information, we use concatenate them together (along the channel axis).

After the concatenation, we use a 1-demensional convolutional layer to process the 1-dimensional sequence. The activation function is ReLU. The convolutional layer is followed by a 1-dimensional max pooling layer.

Then there is an averaging layer here. The averaging goes along the x-axis (time axis). So that the output is fixed. And this neural network can process audio with arbitrary duration.

Finally, the classification block will organize the result into probabilities of classes. The classification block is simple. There is a fully connected layer and a softmax layer.

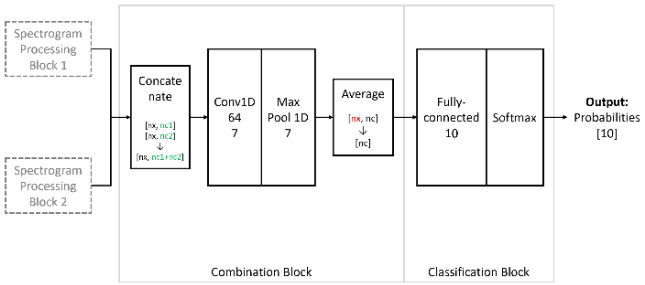

Figure 17. The structure of combination block and classifcation block

4.3 Other details

The input spectrogram has a total of 84 values on the y-axis. That is, from each frame we extract 84 MFCCs and we perform the CQT for 84 frequencies. We unify the sizes of the two spectrograms in order to process them using neural networks with the same structure.

As a convention, we use the combination of softmax and cross entropy for classification.

The probabilities are calculated using this formula:

$$p(k) = \frac{e^{y_k}}{\sum_j e^{y_j}}$$

$p(k)$ – the probability to be the k-th class.

y_k – the k-th output from the fully-connected layer.

And the loss function is:

$$\begin{aligned}\text{loss}(\boldsymbol{p}_{true}, \boldsymbol{p}_{pred}) &= \text{crossentropy}(\boldsymbol{p}_{true}, \boldsymbol{p}_{pred}) \\ &= -\sum_j p_{true}(j)\, log\, p_{pred}(j)\end{aligned}$$

$\boldsymbol{p}_{true}$ – the real probabilities (one-hot encoded).

$\boldsymbol{p}_{pred}$ – the k-th output from the fully-connected layer.

In addition to this, a L2 regularization is applied on all "kernel weights". So called kernel weights are such weights, by which the layer inputs are that are multiplied. The regularization is not applied on the weights that are added to the layer inputs (bias). Applying regularization is to add a component to the loss.

$$\text{total\_loss} = \text{crossentropy}(\boldsymbol{p}_{\text{true}}, \boldsymbol{p}_{\text{pred}}) + \alpha \sum_{w \in \text{ker.ws}} w^2$$

α – a parameter to adjust the "strength" of regularization.

w – one of the kernel weights.

This component will be large if the weights are large. So that the weights are prevented to be large. We use regularization to overcome overfitting.

5 Experiments

5.1 Training

During training, the main difficulty is the overfitting. Some tricks are applied to prevent overfitting.

The first is the random clipping. That is, during the training, every time we randomly cut out 5 seconds for the whole 30-second audio. But for validation and prediction, we use the whole audio.

In addition, during the training we add noise to the spectrograms. That is, we add random values on each point of the spectrograms. In this project we used gaussian distributed noise. Such noise in the spectrogram corresponds to pink noise in audio.

The dataset is split into 3 subset – training set, test set, validation set. The ratio 6:2:2, respectively. The training set is for training. The test set is used to compare the accuracy of the model when adjusting the hyperparameters. The validation set is used to estimate the accuracy of the final model. When splitting the data set, data of each class is split separately to ensure that the amount of data of each class is consistent.

The optimizer we chose for training is the commonly used Adam optimizer.

