2022R1 Advance Topics in AI (CMSC5707)
Assignment 3 Report

LI Yuxin
1155107874
ryanli@link.cuhk.edu.hk

1. Enviorment Settings

All training tasks in this assignment are run on a MacBook Pro with CPU 2.6 GHz 6-Core Intel Core i7 and MacOS Ventura 13.0. The model is implemented by Tensorflow with version 2.11.0.

2. Demo Reproduction

Fixed Issues The demo downloaded from https://github.com/ruohoruotsi/LSTM-Music-Genre-Classification can't be executed directly. There is some import issues. First,

```
from keras.layers.recurrent import LSTM
```

should be changed to

```
from keras.layers import LSTM
```

And the function $Path$ also misses it reference. So, add another imported package as follows:

```
from pathlib import Path
```

Demo Training and output I train the LSTM music genre classifier using the default model settings and hyperparameters.

```
Validating ...
1/1 [==============================] - 1s 561ms/step - loss: 0.4973 - accuracy: 0.8400
Dev loss:   0.4973221719264984
Dev accuracy:   0.8399999737739563

Testing ...
1/1 [==============================] - 0s 41ms/step - loss: 0.4141 - accuracy: 0.8800
Test loss:   0.4141201674938202
Test accuracy:   0.8799999952316284
```

Figure 1. Demo Execution Result

The model obtains a high accuracy and low loss after training and performs quite well in testing.

3. Speech Recognition System Benchmark

Based on the LSTM model I was just farmiliar with, I extend the model to build a speech recognition system.

Data I record audios for five names (i.e., Peter, Jelly, Billy, Alice, Jackson) for each 22 times. Each audio lasts 3.065 seconds with 24kHz-16bit and is in $.au$ format. Note that $Billy$ and $Jelly$ are similar in pronunciations. These two names are aims to add some challenges in recognition processes. For each name, the 22 audios are split into 3 groups, No.0-9 for training, No.10-14 for testing, No.15-19 for validation, No.20-21 for prediction.

Benchmark Model Audio data are subsampled with a length of 128 and then converted to MFCCs. All MFCCs data (i.e., 13) are inputed into two LSTM layers with distinct hidden size(i.e, one is 128, the other is 32). The model trains 400 epochs.

Benchmark Output The benchmark model has testing accuracy of 0.64. Figure 3 shows one of it successful prediction.

```
Validating ...

1/1 [==============================]
1/1 [==============================]
Dev loss:   0.538504421710968
Dev accuracy:   0.7599999904632568

Testing ...

1/1 [==============================]
1/1 [==============================]
Test loss:   0.6743209362030029
Test accuracy:   0.6399999856948853
```

