https://github.com/Christian-Martens-UNCC/ECGR-4106/tree/main/Homework_4-GRUs_%26_LSTMs

Problem 1:

A) The hyper parameters I chose to focus on were training time/epochs and number of hidden states. I used epochs of sizes [5, 10, 25, 50] and number of hidden layers of sizes [8, 16, 32, 64]. In total that creates 16 different GRU models to compare and contrast. All 16 models have been plotted in the code documentation, so I'm just going to go over a few of the more interesting ones here and give some observations on them. I used a few different prompts to evaluate the models which are listed in the code as the "phrases". A few common threads between all of the plots are:
   - Increasing the number of hidden layers increased the training time, MACs, and saved parameters.
   - Overfitting seemed to occur around the 30th epoch.
   - "and" and "the" were quite common as well as "time" and "traveler" in the more advanced models. This is to be expected since these are amongst the most common words in the text file used to generate the dataset for this homework.
   - Good models generally tended to be under a training and validation error of 10.
   - The lowest validation loss bottomed out at around 8.

GRU – 16 Hiddens, 10 Epochs

```
train_nlp(c[f'hiddens{hiddens[1]}_epochs{epochs[1]}'],
          'gru',
          data,
          chars,
          phrases,
          epochs[1],
          bs)
```

```
Input Phrase : 'the boy and the girl'
        Model prediction out to 50 characters :
        the boy and the girle and and and and and and and and and and and
```

This was the first model that didn't just repeat "and" or "the" over and over. Granted, its only for 1 letter (which is 'e', so not exactly revolutionary), but still it's noteworthy.

GRU – 32 Hiddens, 25 Epochs

```
train_nlp(c[f'hiddens{hiddens[2]}_epochs{epochs[2]}'],
          'gru',
          data,
          chars,
          phrases,
          epochs[2],
          bs)
```

```
Input Phrase : 'it has'
        Model prediction out to 50 characters :
        it has is a moure a moure a moure a moure a moure a mour

Input Phrase : 'it has a'
        Model prediction out to 50 characters :
        it has a moure a moure a moure a moure a moure a moure a m

Input Phrase : 'it has a strong'
        Model prediction out to 50 characters :
        it has a strong the time the time the time the time the time the

Input Phrase : 'it has a strong sense'
        Model prediction out to 50 characters :
        it has a strong sensed and the time the time the time the time the time

Input Phrase : 'the boy'
        Model prediction out to 50 characters :
        the boy the time the time the time the time the time the

Input Phrase : 'the boy and the girl'
        Model prediction out to 50 characters :
        the boy and the girly of the time the time the time the time the time
```

This was the first model to consistently give different results from “and” or “the”. The sentences are not comprehensible, but they are much more interesting compare to the alternative.

GRU – 32 Hiddens, 50 Epochs

```
train_nlp(c[f'hiddens{hiddens[2]}_epochs{epochs[3]}'],
          'gru',
          data,
          chars,
          phrases,
          epochs[3],
          bs)
```

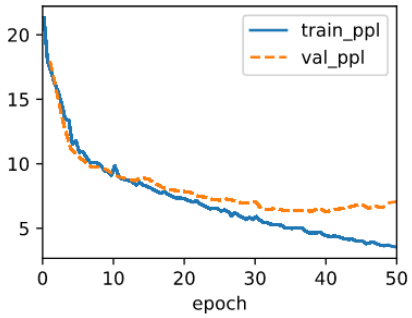
```
Input Phrase : 'it has'
       Model prediction out to 50 characters :
       it has all the prover dimensions of space and the prover
```

This was the first model that had what I consider to be a “readable” sentence. It almost sounds like it actually is writing something that might be similar to what could be in the actual book.

GRU – 64 Hiddens, 50 Epochs

```
train_nlp(c[f'hiddens{hiddens[3]}_epochs{epochs[3]}'],
          'gru',
          data,
          chars,
          phrases,
          epochs[3],
          bs)
```

```
Input Phrase : 'study to show that you are'
       Model prediction out to 50 characters :
       study to show that you are prou the mere why one dimension of the time trave
GRU(
  18.05 k, 100.000% Params, 18.94 MMac, 100.000% MACs,
  (rnn): GRU(18.05 k, 100.000% Params, 18.94 MMac, 100.000% MACs, 28, 64)
)
```



The largest GRU model I trained. It took over 8 minutes to train on my laptop.

B) Using the same hyperparameters and number of models as part A, here are a few observations and a few of the more interesting models for the LSTMs:
- Increasing the number of hidden layers increased the training time, MACs, and saved parameters.
- Overfitting seemed to occur around the 35th epoch.
- 'a', 't', and 'e' were all very common. They showed up multiple times either by themselves or paired with one or two of the other common letters. While that is no surprise seeing how these are the most common letters, it is surprising to see just how reluctant the models were to form longer, more complex words.
- Good models generally tended to be under a training and validation error of 10.
- The lowest validation loss bottomed out at around 8.

