ECGR 5106 – Real Time Machine Learning (Spring 2023)

Nahush D. Tambe – 801060297

https://github.com/Ntambe25

Homework # 4

```
'Scientific people,' proceeded the Time Traveller, after the pause

'But,' said the Medical Man, staring hard at a coal in the fire, 'if
```

Figure 1: Actual Text from 'The Time Machine' Dataset

To test the output sequence accuracy for all the different models, I used two actual output sequences from the Time Machine dataset. In addition to "it has", I used "proceeded the" and "staring hard". Figure 1 shows the two lines from the dataset. The dataset was referenced from Kaggle.

Problem 1:

GRU

For part 1 of Problem 1, the basic code for GRU model was used from lecture notes and d2l book. The hyperparameters, in this case, the hidden states, were experimented with. Figures 2 through 5 show the results for the GRU model with 32, 8, 128, and 1024 hidden states respectively.

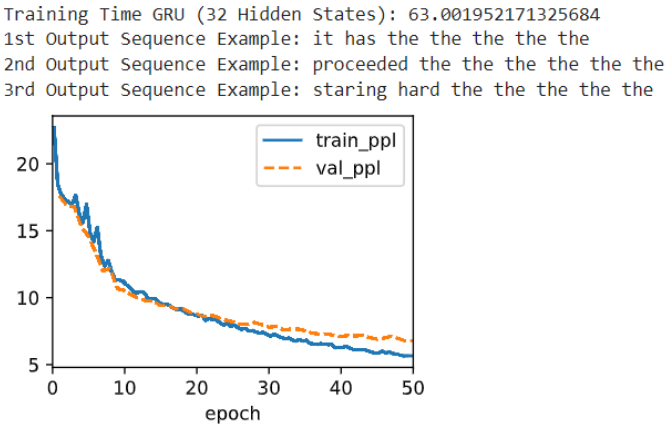

Figure 2: Results for GRU Model with 32 Hidden States

For GRU model with 32 hidden states, the training and validation perplexity starts at around 23 and goes down to about 5. At first, the generalization gap is not significantly visible, but as it goes further in epochs, the gap is somewhat visible. The training time for this model was 63

seconds. Figure 2 above shows the same results. The output sequences do not make much sense at all.

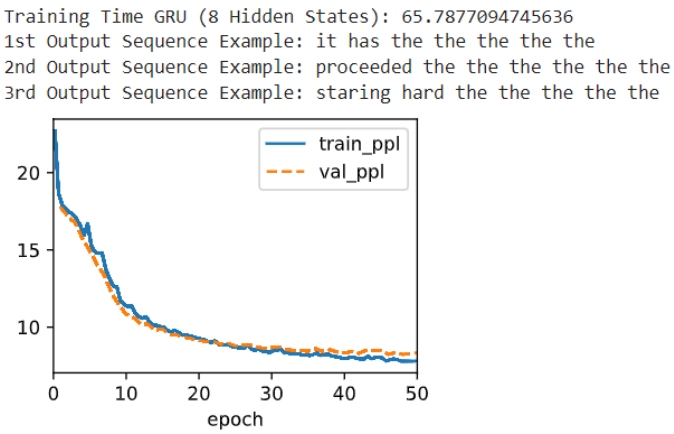

Figure 3: Results for GRU Model with 8 Hidden States

For GRU model with 8 hidden states, the training and validation perplexity also starts at around 23 and goes down to about 7. The generalization gap is not significantly visible throughout, and thus in terms of generalization, this model is better than the earlier model with 32 hidden states. But overall, the perplexity is higher. The training time for this model was 66 seconds which is slightly higher than the previous model. Figure 3 above shows the same results. The output sequences do not make much sense at all too.

