Main Process

MLPClassifier

At first, I use the `MLPClassifier` provided in template and add `bagging`. But it cost too much time and the accuracy was not good. For each training, I divided the provided dataset into training data (90%) and validation set (10%)

Neural Network

During the next 5 days of my training, I attempted to train a neural network whose hidden layers are $12000, 4000, 500, 500$ in size respectively, and all apply `leaky_relu` activation function. For training this model I used the provided dataset in `bof/`, applied SGD optimizer, set 1×10^{-3} as learning rate and 300 as number of epochs. To speed up the training process and facilitate customization, I used `Pytorch` library. One of reasons why I chose this library, these parameters and structures, is that I can customize the model easily and use GPU to speed up training. Another reason is that I have get familiar with this library with the help of UBRTP. The training result is not good enough. This model meets a problem of severe overfitting whatever parameters I used and could not increase the score.

RNN

Then I asked for some suggestions for increasing my accuracy. Someone told me that training an RNN with the MFCC sequences can be effective then But because of some unknown mistake, I could not get high enough accuracy. Because of the high time consumption for training, I gave up this model eventually.

CNN

On the day before the deadline, occasionally, I notice a blog which says CNN can also be trained to handle the MFCC sequences. But I cannot find the blog again. Then I built up a convolution neural network, which is very similar to the ones used for image classification. Then I trained the model using (nearly) same code of RNN. Overfitting still exists. The default parameters in `train_cnn.py` are used for my best model. Although the same parameters are used, the final training results will fluctuate significantly due to the strategy of random data set division. In the end, I chose a model with the highest accuracy on the validation set as the result.

Files and Reproduction

`cnn.py`: model definition and main process of training.

`train_cnn.py`: preparation of training data and parameters.

`test_cnn.py`: generation of prediction for test data.

`model_structure.png`: overview of structure of my model

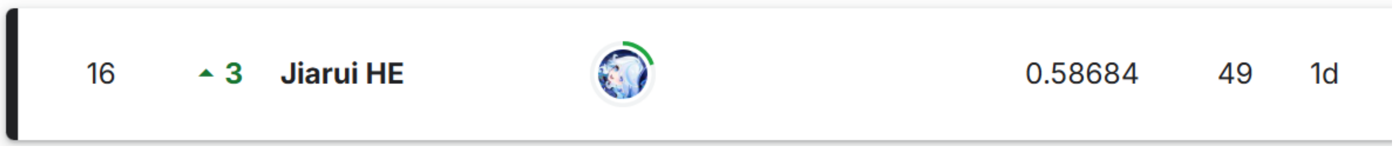
Command for training:

```
python3 train_cnn.py mfcc/ 39 labels/trainval.csv cnn.model cnn-1
```

Command for testing:

```
python3 test_cnn.py cnn.model mfcc/ 39 labels/test_for_student.label mfcc-cnn.csv
```

Screenshot



16	▲ 3	Jiarui HE	0.58684	49	1d