ECGR 5106 – Real Time Machine Learning (Spring 2023)

Nahush D. Tambe – 801060297

https://github.com/Ntambe25

Homework # 5

Problem 1:

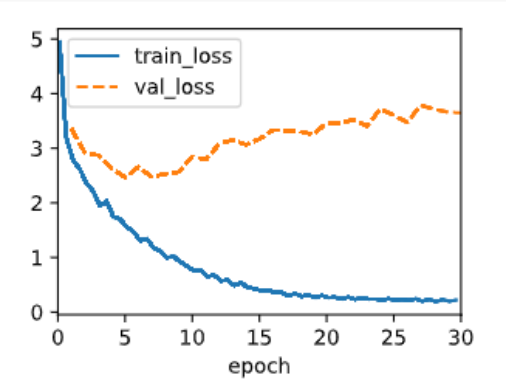


Figure 1: Results of Baseline Seq2Seq Model for Machine Translation

For Problem # 1, Part 1, the baseline model was first trained. The hyperparameters were kept the same as in the d2l book and the lecture notes. Figure 1 above shows the results for the same.

The training loss starts at around 5 and goes down to almost 0, and on the other hand, the validation loss starts at about 3.5 and goes to about 3.75 to 4.0. Thus, it shows that the model is not great for generalized data as the generalization gap present is quite high and the validation loss shows a upwards trend. The hyperparameters for this model were as follows: embedding size and number of hidden layer size of 256, number of hidden layers is 2 and a dropout probability of 0.2.

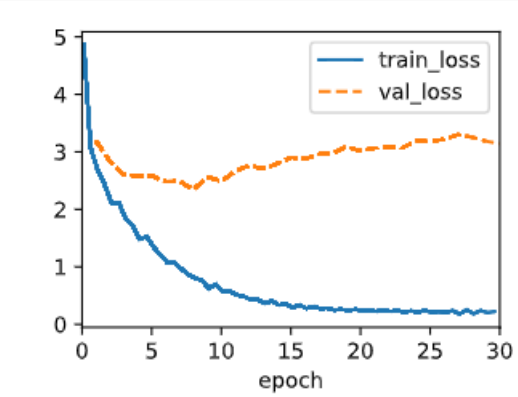


Figure 2: Results of Seq2Seq Model for Machine Translation with Changed Hyperparameters (Experiment 1)

Once the baseline model was plotted and training results were collected, the hyperparameter embedding size was changed from 256 to 1024. The model was trained again.

It can be seen that the plot is slightly better than the baseline model. The training loss is seen to be showing the same trend, but the validation loss is more stable and slightly lower than the baseline model. Figure 2 above shows the same results.

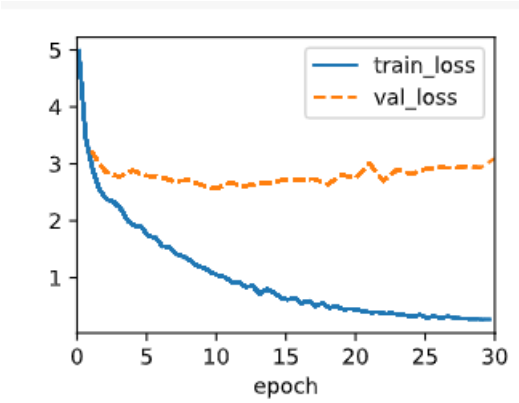


Figure 3: Results of Seq2Seq Model for Machine Translation with Changed Hyperparameters (Experiment 2)

After adjusting the embedding size, the hidden layer size was changed from 256 to 128, and the dropout was also decreased to 0.1 from 0.2. Changing the hyperparameters as mentioned, the plot is better than the earlier model and the baseline model. The training loss is the same all throughout, but the validation loss is much more stable and slightly lower.

Hence, by increasing the embedding size and decreasing the hidden layer size and dropout, the model showed better results. The hidden layer number was changed from 2 to 1 and then to 3, but the plot showed almost the same trend. Not a lot of difference was observed. Figure 3 shows the same results.