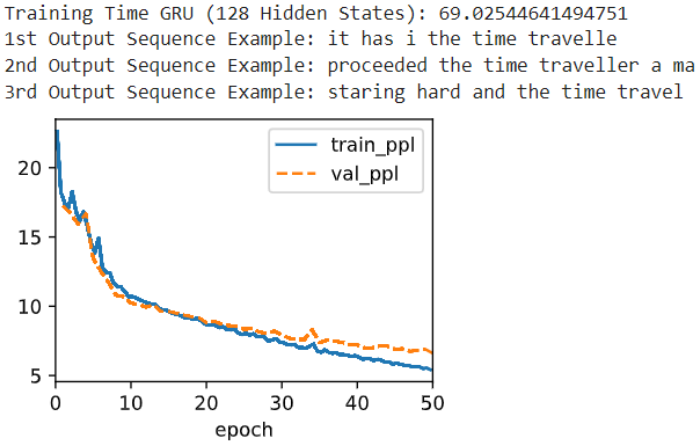

Figure 4: Results for GRU Model with 128 Hidden States

For GRU model with 128 hidden states, the training and validation perplexity also starts at around 23 and goes down to about 5. At first, the generalization gap is not significantly visible, but as it goes further in epochs, the gap is somewhat visible, but at the same time it is better than the gap in the model with 32 hidden states. The training time for this model was 69 seconds, which is just slightly higher than the previous models. Figure 4 above shows the same results.

The output sequences for this model are very close to the actual output sequences from the dataset, and hence this model is so far, the best.

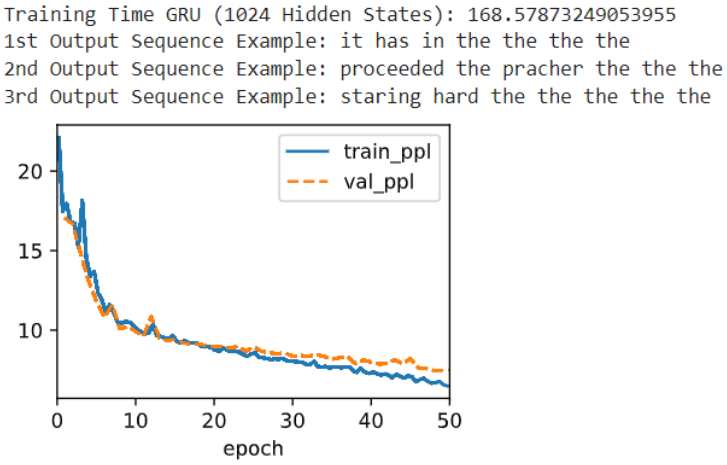

Figure 5: Results for GRU Model with 1024 Hidden States

For GRU model with 1024 hidden states, the training and validation perplexity also starts at around 23 and goes down to about 5. The generalization gap is not significantly visible throughout and thus in terms of generalization, this model is better than the model with 32 hidden states and about the same as the model with 128 hidden states. But the overall perplexity is higher, and the output sequences do not make much sense at all when compared to the previous models. The training time for this model was 169 seconds which is significantly higher than the previous models. Figure 5 above shows the same results.

In conclusion, the model with 128 hidden states has shown the best results. Of course, some aspects like the overall perplexity is not as great as other models, but in terms of generalization and predicting the output sequences, the model is the best.

LSTM

For part 2 of Problem 1, the basic code for LSTM model was used from lecture notes and d2l book. The hyperparameters, in this case, the hidden states were experimented with, and the results were compared with each other. Figures 6 through 9 show the results for the LSTM model with 32, 8, 128, and 1024 hidden states respectively.

```
Training Time LSTM (32 Hidden States): 69.16500520706177
1st Output Sequence Example: it has in the the the the
2nd Output Sequence Example: proceeded the the the the the the
3rd Output Sequence Example: staring hard the the the the the
```

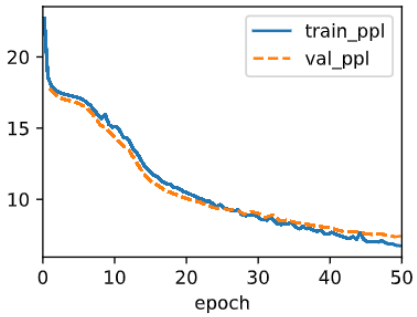

Figure 6: Results for LSTM Model with 32 Hidden States

For LSTM model with 32 hidden states, the training and validation perplexity starts at around 25 and goes down to about 5. The generalization gap is not significantly visible throughout the graph. The training time for this model was 69 seconds. Figure 6 above shows the same results. The output sequences do not make much sense at all.

```
Training Time LSTM (8 Hidden States): 66.05150318145752
1st Output Sequence Example: it has the the the the the
2nd Output Sequence Example: proceeded the the the the the the
3rd Output Sequence Example: staring hard the the the the the
```

