Comp497650 individual project report

Yinjie Liu
Affiliation: University College Dublin
student number: 20211091
phone: 0871687153
email: yinjie.liu@ucdconnect.ie

Abstract

This report is to introduce the my individual project in COMP47650. Nowadays, it is very common to use deep neural network model to deal with all kinds of data. This article is about how I preprocess and segment a large number of audio files and classify them by deep learning technology. It is very interesting that I have implemented different deep neural network models, and I have different ways of data processing according to different neural network models.

1 Introduction to my project

First of all, my task is to classify a large number of audio files in multiple categories through deep neural network technology. In other words, I need to preprocess and split these data and then transfer them into my model for training. The training result is the one with the highest possibility among ten different music categories.

Specifically, how do I achieve my goal? I have implemented three neural network models, one of which is convolution network model and the other two are full connection network models. For the convolution network model, my data processing method is to save the mfcc(Mel-Frequency Cepstral Coefficients) data of each audio file into a json file and also save the label of each mfcc data. After testing the trained model, I found that the accuracy is not very ideal. So I started to deal with the csv file already provided in the data file. This file is the mean and variance of the audio file obtained by feature extraction, but it splits each 30-second audio file into ten copies, which can greatly increase our original data. For the training of these two fully connected networks, I found that the simpler neural network model not only trains faster, but also has higher accuracy under the condition of better data preprocessing.

There are several parts to show the detailed processes including a short introduction, some related works to show what's motivation to push me to achieve this work in this way, some detailed description of the dataset used, preprocessing I applied, hyperparameter tuning done etc. . . ., a result about evaluation about my models and a conclusion to show what I find during the work.

2 Related work

Generally speaking, the common method to solve the audio classification problem is to preprocess the audio input to extract useful features, and then apply the classification algorithm to it. For example, in a case study, it gives a 5-second sound transcription, and requires a classifier/neural network to determine which kind of sound belongs to-barking or drilling. The proposed solution

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

is to extract an audio feature named MFCC, and then use neural network to find the appropriate category.

However, I found that in the past work, mfcc data were usually extracted from audio files, and only a simple neural network model was established, which led to a low accuracy after model training. So after I think about it, I think there are three ways to improve the status quo.

- They only applied a simple neural network model to the problem. Our immediate next step should be to understand where does the model fail and why. By this, we want to conceptualize our understanding of the failures of algorithm so that the next time we build a model, it does not do the same mistakes.

- We can build more efficient models that our "better models", such as convolutional neural networks or recurrent neural networks. These models have be proven to solve such problems with greater ease.

- I knew the concept of data augmentation. We could try it to see if it works for the problem.

Therefore, according to the above three ideas, I tried to make some substantial improvement in my project. I used the convolutional neural network model for training, and divided each audio data into five parts to save each mfcc data, so as to achieve the purpose of data expansion through this operation. However, I found that although the accuracy rate has improved, it is still not ideal. Part of the reason may be that the training epoch I set is not big enough. However, I immediately processed the csv file of the mean and variance of the data after feature extraction, and trained it through the fully connected neural network, and found that the results obtained were very impressive.

3 Experimental Setup

In this part, I will describe in detail some information of the data I used and how I preprocessed the data. There are also some basic methods that I have implemented and some hyperparameters adjustments to the model.

3.1 dataset and preprocessing

Firstly, I picked one of audio files to show its Zoomed audio wave graph, Simple audio waveplot and Principal component analysis on Music Genres graph amongest ten different labels.

