

Frame level speech recognition

Yi Yuan

Introduction

With the birth of Siri in 2005, the public's discussion and interest in speech recognition have exploded. Even though the existing artificial intelligence assistants are still far from reaching the level of intelligence, people still give great hope that speech recognition can revolutionize people's lives. This project aims to perform frame level speech recognition on a dataset of audio recordings and their phoneme state labels.

This dataset is the audio recording of some article which published on Wall Street Journal. And the raw audio data have been transformed to Mel spectrogram. The labels are 71 possible phonemes (just like 26 letters). Regarding to the letters which is the atomic elements of the written language, the phonemes is the atomic elements of speech. In phonetics and linguistics, a phoneme is a sound unit that can distinguish one word from another in a specific language. This data has 3 dimensions, the 1st dimension is the number of utterances, the 2nd dimension is the number of frames in each utterance, and the 3rd dimension is the feature length. The feature length (the 3rd dimension) equals to 40 which is fixed and represent the 40 different frequencies in Mel scaled. The phonemes labels correspond frame level Mel spectrogram pieces.

I chose feedforward neural network to recognize this audio dataset. The challenge of this project is how to deal with 3-dimensional data and how to use phonetic characteristics to improve the accuracy of recognition. How to tuning and regularize neural network. As result, the final model gains 77.85% accuracy in validation set.

Implementation

In order to improve the accuracy of the model and make full use of the characteristics of phonetics. I used the idea of n-gram to perform feature engineering on data. The reason is that when we speak, the pronunciation of some words may be weak or unclear, but through the context, we can infer the correct words. Originally, the one frame Mel spectrogram pieces corresponds to one label (phoneme). After feature engineering, our input becomes one Mel spectrogram frame plus $2*k$ neighbor frames. I mean that in total $2*k+1$ frames are expected to corresponds to one label.



Figure 1 The idea of context

Obviously, there is a problem with the start and the end of one utterance. The first frame of one utterance doesn't have previous neighbor, and the end frame doesn't have any frame after it. To solve this problem, I pad these utterances with zero. Zero means silence, so they contribution nothing to prediction, in the meantime we can maintain the data consistency.

After checking the data, I found that each utterance has a lot of zeros at the beginning and end. The reason is that in reality our recordings always unavoidably have silence at the beginning

and end. Therefore, I can concatenate all the raw utterances and only pad the beginning and the end of this huge utterance with k-frames zeros.

```
In [72]: 1 y = np.load(y_train_path,allow_pickle=True)
2         with np.printoptions(threshold=np.inf):
3             print(y[155])
```

```
[ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
  0.  0.  0.  0.  0. 20. 20. 20. 20. 20.  9.  9.  9.  9. 58. 58. 58. 58.
 58. 58. 58. 67. 67. 67. 67. 67. 67. 67. 37. 37. 37. 37. 46. 46. 46. 35.
 35. 35. 40. 40. 40. 40. 40. 40. 40. 40. 43. 43. 43. 43. 43. 43. 43. 43.
 43. 43. 31. 31. 31. 31. 31. 31. 31. 31. 31. 31. 31. 31. 31. 31. 31. 45. 45.
 45. 45. 45. 45. 45. 45. 45. 45. 58. 58. 58. 58. 58. 58. 58. 58. 58. 58.
 64. 64. 64. 64. 44. 44. 44. 44. 44. 44. 44. 44. 44. 44. 44. 44. 44. 44. 61.
 61. 61. 61. 61. 61. 61. 61. 61. 61. 61. 61. 61. 61. 61. 61. 61. 61. 61.
 61. 61. 61. 43. 43. 43. 43. 43. 43. 43. 43. 43. 43. 43. 43. 43. 43. 43.
 43. 43.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 23. 23. 23. 23. 23. 23. 23. 25. 25. 25. 25. 55. 55. 55. 55. 55. 55. 55. 67.
 67. 67. 67. 67. 67.  8.  8.  8.  8. 69. 69. 69. 69. 69. 69. 69. 69. 69.
 69. 46. 46. 46. 46. 46. 46. 49. 49. 49. 49. 49. 49. 49. 49. 49. 49. 43. 43.
 43. 43. 43. 43. 43. 43. 43. 43. 43. 43. 40. 40. 40. 40. 40. 40. 40. 40.
 40. 40. 35. 35. 35. 35. 35. 35. 35. 35. 35. 35. 49. 49. 49. 49. 49. 49.
 49. 49. 49. 49. 44. 44. 44. 44. 44. 44. 44. 44. 44. 58. 58. 58. 58. 58. 58.
 58. 58. 64. 64. 64. 64. 64. 64. 64. 64. 64. 64. 64. 64. 64. 36. 36. 36.
 36. 36. 36. 36. 36. 36. 58. 58. 58. 58. 58. 58. 58. 58. 58.  0.  0.  0.  0.
  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
```