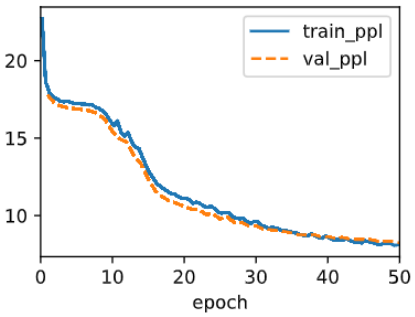

Figure 7: Results for LSTM Model with 8 Hidden States

For LSTM model with 8 hidden states, the training and validation perplexity starts at around 25 and goes down to about 7. The generalization gap for this model also is not significantly visible throughout and thus in terms of generalization, this model is about the same as the earlier model with 32 hidden states. But overall, the perplexity is higher. The training time for this model was 66 seconds which is slightly lower than the previous model. Figure 7 above shows the same results. The output sequences for this model also do not make much sense at all.

```
Training Time LSTM (128 Hidden States): 77.44745874404907
1st Output Sequence Example: it has the the the the the
2nd Output Sequence Example: proceeded the the the the the the
3rd Output Sequence Example: staring hard the the the the the
```

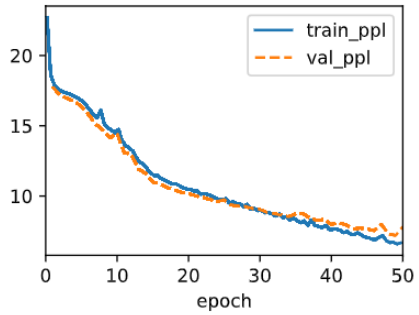

Figure 8: Results for LSTM Model with 128 Hidden States

For LSTM model with 128 hidden states, the training and validation perplexity also starts at around 25 and goes down to about 5. The generalization gap is not significantly visible throughout and thus in terms of generalization, this model is better than the earlier models. The training time for this model was 77 seconds which is not significantly higher than the previous models. Figure 8 above shows the same results. The output sequences for this model were also the same as previous models.

```
Training Time LSTM (1024 Hidden States): 212.132826089859
1st Output Sequence Example: it has and the the the the
2nd Output Sequence Example: proceeded the the the the the the
3rd Output Sequence Example: staring hard the the the the the
```

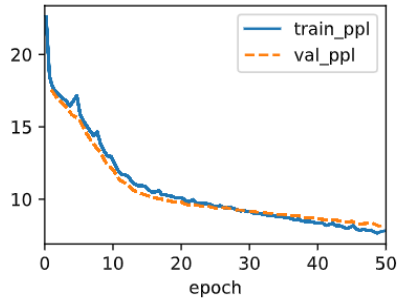

Figure 9: Results for LSTM Model with 1024 Hidden States

For LSTM model with 1024 hidden states, the training and validation perplexity starts at around 25 and goes down to about 5. The generalization gap is not significantly visible throughout and thus in terms of generalization, this model is quite better than the one with 32 and 8 hidden states. But the overall perplexity is higher than the previous model with 128 hidden states, and the output sequences do not make much sense at all. The training time for this model was 169 seconds which is significantly higher than the previous models. Figure 5 above shows the same results.

In conclusion, the model with 128 hidden states has shown the best results and the parameters used to come up with the conclusion were the overall perplexity, which was one of the lowest, and the training time, which was also quite lower than some of the other models.

Comparison between rnn.RNN, rnn.LSTM, and rnn.GRU

For part 3 of Problem 1, the basic code for rnn.RNN, rnn.LSTM, and rnn.GRU was used from the lecture slides and the d2l book. The concise implementations of each of the three types was used to compare training runtime, computational and model size complexities, training and validation perplexity, the output sequences. To accurately compare the three models, the same number of hyperparameters, in this case, the number of hidden states was used. For the previous model, 128 hidden states gave the most accurate results and hence, for this part of the problem, 128 hidden states were specified for all the models. Figures 10 through 15 show the results which includes the plots as well as the computational complexities and total number of parameter count.