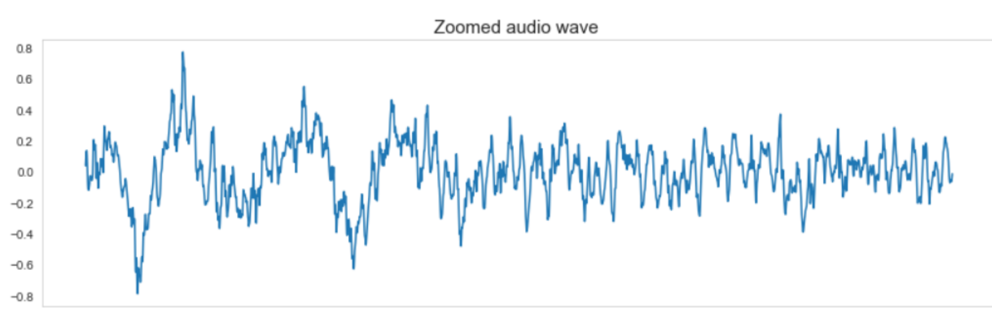


Figure 1: zoomed_audio_wave

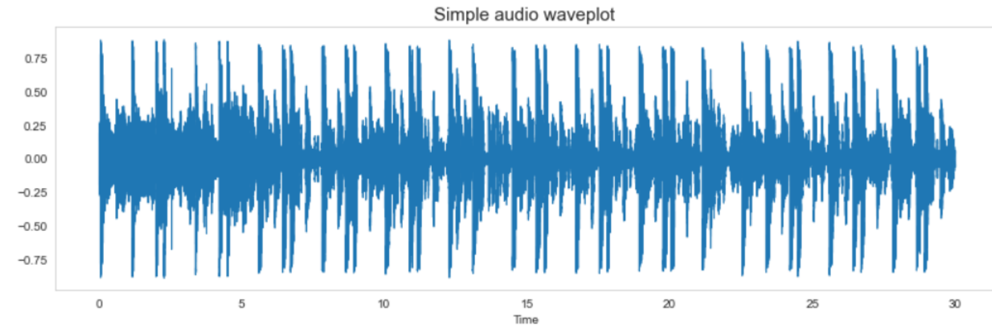


Figure 2: simple_audio_waveplot

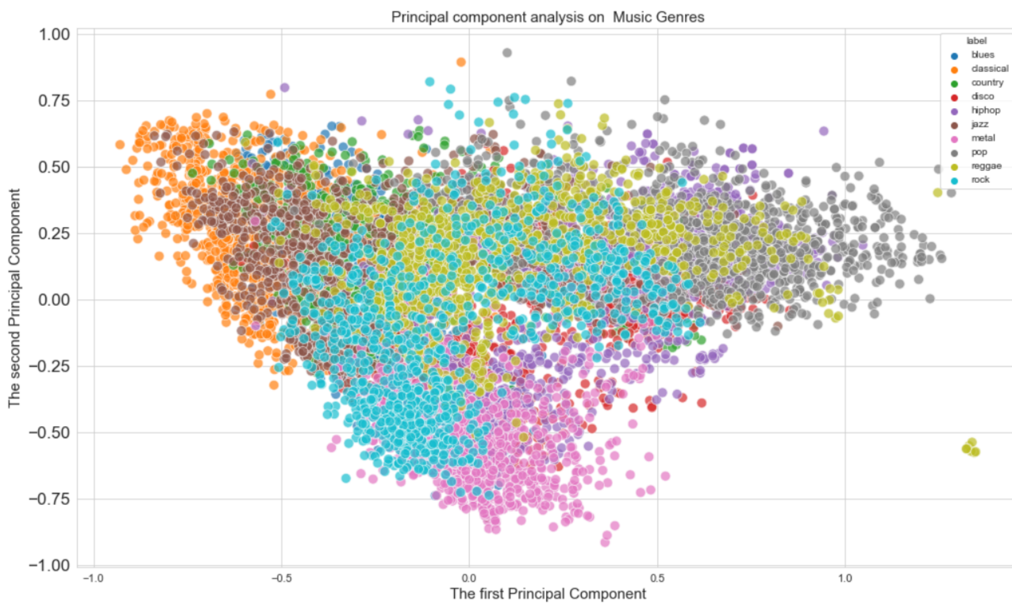


Figure 3: pca

Here, we have done a series of data analysis and processing, and we have done different processing for different models.

First of all, for convolutional neural network, I refer to some resources on the network to learn if I get mfcc data of audio files,

```
https://www.analyticsvidhya.com/blog/2017/08/
audio-voice-processing-deep-learning/
```

and in order to deal with the problem of insufficient data, I divide each audio file into five parts to save mfcc data. Then I saved all kinds of music categories, all mfcc data and their corresponding labels in a json file. Then we only need to read the json file, and I divided the data into 60% training data, 15% valid data and 25% test data.

