Deep Learning Spring 2020
Assignment 3

Minghao Zhang

1 Basic Task

I used **RNN, GRU, LSTM** to train for thiss task. Empirical result shows that **GRU** with bigger batch size outperforms the other models and settings. The following curves are the baselines for this assignment:

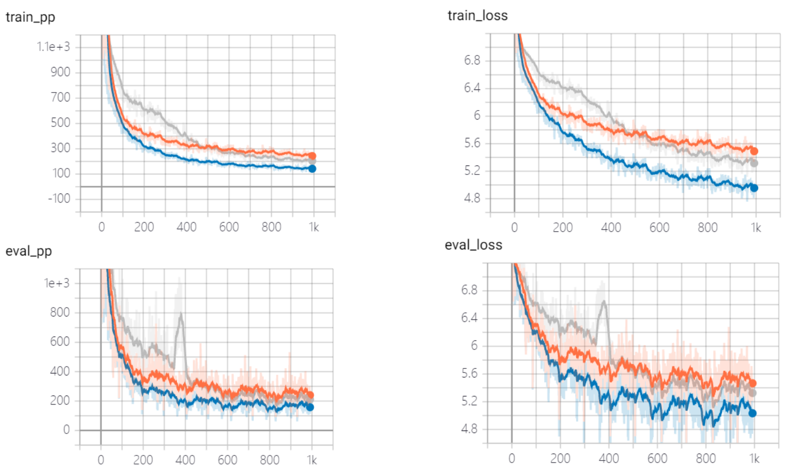

Figure 1: RNNRNN: Grey, LSTM: Orange, GRU: Blue

2 Bidirectional Setting

If simply turning on the *bidirectional* of RNNs, the performance will be improved a lot. But since this is a prediction task, we shouldn't let this setting happen, which means your have already given the prediction ground truth to the model.
However, in my opinion, if we mask the needed part we don't want to see in the reversed direction, there is no meaning of this auxiliary process, i.e., there is no reasonable explanation for adding masks, doing bidirectional inference and training it for a **predictive** task.

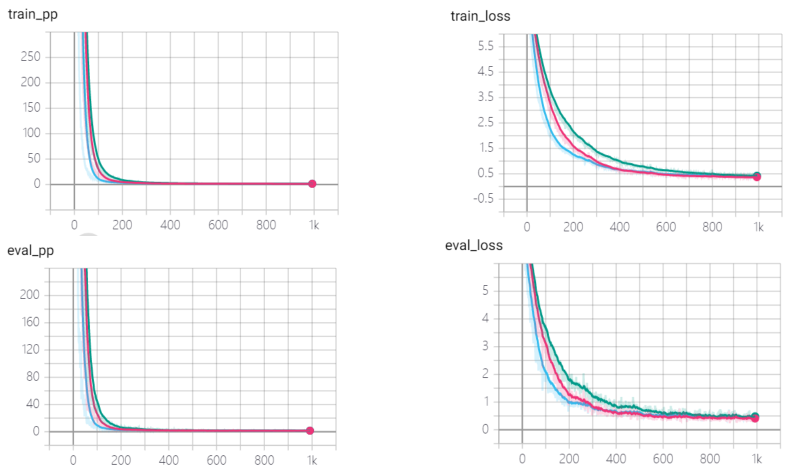

Figure 2: Bidirectional RNN.RNN: Green, LSTM: Pink, GRU: Blue

3 Attention

I used temporal attention mechanism and self attention mechanism to try to improve the performance but **failed**. But I have the following insight to explain: better than convolutional-based temporal attention. Note that this task is to predict the future item. From this perspective, we must make sure that the former part of sequence can not pay attention to the latter one. Besides, the output of model is a sequence, so it is better to use point-wise product to get the sequential output. So the overall influence of the attention is not so different from linear layer. Last but not least, as the dataset is not so big, the complex model will be easy to overfit. So the attention module may not be a proper choice for this task.