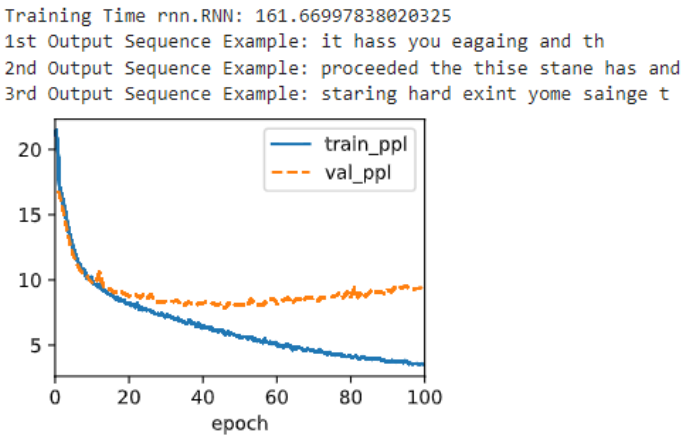

Figure 10: Results for rnn.RNN Model with 128 Hidden States

For rnn.RNN model, the concise implementation of the RNN, with 128 hidden states, the training and validation perplexity starts at around 22 and goes down to about 2. The generalization gap is not significantly visible at first but starting from around 10 epochs, the gap has increased. This shows that there is some overfitting for this model. The training time for this model was 160 seconds. The output sequences make somewhat sense and look like they are actual sentences. Figure 10 above shows the same results.

```
RNN(
  20.22 k, 100.000% Params, 20.84 MMac, 100.000% MACs,
  (rnn): RNN(20.22 k, 100.000% Params, 20.84 MMac, 100.000% MACs, 28, 128)
)
Computational complexity:       20.84 MMac
Number of parameters:           20.22 k
```

Figure 11: Computational Complexity and Parameter Count for rnn.RNN

Using ptflops, the computational complexity and the total number of parameter count was printed. Figure 11 shows the same. The computational complexity is around 21 MMac, and the total number of parameter count is around 22 k.

```
Training Time rnn.LSTM: 160.63158440589905
1st Output Sequence Example: it has on a lante the time
2nd Output Sequence Example: proceeded the time traveller was
3rd Output Sequence Example: staring hard at a coal in the fo
```

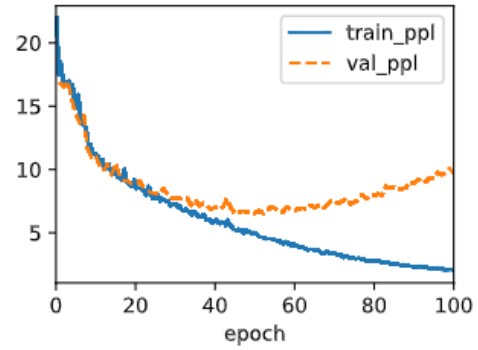

Figure 12: Results for rnn.LSTM Model with 128 Hidden States

For rnn.LSTM model, the concise implementation of the LSTM, with 128 hidden states, the training and validation perplexity starts at around 22 and goes down to about 1. The generalization gap is not significantly visible at first but starting from around 20 epochs, the gap has increased. This shows that there is some overfitting for this model. The training time for this model was 160 seconds. The output sequences for this model are very close to the actual output sequences from the dataset, and hence this model is better than the rnn.RNN model. Figure 12 above shows the same results.

```
LSTM(
  80.9 k, 100.000% Params, 84.15 MMac, 100.000% MACs,
  (rnn): LSTM(80.9 k, 100.000% Params, 84.15 MMac, 100.000% MACs, 28, 128)
)
Computational complexity:       84.15 MMac
Number of parameters:           80.9 k
```

Figure 13: Computational Complexity and Parameter Count for rnn.LSTM

Using ptflops, the computational complexity and the total number of parameter count was printed. Figure 13 shows the same. The computational complexity is around 84 MMac, and the total number of parameter count is around 81 k. Both the computational complexity and the parameter count has increased significantly compared to the rnn.RNN model, which was expected.

```
Training Time rnn.GRU: 189.15448093414307
1st Output Sequence Example: it hass to the the time tr
2nd Output Sequence Example: proceeded the the the the the the
3rd Output Sequence Example: staring hard to the the time tra
```

