Deep learning COMP47650

Assignment

Classification of GTZAN dataset

Name: Qiang Wang

Student number: 17203085

Abstract

This paper will introduce some features of music briefly, such as zero crossing rate, spectral centroid, Melhr frequency cepstrum coefficients, etc. The librosa library in Python will be used to extract the above features from GTZAN audios. The generated features data will be used to train and test the neural network, so that it can classify 10 different types of music in GTZAN dataset. In this paper, the applied neural network is relatively simple, which is a fully connect deep neural network, and the classification accuracy can reach about 68-70% by it. In addition, adding dropout layer to the neural network can effectively prevent its overfitting and improve the classification accuracy to a certain extent.

Keywords: GTZAN dataset, DNN, Audio classification, Audio features

1. Introduction

At present, music has become a part of our life, listening to music can make us relax, edify sentiment. In addition, music signal processing, analysis, classification, recognition and so on have become important directions of artificial intelligence research and people have made considerable achievements in these aspects, for example some companies are developing music software that can accurately identify what music is being heard and send the results back to customers. This paper will introduce a kind of basic artificial intelligence application in audio, that is, audio classification.

The task of classifying music is difficult because machines cannot 'listen to music' directly, which means they cannot get information directly from the raw music. The information available to the machine is a series of signal information(digital, graph, etc.) rather than audio playing. Therefore, we need to use some means to extract the useful information from music and convert it into information that can be recognized by the machine. Through the long-term research of scholars from all walks of life, it is found that some features in music can be used to analyze music. Let us take a look at the features in the music:

1. Audio spectrum and chromatogram are the most important images for audio signal processing. After the original audio signal is divided into frames, FFT (fast Fourier transform) is done for each frame. The function of Fourier transform is to convert the time domain signal to the frequency domain signal. After each frame has done FFT, the frequency domain signal is stacked up in time to get the audio spectrum. Chromatogram, also known as pitch class profile, shows the distribution of energy across a series of notes, usually this are the twelve scales of music (C, C# D, D# E, E# F, G, G# A, A# B). We can think of the chromatogram as a fold of the CQT on the evaluation axis. Some parameters, such as chromatogram STFT, chromatogram CQT, chromatogram CENS, etc. can be obtained during the calculation of spectrum and chromatogram, which can be used as the features of the classification task[1].
2. Zero-crossing rate[2]: refers to the rate of change of signal symbol, that is, the number of times that the audio signal changes from positive to negative or from negative to positive per frame. It is widely used in music retrieval and audio recognition.

3. Spectrum centroid[2]: refers to the location of the audio's "centroid", which is calculated according to the weighted average of the frequency of the audio.

4. Spectrum roll down[3]: denotes the frequency of spectral energy at a specific percentage. It is a measurement of the shape of the data signal.

5. Melhr frequency cepstrum coefficient[1]: is a set of 10 to 20 features that simply describe the overall shape of the spectrum and simulate the characteristics of the human voice;

Since different music has different above features values, and which of different kinds of music are quite different, we can use this characteristic to classify music. Librosa is a Python library for audio. It is very powerful for common audio processing, such as feature extraction, drawing audio graphics, etc. In this paper, I will use Librosa library to extract the above features in music, and it can automatically convert such features into digital information and output them. In addition to the above mentioned Librosa, there are also MadMom, PyAudioAnalysis and other commonly used music processing libraries, which are open source and powerful function libraries as well.

Deep Neural network is a very effective tool to deal with value-data information. It can fit all kinds of linear and non-linear data relations. In this paper, the fully connected DNN will be builed to fit the relationship between the music features extracted by Librosa and the music labels. Moreover, it was found through experiments that the performance of DNN on the training set was much higher than that on the validation set and the test set. This was because DNN overfitted the training set, so in this paper, dropout will be used to shut down some neurons. Through experiments, it was found that dropout method can effectively reduce the overfitting and improve the accuracy of test set and validation set by about 8%.

2. Related Work

The first step of audio classification is undoubtedly to extract the features in the audio. As far as GTZAN data set is concerned, at present, scholars mainly apply existing feature extraction libraries to carry out this operation, such as Librosa mentioned above. There has not been much progress in feature extraction in these years.