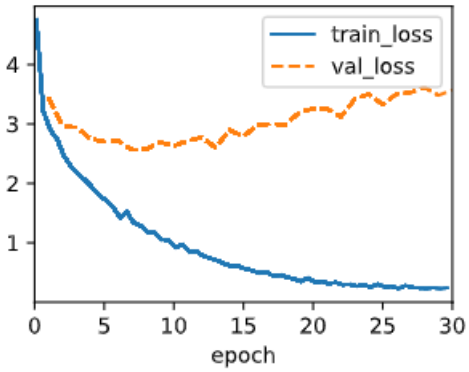


Figure 4: Results of Seq2Seq Model for Machine Translation with Different Encoder Decoder Layers

For Problem 1, Part 2, the requirement was to change the Encoder Decoder code to accommodate a different number of layers. To do this, a linear transformation layer was added to transform the hidden state from the encoder to match the dimensions expected by the decoder. For the encoder, the number of hidden layers used were 3 and 2 for the decoder.

It can be seen that the plot is very similar to that of the baseline mode with 2 layers for both encoder and decoder. Although very similar, the plot for different number of layers is slightly better in the sense that the validation loss is more stable and slightly lower, but the difference is very subtle. Figure 4 shows the same results.

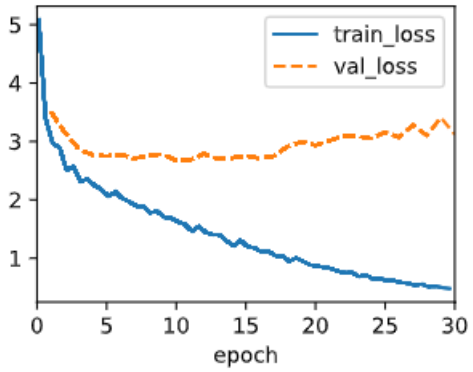


Figure 5: Results Seq2Seq Model for Machine Translation with LSTM

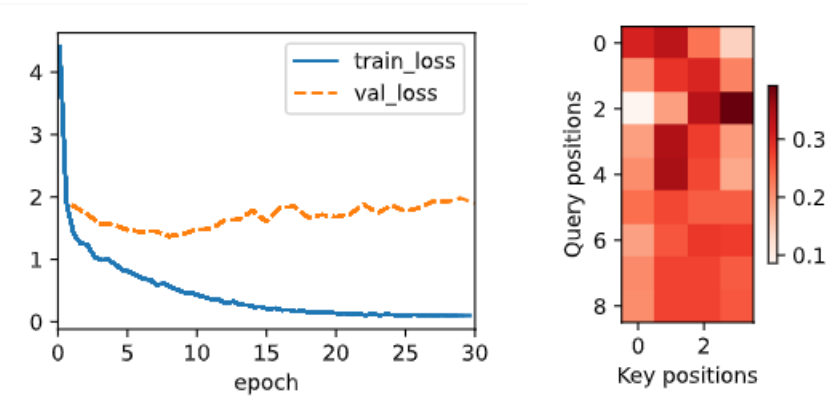
For Problem 1, Part 3, the same code used for the baseline model was used. Only this time, the GRU was changed with LSTM.

To replace GRU with LSTM, a few changes were made to the Encoder and Decoder including changing d2l.GRU to nn.LSTM and adjusting the “forward” methods to handle the LSTM’s cell state.

It can be seen that the training loss starts at around 5 and goes down to almost 0. The validation loss is slightly stable and lower than the baseline model with GRU. Figure 5 shows the same results. To conclude, the LSTM shows slightly better results than GRU.

Problem 2:

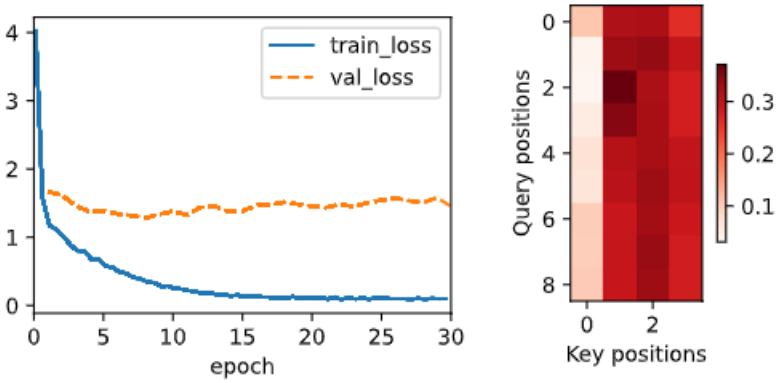
For Problem # 2, Part 1, the code for baseline Bahdanau attention model for Machine Translation based on Seq2Se2 was used from the d2l book and lecture notes to plot the model. Next, as required, the number of hidden layers were adjusted from 1 up to 4, and the model was retrained 4 different times. Figure 6 through 9 show the plots, the BLEU Scores of few Sequences that were translated from English to French using the trained model, and the Attention Weight Matrices for the models with different hidden layers , 1, 2, 3, and 4.




```
go . => ['va', '!'], bleu,1.000
i lost . => ["j'ai", 'perdu', '.'], bleu,1.000
he's calm . => ['<unk>', '.'], bleu,0.000
i'm home . => ['je', 'suis', 'chez', 'moi', '.'], bleu,1.000
```