For the data processing of fully connected neural network, it is to analyze and preprocess the data of csv file. Here, you can see some detailed analysis of the data in the file in the readme file, including querying how many rows of data are in each category, deleting some unrelated columns, and replacing the label with a simple digital label.I also deal with the missing values using "df.fillna(0)"(there aren't missing values). Then we shuffle all the data and divide them into 70% training data, 20%

valid data and 20% test data. And save them to three different csv files for later operation on the model.

3.2 models building

In this project, three neural network models are built.

One of them is convolutional neural network. pooling layer is added after the convolution layer of the first three layers to keep the properties and do reduction parameters of data features, and normalization operation is added after the pooling layer to unify the scattered data. Before the data is transmitted to the full connection layer, we add flatten layer to reduce the dimension, and then after the operation of dropout, we connect a full connection layer again, and the softmax activation function of this layer outputs the final desired result.

For the other two fully connected network models, one of which is a simple four-layer fully connected network, the final prediction result is obtained through the softmax activation function of the last layer. In the other model, I used a total of six-layer of higher-dimensional fully connected network, but after each layer, I added dropout layer to prevent over-fitting.

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 214, 11, 32)	320
max_pooling2d (MaxPooling2D)	(None, 107, 6, 32)	0
batch_normalization (BatchNo	(None, 107, 6, 32)	128
conv2d_1 (Conv2D)	(None, 105, 4, 32)	9248
max_pooling2d_1 (MaxPooling2	(None, 53, 2, 32)	0
batch_normalization_1 (Batch	(None, 53, 2, 32)	128
conv2d_2 (Conv2D)	(None, 52, 1, 32)	4128
max_pooling2d_2 (MaxPooling2	(None, 26, 1, 32)	0
batch_normalization_2 (Batch	(None, 26, 1, 32)	128
flatten (Flatten)	(None, 832)	0
dense (Dense)	(None, 64)	53312
dropout (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 10)	650

Total params: 68,042
Trainable params: 67,850
Non-trainable params: 192

(a) cnn_model_structure

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 1024)	59392
dropout (Dropout)	(None, 1024)	0
dense_1 (Dense)	(None, 512)	524800
dropout_1 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 256)	131328
dropout_2 (Dropout)	(None, 256)	0
dense_3 (Dense)	(None, 128)	32896
dropout_3 (Dropout)	(None, 128)	0
dense_4 (Dense)	(None, 64)	8256
dropout_4 (Dropout)	(None, 64)	0
dense_5 (Dense)	(None, 10)	650

(b) fcnn2_model_structure

Figure 4: The structure for two different models

3.3 configuration and model training

Before we start training the model, we need to do some configuration work to debug the best training environment. In this project, I use YAML file to save the hyperparameters of each model, including the path to save models and logs, different optimizer algorithms used for different models, the size of batch, and the number of training times(30 epochs for cnn model,70 epochs for fcnn1 model, 500 epochs for fcnn2 model). After that, if tensorflow wants to train the model on gpu, it needs to configure CUDA and cuDNN. Note: We need to install the version that strictly abides by python, keras, tensorflow, CUDA and cuDNN, because there are strict dependencies among them.

python3.6.0
keras2.3.1
tensorflow-gpu2.0.0
CUDA10.0.0
cuDNN7.6.0.64 for CUDA10.0.0

Figure 5: packages_version

Then I began to train the model (based on the hyperparameters read from YMAL file), and draw and saved the graphs about training and validation's loss and accuracy. While saving the model to the specified path, we also save some basic data of the model (such as the number of rounds with the highest accuracy in the training process) into json file, so that we can test the model later.

As for the results of each model after training, the convolution neural network model only achieves 67% accuracy, the first fully connected neural network model achieves 88% accuracy, and the last fully connected neural network model achieves 93% accuracy, which is very impressive.

4 Results

For the evaluation part of the model, I referred to the sample prject, I used the best epoch which we got during the models' training with higest accuracy, and each time I trained it until the epoch saved by our previous training. After testing the model with the test set, the data obtained will be saved, and then the average accuracy of the model and the fluctuation range of the accuracy will be obtained by using np.sqrt 1.