In recent years, scholars have been focusing on classification methods, such as machine learning algorithms(decision tree, random forest, logistic regression, support vector machine, etc., and deep learning methods(DNN, CNN and RNN). The following will briefly introduce the research progress of the above algorithms on audio classification.

1. Decision tree algorithm: The decision tree model is built by using the library function in the sklearn library. The criterion to measure the classification is set as information gain entropy, and other parameters are set as default values. The model was trained according to the decision tree algorithm, and the convergence speed of the model was fast, and the average classification accuracy was about 50%.[4]

2. Logistic regression: Use sklearn to build logistic regression model, using its default parameters. Call L2 regular penalty term. The average accuracy of the model was about 66%.

3. Random forest: In the experiment of decision tree classification model, the forest depth is set as 20, the criterion is set as Gini, and the number of iterations is set as 500. The average accuracy of model classification is about 68% when the model is trained according to the random forest algorithm, which is greatly improved compared with the accuracy of decision tree model.

4. SVM classification model: because the data distribution is linear inseparable, select gaussian kernel function, penalty factor is set to 6, the classification accuracy rate to an average of about 78%.

5. CNN model: Hguimaraes extract the mel-spectralgram in music, and use it to feed the CNN. Add a dropout layer between each two convolution layers, and connect layers to each other through ReLU activation function. Finally, output ten different results through softmax activation function. In this model, there are 2,495,114 parameters to be trained, so the convergence speed is relative slow. The accuracy rate of the final test is about 72%.

6. Fully connected deep neural network: this is the method I used, which will be introduced in detail in the following chapters.

Among all the methods mentioned above, support vector machine algorithm has the best performance, and DNN algorithm has the highest efficiency. In addition, there are other ensemble algorithms that can achieve better performance. For example, multiple CNN models are trained at one time, and the result with the highest possibility is finally selected by the use weight voting mechanism. This algorithm[5] can achieve about 83.2% of the classification effect, which is the state-of-the-art technology for the classification of GTZAN data set at present.

3. Experimental Setup

3.1 Extract the features from audios

Load the music file with the following command by Librosa library[5]: y, sr = librosa.load(Path, mono=True, duration=Duration). The length of the extracted audio can be defined by the user but cannot exceed the total length of the audio. Load the audio as a waveform 'y' and store the sampling rate as 'sr' for later using.

Use the following commands to extract the corresponding features values[5]:

chromatogram CQT	librosa.feature.chroma_cqt(y=y, sr=sr)
chromatogram STFT	librosa.feature.chroma_stft(y=y, sr=sr)
chromatogram CENS	librosa.feature.chroma_cens(y=y, sr=sr)
RMSE	librosa.feature.rms(y)
Spectral centroid	librosa.feature.spectral_centroid(y=y, sr=sr)
Spectral bandwidth	librosa.feature.spectral_bandwidth(y=y, sr=sr)
Spectral rolloff	librosa.feature.spectral_rolloff(y=y, sr=sr)
Zero crossing rate	librosa.feature.zero_crossing_rate(y)
Melhr frequency cepstrum coefficient	librosa.feature.mfcc(y=y, sr=sr)

Table1 Commands to extract the features

The above-mentioned Mehr Frequency Cepstrum Coefficient is the value of 20 continuous and different MFCC groups, and a loop is needed to split the values of each group respectively.

Librosa extracts each of the above features for many times and continuously, and obtains a continuous array of features related to time. Numpy is used to average the values in the array,

so that each feature of a single song outputs a value that can be passed into the subsequent neural network.

The values of the averaged features are stored into a Pandas dataframe. Users can use the data in the dataframe directly for training and testing, or they can output the data in the dataframe into a CSV file for future long-term use.

3.2 Data preprocessing

3.2.1 Features values preprocessing

In the original training data, due to the different sources of features and measurement units, the distribution range of eigenvalues will vary greatly, which will have a great negative impact on the model. For example, when calculating the Euclidian distance between different samples, features with a wide value range will play a leading role. So, each feature is normalized to the same value range, and the correlation between different features is eliminated to obtain the ideal data. Use the following formula to standardize the data[6]:

$$x = \frac{x - \mu}{\sigma}$$

The raw data is subtracted from the mean and then divided by the variance (or standard deviation) to get the standardized data, which conform to the standard Gaussian distribution, with a mean of 0 and a standard deviation of 1. This kind of data has the following advantages: 1. Improve the convergence speed[7]: different value ranges will cause the gradient direction in most positions is not the optimal search direction. When the gradient descent method is used to find the optimal solution, it will require many iterations to converge. If we normalize the data to the same value range, the gradient direction of most positions is close to the optimal search direction. In this way, in the solution of gradient descent, the gradient direction of each step basically points to the minimum value, and the training efficiency will be greatly improved; 2. Improve model accuracy[7]: In machine learning or deep learning, loss calculation of most models requires the assumption that all features of data are zero mean, all feature attributes can be processed in a unified way when loss is calculated.