Figure 6: Results for Bahdanau Attention Model with 2 Hidden Layers

For the 1st experiment, 2 hidden layers were used, and the model was trained. It can be seen that the training loss starts at around 4.5 and goes down to almost 0. On the other hand, the validation loss start at about 2, goes down to about 1.5 and come back up to 2. There is some instability in the plot of the validation loss. The BLEU score for ¾ sequences is 1.00. Figure 6 above shows the same.

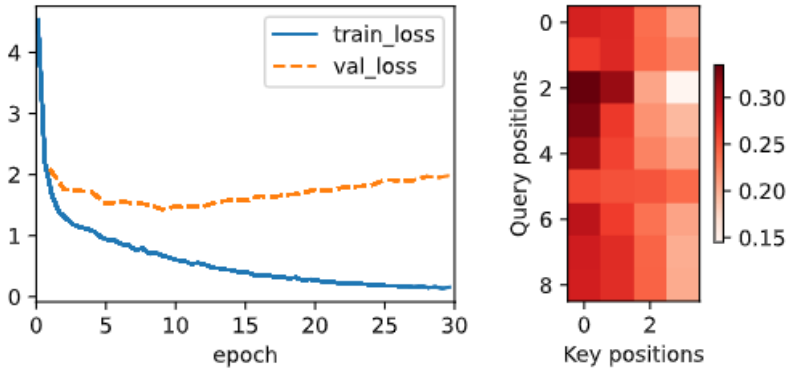



```
go . => ['va', '!'], bleu,1.000
i lost . => ["j'ai", 'perdu', '.'], bleu,1.000
he's calm . => ['<unk>', '.'], bleu,0.000
i'm home . => ['je', 'suis', 'chez', 'moi', '.'], bleu,1.000
```

Figure 7: Results for Bahdanau Attention Model with 1 Hidden Layer

For the 2nd experiment, only 1 hidden layer was used, and the model was trained. It can be seen that the training loss starts at around 4 and goes down to almost 0. On the other hand, the validation loss start at about 2, goes down to about 1.75 and come back up to 2. As compared to

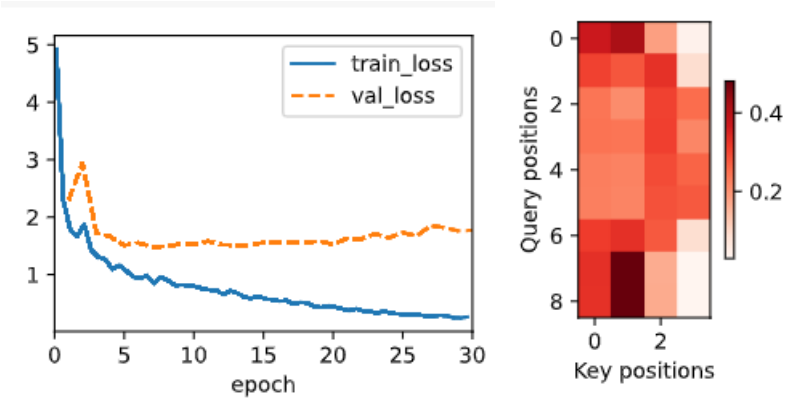
the model with 2 hidden layers, this model is much more stable, and the validation loss is also slightly lower. The BLEU score for ¾ sequences is 1.00. Figure 7 above shows the same.




```
go . => ['va', '!'], bleu,1.000
i lost . => ["j'ai", 'perdu', '.'], bleu,1.000
he's calm . => ['je', "l'ai", 'emporté', '.'], bleu,0.000
i'm home . => ['je', 'suis', 'chez', 'moi', '.'], bleu,1.000
```