Figure 2 Silence on the beginning and the end

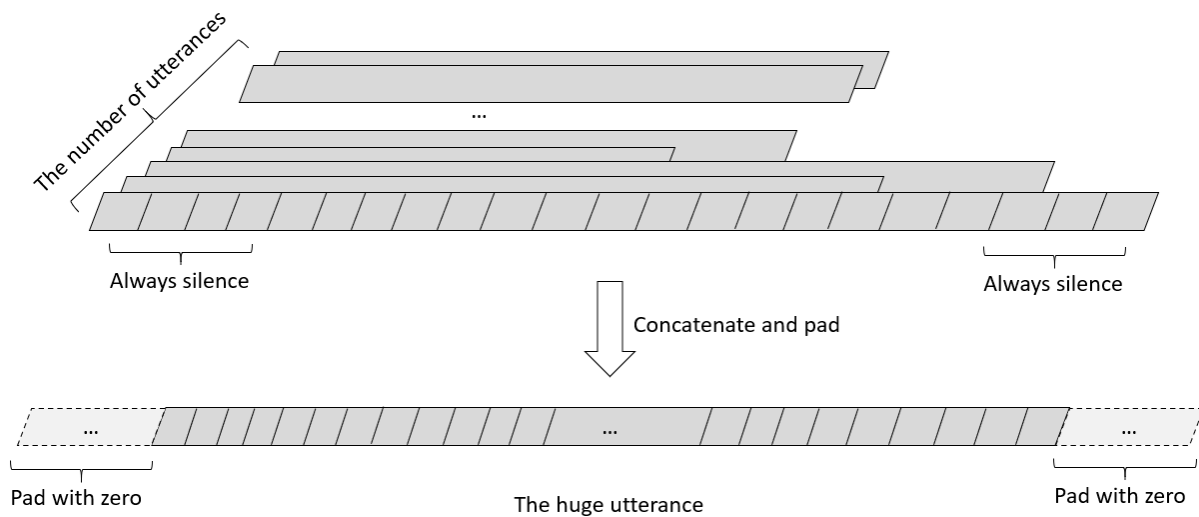


Figure 3 Padding

The MLPs is a universal classifier so it can be used for this task. Due to the training data is too large and it will take tons of time to tuning model on training data. I did a pre-modeling test on a smaller set(validation set). The final structure shows as follows:

```

CreateModel(
    (net): Sequential(
      (0): Linear(in_features=1240, out_features=2048, bias=True)
      (1): Dropout(p=0.3, inplace=False)
      (2): ReLU()
      (3): BatchNorm1d(2048, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (4): Linear(in_features=2048, out_features=2048, bias=True)
      (5): Dropout(p=0.3, inplace=False)
      (6): ReLU()
      (7): BatchNorm1d(2048, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (8): Linear(in_features=2048, out_features=1024, bias=True)
      (9): Dropout(p=0.3, inplace=False)
      (10): ReLU()
      (11): BatchNorm1d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (12): Linear(in_features=1024, out_features=512, bias=True)
      (13): Dropout(p=0.3, inplace=False)
      (14): ReLU()
      (15): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (16): Linear(in_features=512, out_features=256, bias=True)
      (17): Dropout(p=0.3, inplace=False)
      (18): ReLU()
      (19): BatchNorm1d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (20): Linear(in_features=256, out_features=71, bias=True)
    )
)