3.2.2 Label preprocessing

Since the existing label is the name of the corresponding type of the song, namely classical, blue, jazz, disco, etc. This kind of string label cannot be recognized by the machine directly, so

we need to pre-process the label data. What we use here is the label-encoding mechanism. In essence for label-encoding, n different labels are encoded as integers between 0 and n-1. For example, there are three color labels: red, yellow, and blue, we can set red=1, yellow=2, blue=3.

3.3 Split the dataset

In the machine learning algorithm, we always split the dataset into 3 sub-set(Splitting should be as consistent as possible in data distribution), which are training set, validation set and test set respectively. Validation Set is used to avoid overfitting and we usually use it to tune some hyper-parameters in the training process(for example, to tune the epoch size of early stop and the learning rate). The reason why testing sets are not used here is that with the progress of training, the network will slowly overfit the test set, resulting in no reference significance of the final testing set. Therefore, the training set is used to calculate the weight and update the gradient; the validation set is used to tune the hyper-parameters and select the model; and the testing set provides an accuracy to judge the network's performance.

In this design, the whole data set is divided into three parts as above: training set, verification set and test set. The proportion is determined by the user and the default proportion is training set 60%, verification set 20% and test set 20%

3.4 Build Neural Network

In this paper, DNN network was built to fit the linear and nonlinear relationship between features and labels. The structure is shown in the figure below:

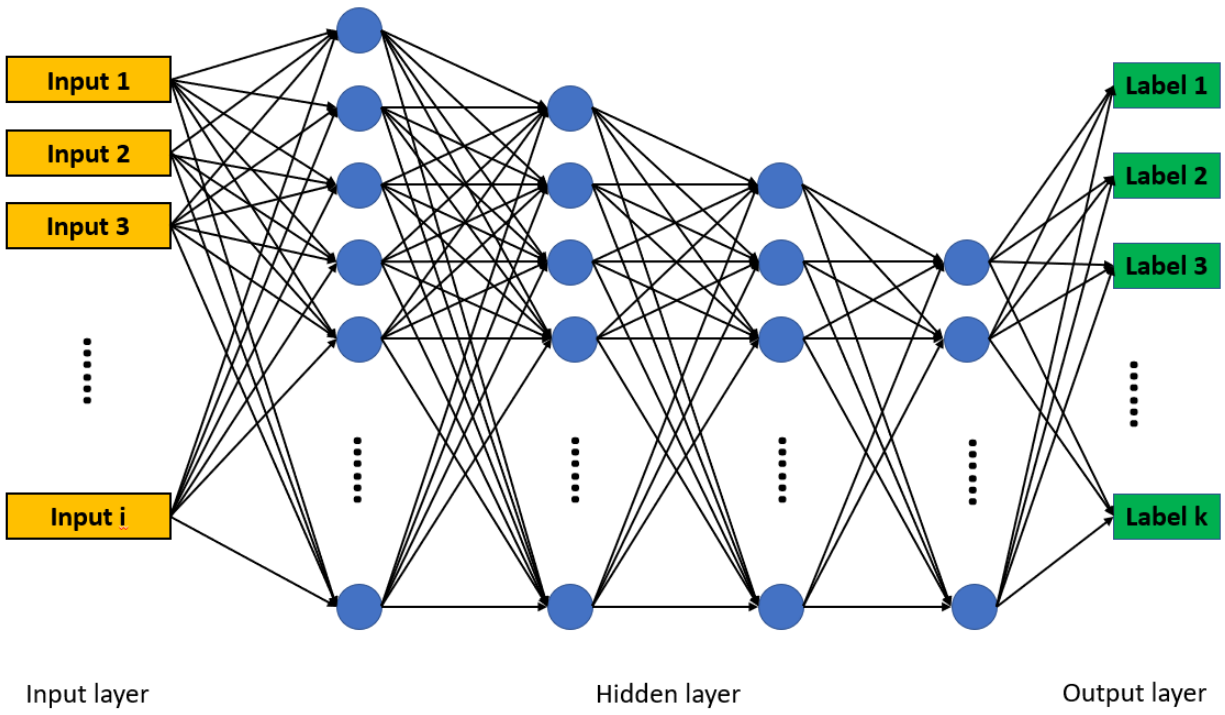

Figure1 DNN structure

DNN can be thought of as a neural network with many hidden layers. Multi-layer neural network and deep neural network(DNN) actually refers to one thing, DNN is sometimes called multi-layer perceptron (MLP). The neural network layer inside DNN can be divided into three types, namely, the input layer, the hidden layer and the output layer.

In this neural network, the input data is the features mentioned above. In the design of this project, the script will automatically identify the input size according to the user's input features, and then automatically adjust the input layer size of the neural network. The last layer is the output layer, the output layer is the corresponding label value, i.e., blue, classical, disco, jazz, country, hip-hop, metal, pop, reggae and rock. The layers between the input layer and the output layer are hidden layers.

The neural network layer is fully connected to each other, in other words, any neuron in layer i must be connected to any neuron in layer i+1. In fact, small local models are the same as perceptron, i.e., a linear relationship $z = \Sigma w_i x_i + b_i$ and an activation function.

In this neural network, ReLU function is used as the activation function between the input layer and the hidden layer as well as between each hidden layer. The last layer of the hidden layer is connected to the output layer by a softmax function. TensorFlow can automatically

constructs the sparse categorical cross entropy loss function, and minimizes the loss function by back propagation and gradient descent to make the model converge. The neural network calls the ADAM optimizer.

In the deep learning model, if the model has many parameters and few training samples, the trained model is easy to overfit. Overfitting is: the model has less loss in training data and higher prediction accuracy; the loss of test data is relatively large and the prediction accuracy is low. Overfitting is a common problem in many machine learning systems. If the model is overfitted, it will be almost unusable. To solve the overfitting problem, model ensemble is generally used, that is, multiple models are trained to combine. At this point, the time spent on training the model becomes a big problem, not only the training of multiple models is time-consuming, but also testing multiple models is time-consuming. To sum up, when training deep neural network, there are always two disadvantages: easy to overfit and time consuming.

Dropout can effectively relieve overfitting and achieve regularization effect to a certain extent. Moreover, the training time of the neural network after dropout is not significantly different from that of the original neural network.

Dropout means that in the training process of deep learning network, neural network units are temporarily dropped from the network according to a certain probability, as shown in the figure below. In general, the proportion of neurons temporarily shut down is set as 0.3-0.5 (drop rate).