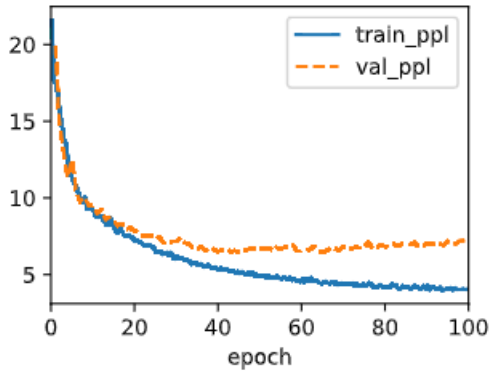

Figure 14: Results for rnn.GRU Model with 128 Hidden States

For rnn.GRU model, the concise implementation of the GRU, with 128 hidden states, the training and validation perplexity starts at around 22 and goes down to about 2. The generalization gap is not significantly visible at first but starting from around 10 epochs, the gap has increased. This shows that there is some overfitting for this model. The training time for this model was 190 seconds. The output sequences for this model are do not make very much sense and thus are similar to the ones for rnn.RNN model. Figure 14 above shows the same results.

```
GRU(
  5.95 k, 100.000% Params, 6.32 MMac, 100.000% MACs,
  (rnn): GRU(5.95 k, 100.000% Params, 6.32 MMac, 100.000% MACs, 28, 32)
)
Computational complexity:       6.32 MMac
Number of parameters:           5.95 k
```

Figure 15: Computational Complexity and Parameter Count for rnn.LSTM

Using ptflops, the computational complexity and the total number of parameter count was printed. Figure 15 shows the same. The computational complexity is around 6.32 MMac, and the total number of parameter count is around 6 k. Both the computational complexity and the parameter count has decreased significantly compared to the rnn.RNN and rnn.LSTM models.

Thus, in comparison for the 3 models, rnn.RNN, rnn.LSTM, and rnn.GRU, the LSTM model is the best model as the generalization gap is lower than the other two models and overall perplexity is also smaller. But if compared to in terms of computational complexity and total number of parameter count, the rnn.GRU model is the best having significantly lower values.

Problem 2:

For part 1 of Problem 2, the basic code for Deep LSTM and Deep GRU models was used from lecture slides and d2l book. The concise implementation for GRU was used and it was modified for the LSTM model. Figures 16 and 17 below show the results for the two models with 2 hidden layers and 32 hidden states.

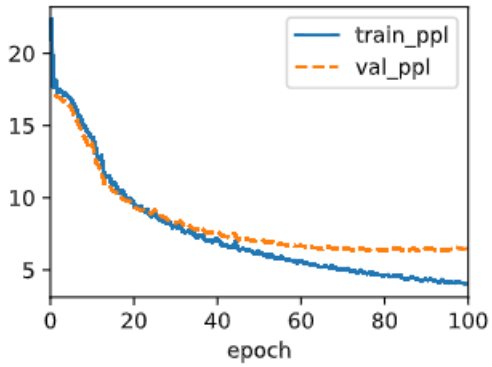

Figure 16: Results for Deep GRU Model

For the Deep GRU model with 2 hidden layers and 32 hidden states, the training and validation perplexity started at around 20 and goes down to about 5. The generalization gap is not visible at first, but starting at around 30 epochs, it has increased slightly. Figure 16 above shows the same results.

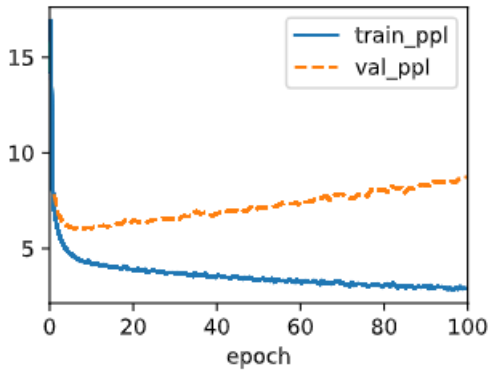

Figure 17: Results for Deep LSTM Model

For the Deep LSTM model with 2 hidden layers and 32 hidden states, the training and validation perplexity started at around 17 and goes down to about 2. The generalization gap has increased starting with 1st epoch and it increased all through the 100 epochs. Figure 17 above shows the same results.