```

Figure 4 Model Architecture

Table 1 Hyperparameters for baseline model

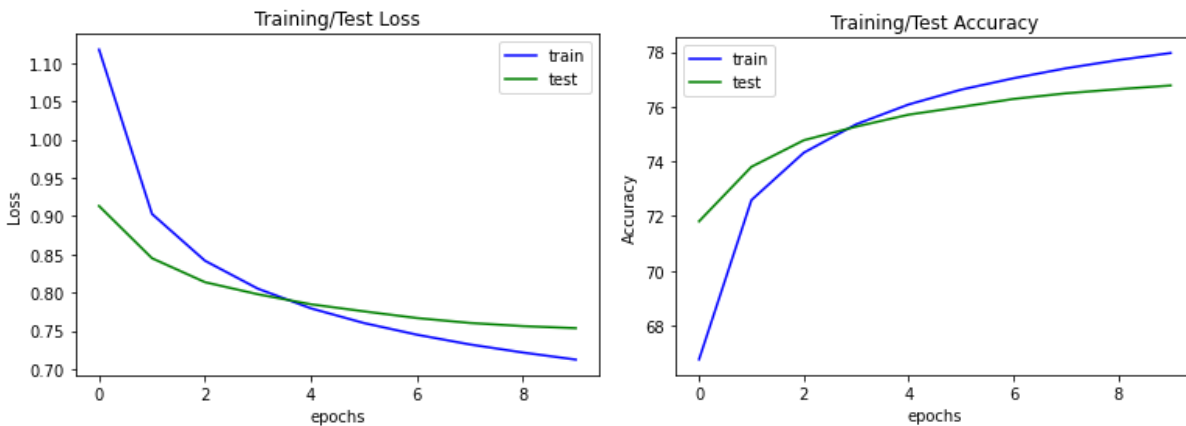
Some hyperparameters of baseline model	
Layers and nodes	[40*(2*context+1),2048,2048,1024,512,256,71]
Optimizer	“Adam”
Activation function	ReLU()
Regularization	Dropout(p = 0.3)
Normalization	Batch_norm
Batch sizes	4096
Criterion	CrossEntropyLoss()
Number of Epochs	10

Next, I tuned the hyperparameter: context, Adam learning rate on the training set.

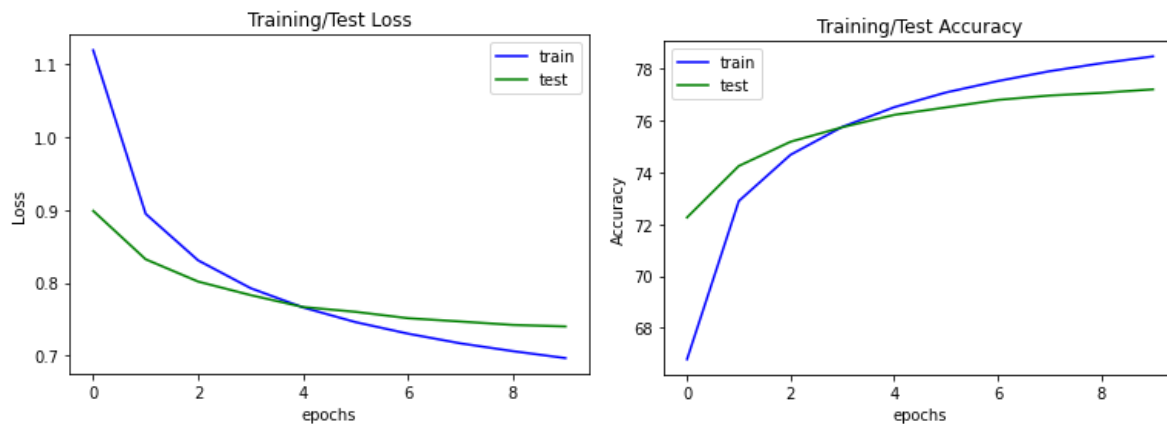
Data Analysis

This project is a little bit more useful than handwritten digits recognition task, and it also requires lots of knowledge to training. In data analysis, I want to compare the predict results of different combination of hyperparameters.

The results are pretty good. For the first model which I set context equals to 20 and learning rate equal to 0.001. The accuracy reaches 76.78%. The following plots show the training process is pretty smooth, I think this may be benefited from regularization and normalization. However, it still not reaches the best score. So, I decided to increase the number of context.