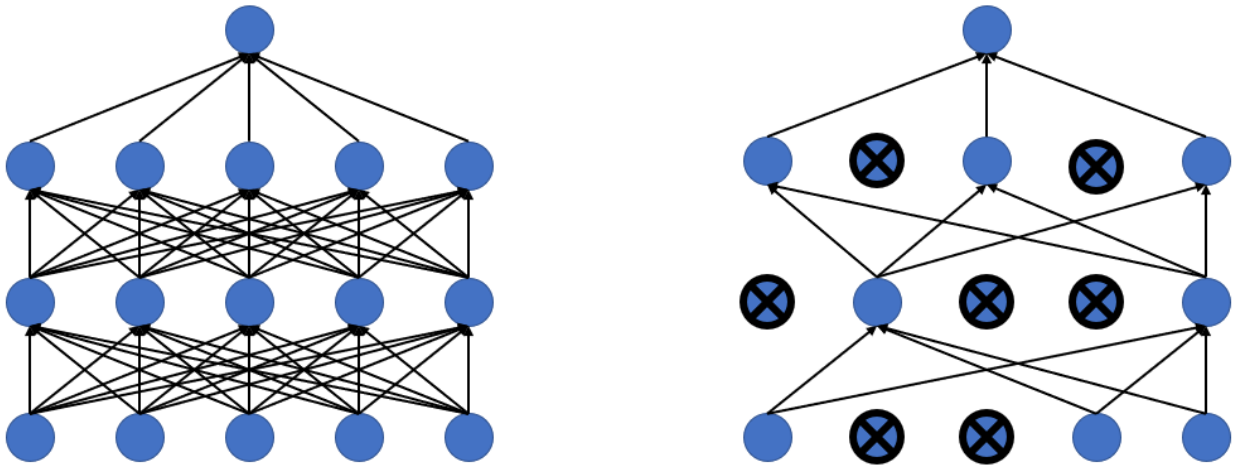

Figure2 DNN with dropout versus DNN without dropout

In this model, after the initial test, it was found that the original neural network （without dropout） would overfit the relationship between features and labels, so the dropout layer was added into the model. After the test, it was found that the overfitting was effectively reduced and the accuracy rate was improved to a certain extent. We'll talk about that later in the result part.

3.5 Trian and test

Features are extracted according to the method in 3.1, and each file will obtain a features vector of 1*28, among which MFCC occupies 20. DNN was built as shown in Figure 2, and 4 hidden layers were set, among which the number of units in the hidden layer was 256, 128, 64, 32 respectively, and the number of units in the output layers was 10. In the end, there are 50,986 parameters to be trained in the whole neural network model. After testing, the training time is generally about 6 seconds.

After the operation in 3.2 above, we can get the pre-processed training set, verification set and test set. After many attempts, I found that for the GTZAN data set, the model basically converges after 100 epochs of training with the neural network mentioned above. For this design, the user can customize the number of training epochs.

3.6 Scripts introduction

The user is recommended to use the command line. Users can do the following customizations on the command line:

1. Customize the <ID> of this round of training and testing: the results of training and testing will be saved in the folder named <ID>. The script will create a folder named ‘results’ in the directory where the project is running, and this folder will include a user-defined folder named <ID>.
2. Customize running mode: training&test mode and feature extraction mode. By default, the script goes into train&test mode. If the user feeds path to the --generate-features on the command line, the script will extract the features of the audio from that path.
3. Load the feature file. By default, the script automatically looks for the feature file from the path where the current project is running. If the user has other feature files

stored elsewhere on the PC, you can load the feature files by calling --load-features-path.

4. The number of epochs of training, the default value is 100.

5. Random seed of shuffling data set, the default value is 0.

6. Batch size. Default value is 64.

7. The size of the test set and the training set. The default value is 0.2.

8. Customize whether to call Dropout or not.

9. Customize whether the current script is only for testing

10. Call the previously trained model file, skip the training process and directly enter the testing process. If the user enters the corresponding path in the --models, the script will go to that path to load the model file. If the user does not enter a path, the script will look for the model file in the folder where the current project is running.

4. Results

First, I used the original neural network without any dropout processing for training, and the training results obtained are shown in the figures below.