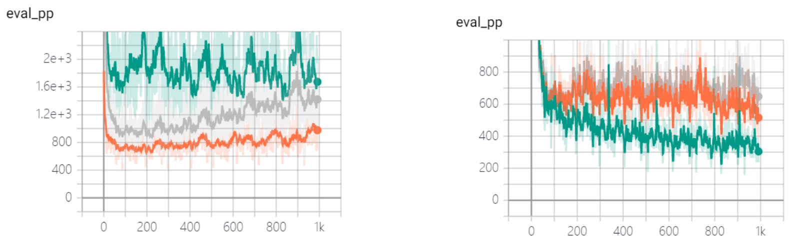

Figure 3: Left: RNNs with temporal attention. Right: RNNs with self attention. From top to bottom: RNN, LSTM, GRU

4 More Tricks

To improve the results, I tried a lot of tricks. The following ones worked well:

- Bigger batch size. The bigger training batch size leads to the lower eval pp value, before reaching about 200. The size of 150 worked greatly.
- Orthogonal initialization gave some improvement to the result. The initialization method always matters for deep learning model.
- Adam optimizer and bias in RNNs both make it better.

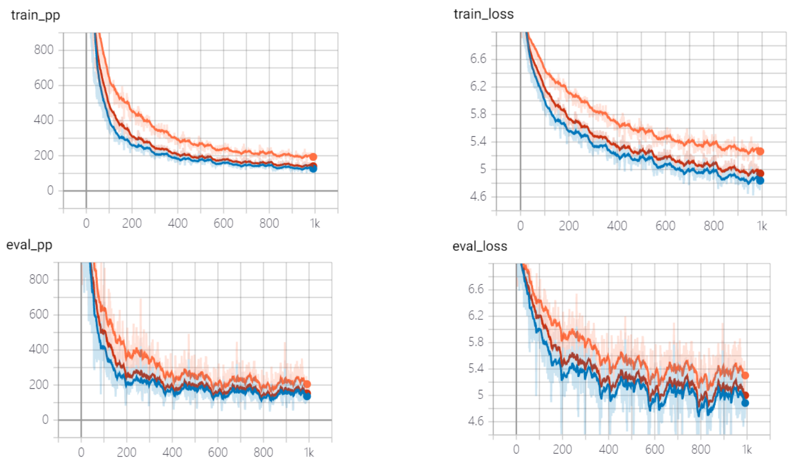

Figure 4: RNN with more tricks. RNN: Orange, LSTM: Red, GRU: Blue

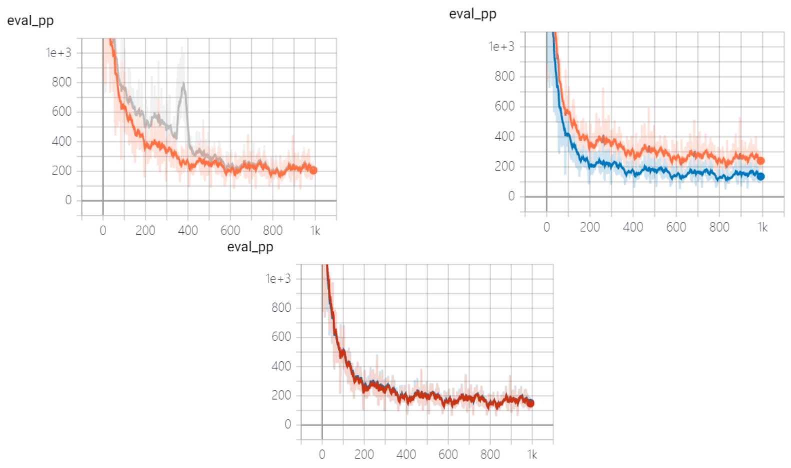

Figure 5: RNN with more tricks (comparison). The listed ones are results of RNN, GRU, LSTM

5 Bonus

I implemented RNN's forward and backward module. Simply run the "main" module in "myrnn.py" can show the example's result. I refer the details to the code.