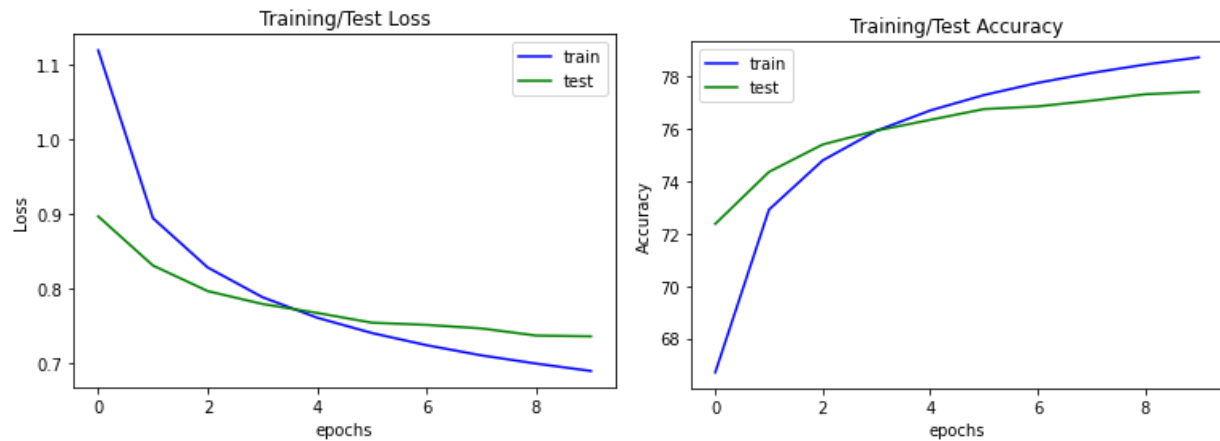


The second model, I adjust the context from 20 to 25. And the results show as following. This result is not too much different from the previous one. But its accuracy increased to 77.19%. I think perhaps I should increase the learning rate to get convergence faster.



For my third trial, I continues increase the number of context to 25 and I increased the learning rate of “Adam” to 0.002. So, I got 77.40% accuracy on testing set. So, this indicated a

bigger number of contexts is not equals to a better performance. However, training needs time.





















Neural network could be easily overfitting because there are thousands of parameters in it. The model is complex. The plot shows that with the epochs increasing the training loss always decrease. So, regularization is very important for neural network.

The final model may not the best model. I save model for each epoch because sometimes performance may decrease after a certain point.

> This PC > Documents > ALY 6020 > group_project > final

Search final

Name	Date modified	Type	Size
 model_V1_1.pt	2/24/2021 7:55 PM	PT File	37,258 KB
 model_V1_2.pt	2/24/2021 8:33 PM	PT File	37,258 KB
 model_V1_3.pt	2/24/2021 9:11 PM	PT File	37,258 KB
 model_V1_4.pt	2/24/2021 9:49 PM	PT File	37,258 KB
 model_V1_5.pt	2/24/2021 10:26 PM	PT File	37,258 KB
 model_V1_6.pt	2/24/2021 11:04 PM	PT File	37,258 KB
 model_V1_7.pt	2/24/2021 11:44 PM	PT File	37,258 KB
 model_V1_8.pt	2/25/2021 12:23 AM	PT File	37,258 KB
 model_V1_9.pt	2/25/2021 1:01 AM	PT File	37,258 KB
 model_V2_0.pt	2/25/2021 1:55 AM	PT File	40,458 KB
 model_V2_1.pt	2/25/2021 2:34 AM	PT File	40,458 KB
 model_V2_2.pt	2/25/2021 3:12 AM	PT File	40,458 KB
 model_V2_3.pt	2/25/2021 3:51 AM	PT File	40,458 KB
 model_V2_4.pt	2/25/2021 4:30 AM	PT File	40,458 KB
 model_V2_5.pt	2/25/2021 5:09 AM	PT File	40,458 KB
 model_V2_6.pt	2/25/2021 5:48 AM	PT File	40,458 KB
 model_V2_7.pt	2/25/2021 6:27 AM	PT File	40,458 KB
 model_V2_8.pt	2/25/2021 7:06 AM	PT File	40,458 KB

1 selected

36.3 MB

More layers and nodes do not mean a better performance, but always a longer training time. Therefore, it is very important to pretrain your model on a relatively small dataset. It could save tons of time.

Discussion

I think this project is very suitable for using neural networks for the following reasons: First, this data is very large. Using traditional machine learning algorithms, we may need to train the entire batch at once. However, the neural network supports the online method. Through continuous iteration of the mini batch, the final result is relatively excellent accuracy. Second, pytorch supports custom dataset, which greatly facilitates my data engineering. In this project, the k value(context) is the key to improve the accuracy of the model. I tuned this hyperparameter

without reconstruct several huge data matrices. Third, neural networks usually support using GPUs to do the calculations, which can significantly improve training speed.

From the viewpoint of analytics, I think preform “n-grams” idea in this project make it interesting. I found there always silence at the beginning and end of the recording that also save tons of padding operation. These key factors are not from model but the analytic and the common senses.

From this project, I put a lot of neural networks knowledge into practice. Through continuous tuning, I deepen my understanding of various hyperparameters. Finally, I am proud that I can complete such a project independently.

References

Haytham Fayek. (2016, April 21). Speech processing for Machine learning: FILTER banks, mel-frequency Cepstral Coefficients (MFCCs) and What's In-Between. Retrieved February 25, 2021, from <https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html>

11785-spring2021-hw1p2. Retrieved February 25, 2021, from <https://www.kaggle.com/c/11785-spring2021-hw1p2/leaderboard>