Figure 8: Results for Bahdanau Attention Model with 3 Hidden Layers

For the 3rd experiment, 3 hidden layers were used, and the model was trained. It can be seen that the training loss starts at around 4.5 and goes down to almost 0. On the other hand, the validation loss start at about 2, goes down to about 1.5 and come back up to 2. As compared to the model with 2 hidden layers, this model is much more stable, but the validation loss is not as good as the model with just 1 layer. The BLEU score for ¾ sequences is 1.00. Figure 8 above shows the same.




```
go . => ['<unk>', '.'], bleu,0.000
i lost . => ["j'ai", 'perdu', '.'], bleu,1.000
he's calm . => ['trouve', 'confiance', '.'], bleu,0.000
i'm home . => ['je', 'suis', 'chez', 'moi', '.'], bleu,1.000
```

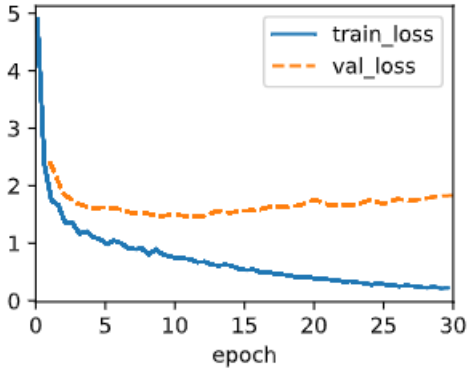
Figure 9: Results for Bahdanau Attention Model with 4 Hidden Layers

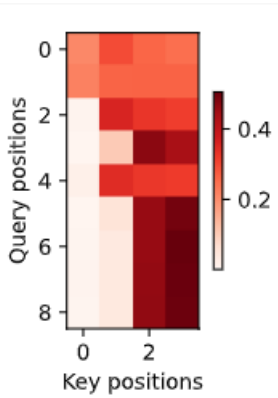
For the 4th experiment, 4 hidden layers were used, and the model was trained. It can be seen that the training loss starts at around 5 and goes down to almost 0. On the other hand, the validation

loss start at about 2, goes up to about 3.0 and come down to 2. As compared to the earlier models this model is in fact stable, but overall, the validation loss is not great. It goes up and then come down. The BLEU score for only ½ sequences is 1.00. Figure 9 above shows the same.

In conclusion, the model with only 1 hidden layer was the best in terms of stability and validation loss. The other models are mediocre in terms of the results, but overall, the validation loss values were higher, and some models are not stable at all.

For Problem 2, Part 2, the baseline Bahdanau Attention model's code from the earlier part of the problem was used as a starting point. Then, to accommodate the changes for replacing the GRU with LSTM, the GRU layer was replaced with an LSTM layer and the "forward" method was modified to handle the LSTM's hidden and cell states.






```
go . => ['<unk>', '.'], bleu,0.000
i lost . => ["j'ai", 'perdu', '.'], bleu,1.000
he's calm . => ['elle', 'est', '<unk>', '.'], bleu,0.000
i'm home . => ['je', 'suis', 'malade', '.'], bleu,0.512
```

Figure 10: Results for LSTM Bahdanau Attention Model

After making the necessary changes to replace the GRU with LSTM, the model was trained again, and the results were plotted. It can be seen that the training loss starts at around 5 and goes down to almost 0. The validation loss, on the other hand, starts at around 2.5, goes down to about 1.75 and comes back up to 2.0. The BLEU score for ½ sequences is 0.0, ¼ sequence is 1.0 and the other is 0.512.

In conclusion, this model is somewhat stable than the original, baseline model with GRU implementation, but the validation loss is higher. Also, the BLEU scores are lower for this model. The baseline model's BLEU scores were 1.0 for ¾ of the sequences. Thus, the GRU implementation for Bahdanau model showed better results in terms of validation loss and BLEU scores.

Bonus Problem:

The goal of this problem was to convert the Bahdanau Attention model to Luong Attention model. For this, I modified the Decoder class to accommodate the new model. The Encoder class was left unchanged. Few of the changes that I made to the Decoder class included a change in the concatenation method, which for Luong was Dot Product instead of Additive Attention. Also, to allow for Luong model to work, the attention score technique used was bilinear function instead of Multi-Layer Perceptron (MLP). Accordingly, the training code was also changed.

When the model was ran, the training code resulted in a few errors, which due to time limitation were not fixed. The code for the Luong model is in the Jupyter Notebook, which along with this report has been pushed to the GitHub Repo.