Figure 2. Benchmark model training output

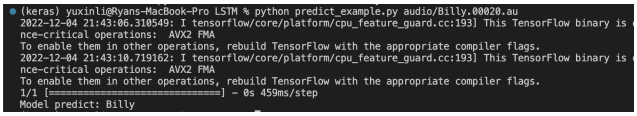

Figure 3. Benchmark model successfully predicts $Billy$

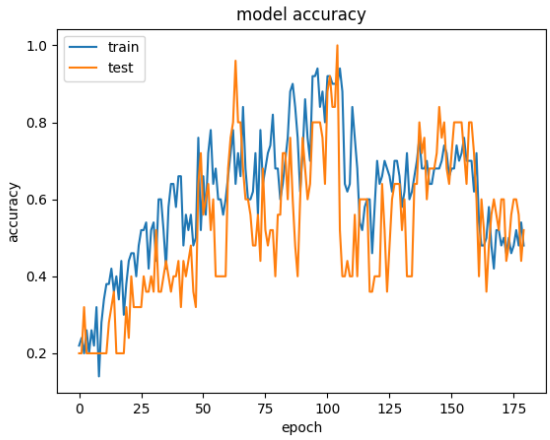

Figure 4. Benchmark Accuracy

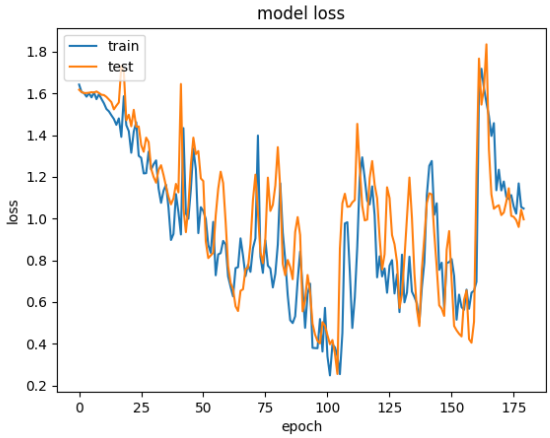

Figure 5. Benchmark Loss

4. My Model

Design Idea The benchmark model doesn't perform good enough. Some changes may potentially improve its ability. One is MFCC, as we all known, the first element in MFCC isn't related to frequence, so it may not help the model. Second, the model subsamples audio signal with each length of 128. Since the length of audio is 3.065 and the model processes them in 22050Hz, the total number of samples is $22050 \times 30.65 = 675832$, frames with length of 128 is too short so that some global information or long term information will lost naturally. So, I consider raising sample length. Since the sample length is changed, the model architecture is also changed. The first LSTM layer should have more hidden states to fit the input and more layers are required to extract information from additional data. The last motivation is overfitting, I monitor the accuray and loss during benchmark experiemnt, I found the model shows overfitting where validation loss increases and validation accuracy decreases at epoch around 100. So, shorter training times will be applied.

Implementation Details I raise sample length from 128 to 256. The hidden size of the first LSTM layer is changed to 256 correspondingly. One additional LSTM layer with hidden size 128 is added to the second layer to extract information from a long sample. Then apply a LSTM layer with hidden size of 32 to finally refine characteristic from data. With my observation and many experiments, I found that MFCC0 actually have some contributions to accuracy, so I didn't remove them. The overfitting of my model is also detected around epoch=270. So, I stop training at epoch=270

Output Originally my model is trained in 400 epochs, many points (e.g., 140, 200, 270, etc.)show potential overfitting. After many experiments, the point around 270 is verified as the point where my model should stop training. Finally, my model has testing accuracy of 0.84.

```
Validating ...
1/1 [==============================]
Dev loss:   0.37569454312324524
Dev accuracy:   0.800000011920929

Testing ...
1/1 [==============================]
Test loss:   0.39109718799591064
Test accuracy:   0.8399999737739563
```

Figure 6. My Model training output

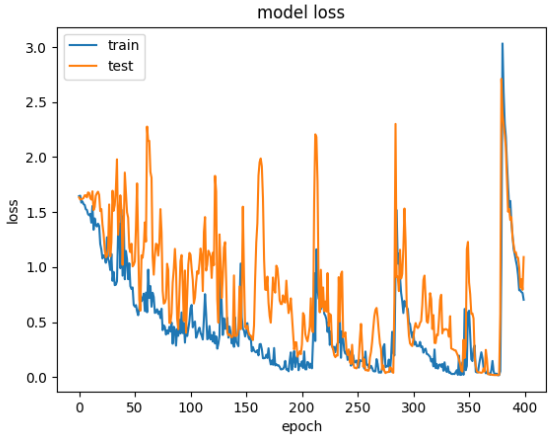

Figure 7. My model trains in 400 epochs, overfitting can be detected around epoch = 270

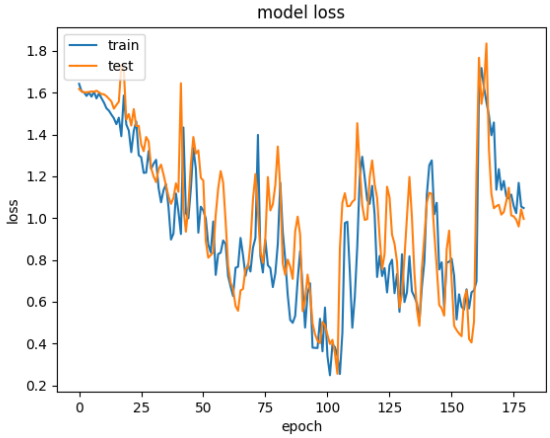

Figure 8. My Model Accuracy

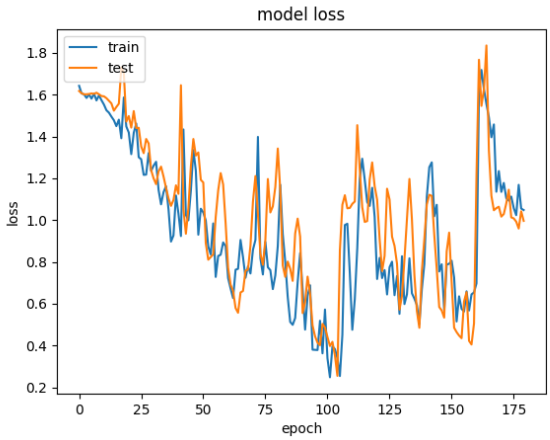

Figure 9. My Model Loss