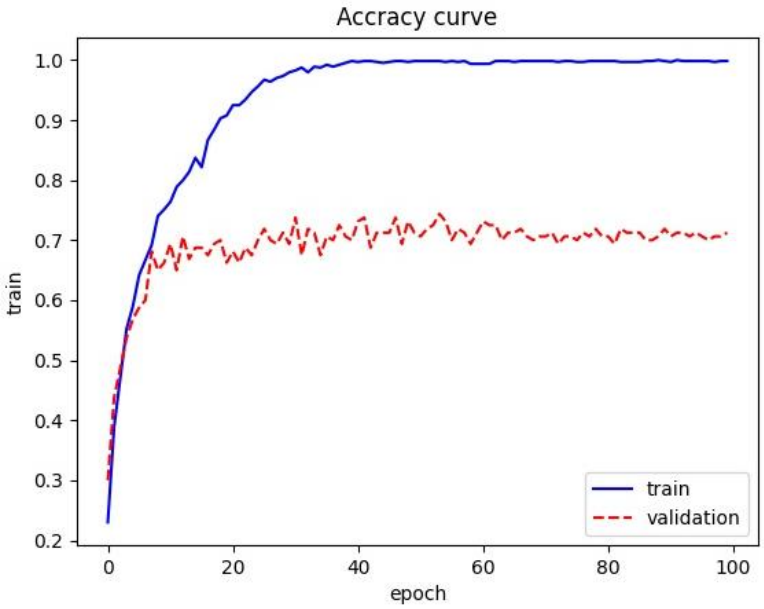

Figure3 Accuracy curve of pure DNN

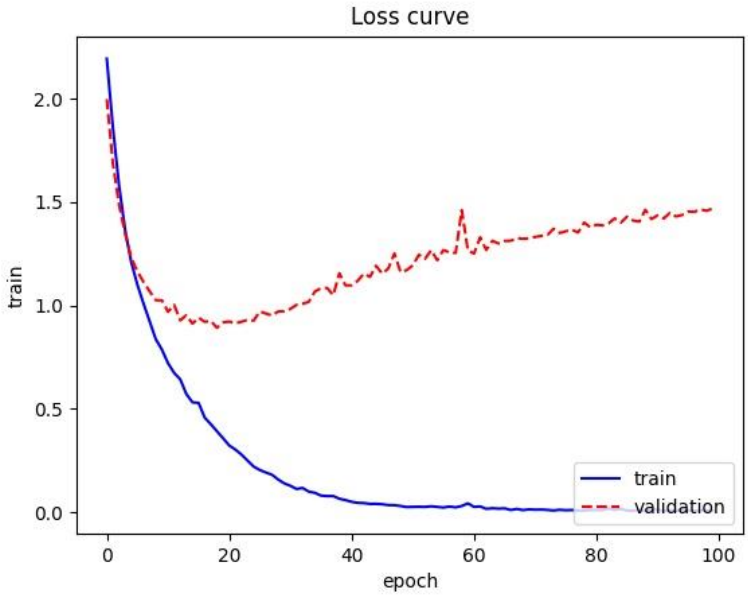

Figure4 Loss curve of pure DNN

The performance of the model on the training set is far greater than that on the test set. The model in this training section is an overfitting model, which is basically useless and does not perform well on the test set. The accuracy for this model is around 66% as shown in the following figure.

```
10/10 [==============================] - 0s 2ms/step - loss: 0.0071 - accuracy: 0.9971 - val_loss: 1.4694 - val_accuracy: 0.7125
7/7 [==============================] - 0s 518us/step - loss: 1.8651 - accuracy: 0.6600
The tested accuracy is 0.6600000262260437
```

Figure7 Accuracy of pure DNN on the test set

Secondly, I used the neural network after dropout for training, and obtained the results as shown in the figures below.