Thus, in conclusion, even though the overall perplexity for the LSTM model was lover, the model is not suited for the generalized data, as the generalization gap is significantly larger compared to the deep GRU model above.

For part 2 of Problem 2, the two models plotted above were used to figure out the training time and predict the output strings. Along with these, the model complexities as well as the total parameter count were also calculated. Figure 18 through 21 below show the same results for Deep GRU and Deep LSTM models respectively.

```
Training Time Deep GRU: 191.88320302963257
1st Output Sequence Example: it has greel the medical m
2nd Output Sequence Example: proceeded the time traveller dime
3rd Output Sequence Example: staring hard have the time trave
```

Figure 18: Training Time and Output Strings for Deep GRU Model

After plotting the training and validation perplexity for the Deep GRU model, the training runtime was calculated, and the output strings were predicted. Figure 18 above shows the same. The output strings for this model was very close to the actual strings from the dataset and the training runtime was around 191 seconds.

```
GRU(
  12.29 k, 100.000% Params, 13.04 MMac, 100.000% MACs,
  (rnn): GRU(12.29 k, 100.000% Params, 13.04 MMac, 100.000% MACs, 28, 32, num_layers=2)
)
Computational complexity:       13.04 MMac
Number of parameters:           12.29 k
```

Figure 19: Computational Complexity and Parameter Count for Deep GRU

Using ptflops, the computational complexity and the total number of parameter count was printed. Figure 19 shows the same. The computational complexity is around 13 MMac, and the total number of parameter count is around 13 k.

```
Training Time Deep LSTM: 192.35123205184937
1st Output Sequence Example: it has his denith bree spe
2nd Output Sequence Example: proceeded the three direction all
3rd Output Sequence Example: staring hard and the time travel
```

Figure 20: Training Time and Output Strings for Deep LSTM Model

After plotting the training and validation perplexity for the Deep LSTM model, the training runtime was calculated, and the output strings were predicted. Figure 20 above shows the same. The output strings for this model make somewhat sense and look like actual sentences, they are not close to the actual sentences referred to from the dataset. The training runtime was around 192 seconds.

```
LSTM(
  16.38 k, 100.000% Params, 17.43 MMac, 100.000% MACs,
  (rnn): LSTM(16.38 k, 100.000% Params, 17.43 MMac, 100.000% MACs, 28, 32, num_layers=2)
)
Computational complexity:       17.43 MMac
Number of parameters:           16.38 k
```

Figure 21: Computational Complexity and Parameter Count for Deep LSTM

Using ptflops, the computational complexity and the total number of parameter count was printed. Figure 21 shows the same. The computational complexity is around 18 MMac, and the total number of parameter count is around 17 k.

Thus, in conclusion, the Deep GRU model performs better on the generalized data and the Deep LSTM model, even though has a overall lower perplexity, performs poorly on unseen data. In terms of model complexity and total parameter count, the GRU model has significantly lower numbers.

For part 3 of Problem 2, the hyperparameters, in this case the number of hidden layers and hidden states was experimented with for the Deep GRU and LSTM models. Figures 22 through 29 show the same results with 128 hidden states and 8 hidden layers for GRU and LSTM respectively. Also included are the figures for the model complexities and the total parameter counts for each of the different models.

```
Training Time Deep GRU (128 Hidden States): 204.67187929153442
1st Output Sequence Example: it has hand a small has a
2nd Output Sequence Example: proceeded the three and so i neve
3rd Output Sequence Example: staring hard and any of the ball
```