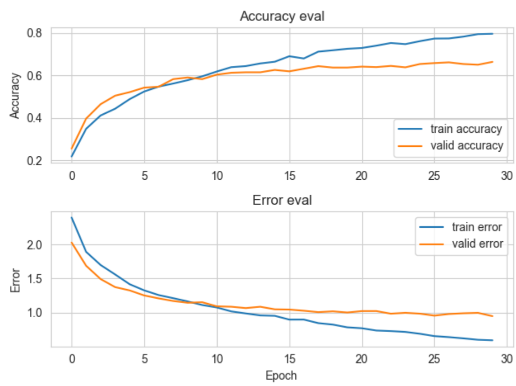
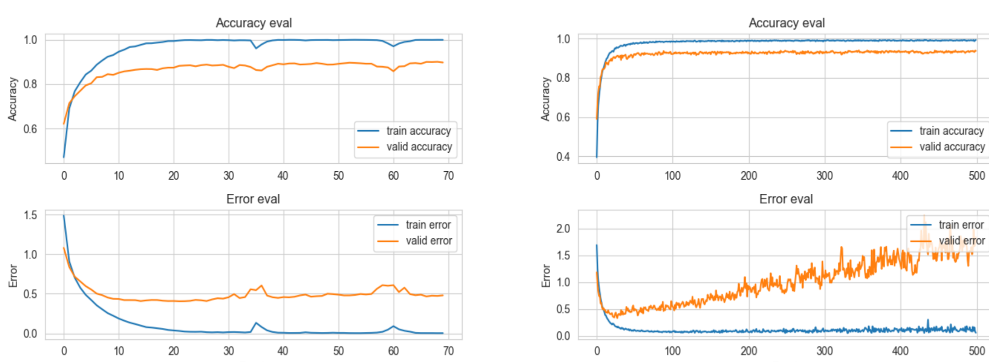


Figure 6: cnn1_training_vis



(a) fcnn1_training_vis

(b) fcnn1_training_vis

Figure 7: The figures for two different models' training result

Table 1: The evaluation results for three models

Index	model	accuracy
0	cnn1	0.66± 0.0110
1	fcnn1	0.89± 0.0087
2	fcnn2	0.93± 0.0039

5 conclusion and future work

For the experience of this project, I learned a lot about deep learning and gained a great of knowledge about how to build tensorflow structure, including data processing, building, training and evaluation of deep neural network model. The following are some experiences that I found and summarized during my study:

- Correct processing of data and correct selection of models are important ways to improve the accuracy of results.
- Some hyperparameter of reasonable debugging model are also an important way to improve the accuracy of results.
- Having a full knowledge of previous and related work can greatly improve our work progress.
- Choosing an appropriate method to evaluate the model is an important point of evaluation the model.
- Different methods are equally important for data processing, building different neural network models and comparing the results.

Of course, in the limited time, there are still some shortcomings in this completed project. If there is a chance to rewrite this project, I will:

- Build some more complicated models to try to improve the accuracy of the results.And choose more methods to evaluation the model.
- I try to extract the data by myself, and save the mean and variance of the data in csv file.
- I will also try to use some different optimization functions and batch sizes to understand their effects on the results.
- When the hardware conditions permit, I will increase the epoch of model training, so as to obtain the best situation of model results.

References

[1] Jiquan Ngiam, Aditya Khosla, Mingyu Kim. (2011) Multimodal Deep Learning. In G. Tesauro, D.S. Touretzky and T.K. Leen (eds.), *Advances in Neural Information Processing Systems 7*, pp. 609–616. Cambridge, MA: MIT Press.

[2] Wei Han, Cheong-Fat Chan, Chiu-Sing Choy and Kong-Pang Pun, "An efficient MFCC extraction method in speech recognition," (2006) *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2006, pp. 4 pp.-, doi: 10.1109/ISCAS.2006.1692543.

[3] T. Kim, J. Lee and J. Nam, "Comparison and Analysis of SampleCNN Architectures for Audio Classification," (2019),*in IEEE Journal of Selected Topics in Signal Processing*, vol. 13, no. 2, pp. 285-297, May 2019, doi: 10.1109/JSTSP.2019.2909479.