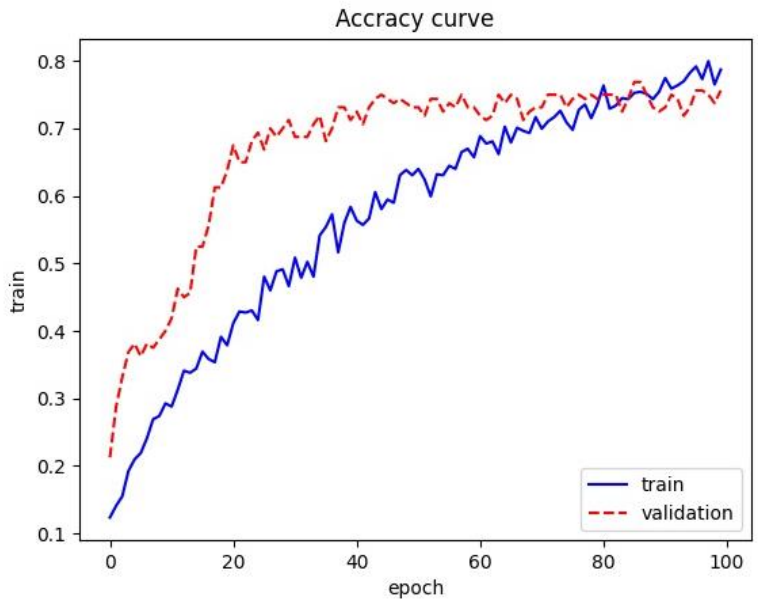

Figure5 Accuracy curve of DNN with dropout

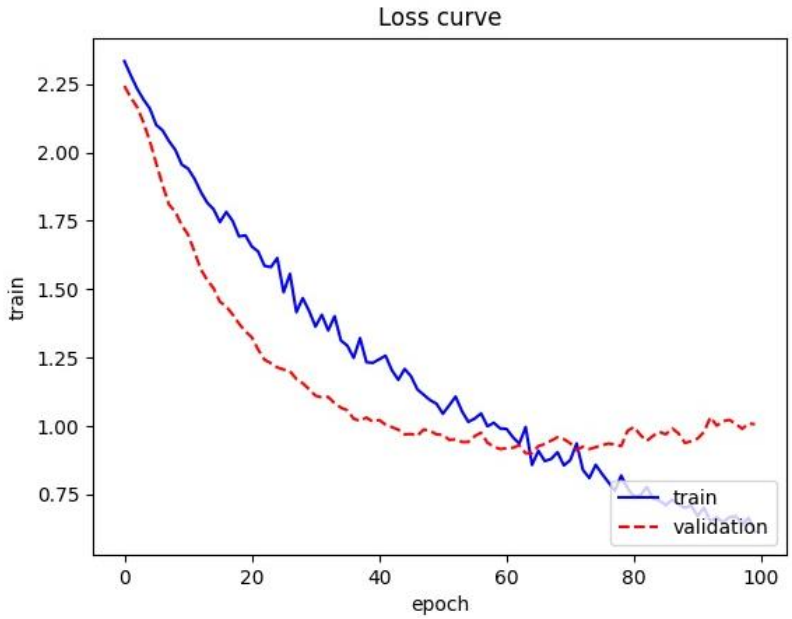

Figure6 Loss curve of DNN with dropout

The performance of the model on the validation set is basically the same as that on the training set. The obtained model has strong practicability, and its performance on the test set is also improved to a certain extent compared with that of the un-dropout neural network. The accuracy for this model in the test set is nearly 75.5%, which is shown in the figure 7.

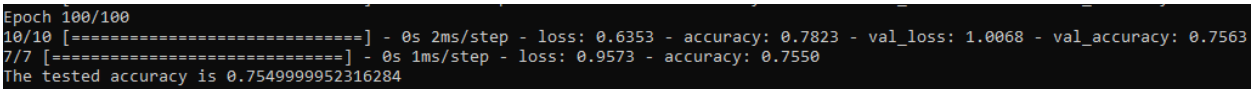

Figure7 Accuracy of DNN with dropout on the test set

5. Conclusion & Future Work

The most difficult part in the application of audio to artificial intelligence is to accurately collect audio files and extract feature information that can be recognized by computers. As for the task of classification of GTZAN data set, Librosa Library in Python can be used for effective extraction of features, but the extraction speed needs to be improved. The extraction time for 1000 30-second files is about 17-20 minutes. After we get the original features of the data, a more efficient data set can be obtained through basic conventional preprocessing.

Fully connected neural network is a good choice for the classification task of GTZAN dataset. It is easy to understand and has high efficiency. In the case of no GPU acceleration, convergence can be achieved after only a few tens of seconds of training, and the accuracy rate of about 75.5%. The dropout layer should be properly added to reduce overfitting. At present, fully connected neural network is not the most accurate classification method, but in my opinion, it is the most cost-effective classification method we can select.

In future research, scholars still need to focus on but not limited to:

1. Explore new audio features and improve existing ones.
2. Study new neural network or deep learning technology that is more suitable for audio classification to improve the accuracy and efficiency of audio classification. For example, the subsequence of audio is predicted by RNN through time, and classification is carried out at the same time of prediction.
3. New feature extraction method to shorten the time required for feature extraction.

Reference

[1] Brian, M., Colin R. and Dawen, W. (2015) Librosa: Audio and Music Signal Analysis in Python. New York University： Center for Data Science.

[2] Theodoros, G and Aggelos, P. (2014) Introduction to Audio Analysis. London: Addison Wesley. Bibliography. pp 251-258

[3] 91xueshu. (2020) Comparative analysis of various machine learning models based on music data set GTZAN. Available at https://www.91xueshu.com/l-yinyuelw/46018.html (Accessed 15 April 2021).

[4] Hguimaraes. (2019) gtzan.keras. Available at https://github.com/Hguimaraes/gtzan.keras (Accessed 15 April 2021).

[5] Librosa. (2021) Librisa tutorials. Available at https://librosa.org/doc/latest/index.html (Accessed 15 April 2021).

[6] Zakaria.J (2021) When and why to standardize your data? Available at: https://builtin.com/data-science/when-and-why-standardize-your-data (Accessed 15 April 2021).

[7] Dalong (2019) The benefits of standardization and common approaches. Available at: https://zhuanlan.zhihu.com/p/88348005 (Accessed 15 April 2021).