5.2 Results

In order to judge whether the image processing model for audio recognizing is superior, we compare it with a baseline model. This model is done by *Parul Pandey* [3]. The inputs of the baseline model are such features like MFCCs, zero-crossing rate, spectral centroid, chroma etc. These features are firstly extracted as short-term features. But the features are scalars – the average along time. (By contrast, we keep all short-term features and use them as spectrograms.) The model itself is a simple neural network composed of 4 fully-connected layers. (By contrast, our model is a complicated image processing model.)

As the results given by the author on the website, the baseline model can achieve about 65% accuracy on our test set. But the accuracy on our validation set is much lower – about 50%. Surprisingly, the results on the test set and the verification set are very different. Our model has a significantly higher accuracy (about 75%) on both test set and validation set.

The exact values and confusion matrixes are shown in Table 1, Figure 18 and Figure 19.

	Test set	Validation set
Baseline model	62.0%	51.0%
Our model	74.5%	75.0%

Table 1. The accuracy of models on different subsets.

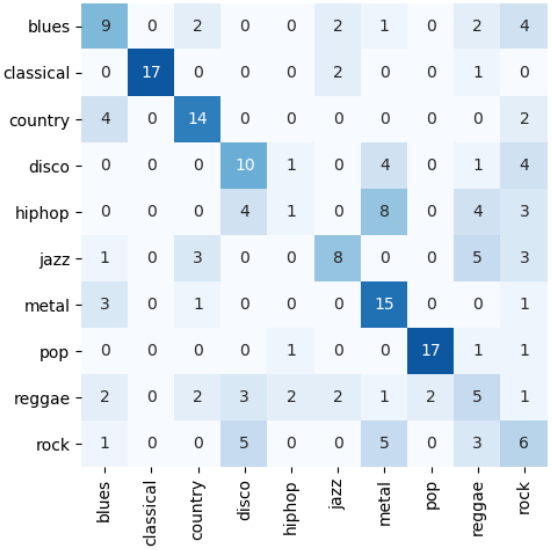

Figure 18. The confusion matrix of baseline model (on validation set)

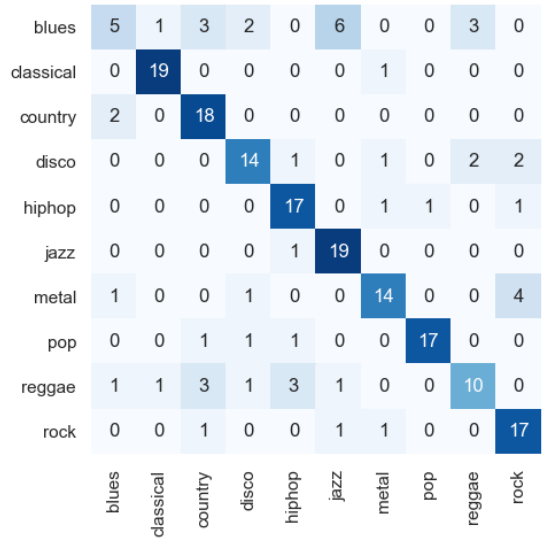

Figure 19. The confusion matrix of our model (on validation set)

6 Conclusion

We introduced a method of music style recognition based on MFCC, CQT and image processing convolutional neural network.

From the results, this method is greatly improved compared to the baseline model. This shows that it is feasible to use the image processing method for audio recognition.

Perhaps such image processing method is not only suitable for music genre recognition, but also for music feature extraction (music indexing), audio content recognition, and even audio generation.

The further work can be:

- Try more methods for spectrogram extraction.
- Optimize the neural network.
- Train the model with a bigger dataset or dataset of other types.
- Try other types of tasks than classification.

References

[1] GTZAN Genre Collection.

Available: http://marsyas.info/downloads/datasets.html

[2] Librosa.

Available: https://librosa.github.io/librosa/

[3] *Parul Pandey*. Music Genre Classification with Python.

Towards Data Science.

Available: https://towardsdatascience.com/music-genre-classification-with-python-c714d032f0d8