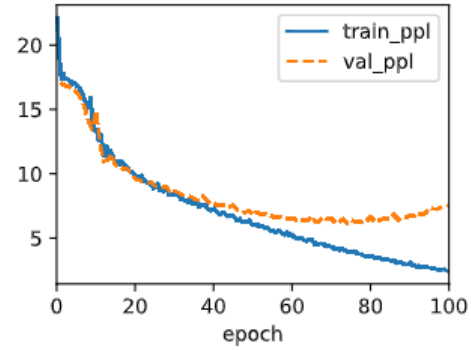

Figure 22: Results for Deep GRU Model with 128 Hidden States

For the Deep GRU model with 128 hidden states, the training and validation perplexity start at around 20 and goes down to about 2. The generalization gap is not significantly visible at first, but starting at around 40 epochs, the gap is slightly visible and it also increases slightly, but gradually as the plot progresses further. Overall, the perplexity is lower than the earlier GRU model with 32 hidden states. The training time is around 204 seconds, which is slightly better than the GRU model with 32 hidden states. Figure 22 above shows the same results.

The output sequences make somewhat sense but are not close to the actual sentences from the dataset.

In comparison to the Deep LSTM model, this model is definitely the better as the generalization gap is significant for the LSTM model.

```
GRU(
  159.74 k, 100.000% Params, 165.41 MMac, 100.000% MACs,
  (rnn): GRU(159.74 k, 100.000% Params, 165.41 MMac, 100.000% MACs, 28, 128, num_layers=2)
)
Computational complexity:       165.41 MMac
Number of parameters:           159.74 k
```

Figure 23: Computational Complexity and Parameter Count for Deep GRU with 128 Hidden States

Using ptflops, the computational complexity and the total number of parameter count was printed. Figure 23 shows the same. The computational complexity is around 165 MMac, and the total number of parameter count is around 160 k. In comparison to the Deep GRU model with 32 hidden states as well as the deep LSTM model with 32 hidden states, both model complexity and total parameter count is significantly higher.

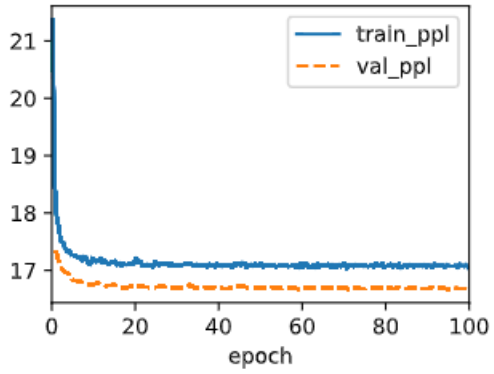

Figure 24: Results for Deep GRU Model with 8 Hidden Layers

For the Deep GRU model with 8 hidden layers, the training and validation perplexity start at around 20 and goes down to only about 16. The generalization gap is constant of about 1 throughout the plot. Overall, the perplexity is very much higher than the earlier GRU models with 32 and 128 hidden states. The training time is around 201 seconds, which is slightly better than the GRU models with 32 and 128 hidden states. Figure 24 above shows the same results.

In comparison to the Deep LSTM model, this model is definitely not better as even though the generalization gap is not significant, the overall perplexity is higher than the LSTM model.

The output sequences were looking like actual sentences, but were not close in comparison to the sentences from the dataset.

```
GRU(
  50.3 k, 100.000% Params, 53.35 MMac, 100.000% MACs,
  (rnn): GRU(50.3 k, 100.000% Params, 53.35 MMac, 100.000% MACs, 28, 32, num_layers=8)
)
Computational complexity:       53.35 MMac
Number of parameters:           50.3 k
```

Figure 25: Computational Complexity and Parameter Count for Deep GRU with 8 Hidden Layers

Using ptflops, the computational complexity and the total number of parameter count was printed. Figure 25 shows the same. The computational complexity is around 53 MMac, and the total number of parameter count is around 50 k. In comparison to the Deep GRU model with 32 hidden states , this model has a higher model complexity as well as total parameter count value, but lower when compared to the Deep GRU model with 128 hidden states. In comparison to the Deep LSTM model, this model has a higher value for both model complexity and parameter count.

```
Training Time Deep LSTM (128 Hidden States): 206.8498477935791
1st Output Sequence Example: it has in the time travell
2nd Output Sequence Example: proceeded the time traveller the
3rd Output Sequence Example: staring hard the time traveller
```

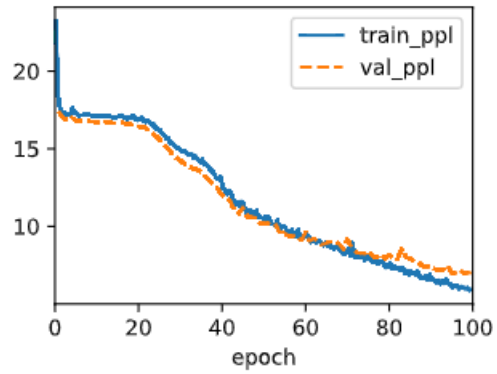

Figure 26: Results for Deep LSTM Model with 128 Hidden States

For the Deep LSTM model with 128 hidden states, the training and validation perplexity start at around 23 and goes down to about 5. The generalization gap not seen throughout the plot. Overall, the perplexity is very much higher than the earlier GRU models with 32 and 128 hidden states. The training time is around 201 seconds, which is slightly better than the GRU models with 32 and 128 hidden states. Figure 26 above shows the same results.

In comparison to the Deep GRU model, this model is definitely not better as even though the generalization gap is not significant, the overall perplexity is higher than the LSTM model.

```
LSTM(
  212.99 k, 100.000% Params, 220.73 MMac, 100.000% MACs,
  (rnn): LSTM(212.99 k, 100.000% Params, 220.73 MMac, 100.000% MACs, 28, 128, num_layers=2)
)
Computational complexity:       220.73 MMac
Number of parameters:           212.99 k
```

Figure 27: Computational Complexity and Parameter Count for Deep LSTM with 8 Hidden Layers

Using ptflops, the computational complexity and the total number of parameter count was printed. Figure 27 shows the same. The computational complexity is around 221 MMac, and the total number of parameter count is around 213 k. In comparison to the Deep LSTM model with 32 hidden states , this model has a higher model complexity as well as total parameter count value, but lower when compared to the Deep LSTM model with 128 hidden states.

The output sequences were looking like actual sentences, but were not close in comparison to the sentences from the dataset.

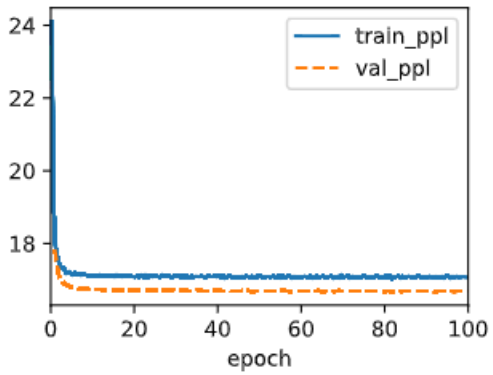

Figure 28: Results for Deep LSTM Model with 8 Hidden States

For the Deep LSTM model with 8 hidden layers, the training and validation perplexity start at around 20 and goes down to only about 16. The generalization gap is constant of about 1 throughout the plot. Overall, the perplexity is very much higher than the earlier LSTM models with 32 and 128 hidden states. The training time is around 201 seconds, which is slightly better than the GRU models with 32 and 128 hidden states. Figure 28 above shows the same results.

The output sequences were looking like actual sentences, but were not close in comparison to the sentences from the dataset.

```
LSTM(
  67.07 k, 100.000% Params, 71.3 MMac, 100.000% MACs,
  (rnn): LSTM(67.07 k, 100.000% Params, 71.3 MMac, 100.000% MACs, 28, 32, num_layers=8)
)
Computational complexity:       71.3 MMac
Number of parameters:           67.07 k
```

Figure 29: Computational Complexity and Parameter Count for Deep LSTM with 8 Hidden Layers

Using ptflops, the computational complexity and the total number of parameter count was printed. Figure 29 shows the same. The computational complexity is around 71 MMac, and the total number of parameter count is around 68 k. In comparison to the Deep LSTM model with 32 hidden states , this model has a higher model complexity as well as total parameter count value, but lower when compared to the Deep LSTM model with 128 hidden states.

The output sequences were looking like actual sentences, but were not close in comparison to the sentences from the dataset.