LSTM – 32 Hiddens, 5 epochs

```
train_nlp(d[f'hiddens{hiddens[2]}_epochs{epochs[0]}'],
          'lstm',
          data,
          chars,
          phrases,
          epochs[0],
          bs)
```

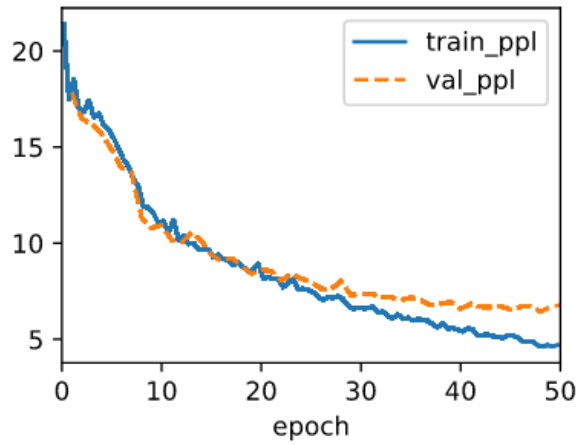
```
Input Phrase : 'it has'
       Model prediction out to 50 characters :
       it has e  e  e   e   e   e   e   e   e   e   e   e   e

Input Phrase : 'it has a'
       Model prediction out to 50 characters :
       it has a e  e   e   e   e   e   e   e   e   e   e   e   e

Input Phrase : 'it has a strong'
       Model prediction out to 50 characters :
       it has a strong e  e  e   e   e   e   e   e   e   e   e   e   e

Input Phrase : 'it has a strong sense'
       Model prediction out to 50 characters :
       it has a strong sense  t e  e  e   e   e   e   e   e   e   e   e   e

Input Phrase : 'the boy'
       Model prediction out to 50 characters :
       the boy  e  e   e   e   e   e   e   e   e   e   e   e   e
```

This one was very unlike any of the other models. It deduced that the optimal strategy was to space e's 3 spaces away from each other… very strange. Obviously, this is likely due to the short training time of 5 epochs.

LSTM – 64 Hidden, 50 Epochs

```
train_nlp(d[f'hiddens{hiddens[3]}_epochs{epochs[3]}']
          'lstm',
          data,
          chars,
          phrases,
          epochs[3],
          bs)
```

```
Input Phrase : 'study to show that you are'
       Model prediction out to 50 characters :
       study to show that you are the time traveller this time traveller this time
LSTM(
  24.06 k, 100.000% Params, 25.3 MMac, 100.000% MACs,
  (rnn): LSTM(24.06 k, 100.000% Params, 25.3 MMac, 100.000% MACs, 28, 64)
)
```



This was the most complex LSTM that was constructed.

C) Generally speaking, the GRUs outperformed the LSTMs in every category. They trained faster, had few MACs and parameters, and produced more interesting sentences with either less training time or fewer hidden layers. The two areas in which LSTMs outperformed GRUs was that the more advanced LSTMs rarely misspelled a word while GRUs constantly had spelling errors and that LSTMs had a lower training-validation loss gap.

Problem 2:

A) In addition to the hyper parameters used in problem 1, and additional hyper parameter of hidden layers was added. The hidden layers could be a size of [2, 3]. To increase the completion time of the models (after all, my computer is not very fast and I needed to have good

information to turn in for the homework), I removed the 5 epochs and 8 hidden layer parameters since those were (obviously) the worst performing and I wanted to see how advanced these models could get compared to the more sophisticated models from problem 1. In total, there are 18 Deep-GRU models. I used the same prompts to evaluate the models as used in problem 1. A few common threads between all of the plots are:

- Number of MACs and parameters scaled with the number of layers roughly linearly.
- Training-validation loss gap seemed smaller than the single layer GRUs. Overfitting seemed to occur around the 35th epoch.
- Training was much more gradual and sporadic compared to the single layer GRUs.
- For the more complex models, error would seemingly randomly spike to absurdly high values before correcting the next epoch (I think this has to do with learning rate and model complexity)
- There is an error spike right at around 10 epochs for each of the models.

Deep GRU – 16 Hiddens, 50 Epochs, 3 Layers

```
train_nlp(e[f'hiddens{hiddens[1]}_epochs{epochs[3]}_layer{layers[1]}'],
          'deep_gru',
          data,
          chars,
          phrases,
          epochs[3],
          bs)
```

```
Input Phrase : 'it has'
        Model prediction out to 50 characters :
        it has the proch and the mere and the proch and the mere

Input Phrase : 'it has a'
        Model prediction out to 50 characters :
        it has and the mere and the proch and the mere and the pro

Input Phrase : 'it has a strong'
        Model prediction out to 50 characters :
        it has a strong the momension and the mere and the proch and the
```

This model was one of the only models to consistently use rare consonants such as ‘c’, ‘p’, and ‘m’ outside of the common words that used them. Additionally, we are seeing more ‘o’ letters in this model than any other model so far.

Deep GRU – 64 Hiddens, 50 Epochs, 2 Layers

```
train_nlp(e[f'hiddens{hiddens[3]}_epochs{epochs[3]}_layer{layers[0]}'],
          'deep_gru',
          data,
          chars,
          phrases,
          epochs[3],
          bs)
```

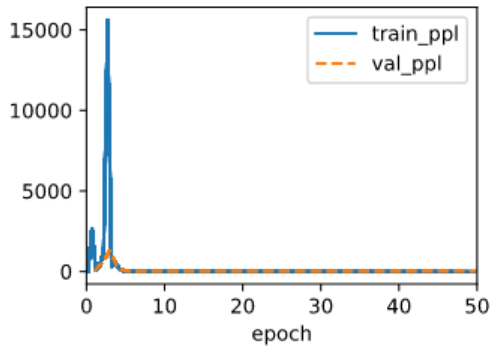
```
Input Phrase : 'the boy'
        Model prediction out to 50 characters :
        the boy said the psychologist can move about in the medic
```

This is the first time a model actually created a completely normal sentence (it would have continued "about in the medical bay…" had the number of prediction characters been increased).

Deep GRU – 64 Hiddens, 50 Epochs, 3 Layers

```
train_nlp(e[f'hiddens{hiddens[3]}_epochs{epochs[3]}_layer{layers[1]}']
          'deep_gru',
          data,
          chars,
          phrases,
          epochs[3],
          bs)
```

```
Input Phrase : 'study to show that you are'
        Model prediction out to 50 characters :
        study to show that you are which there is which there is whick the time trav
GRU(
  67.97 k, 100.000% Params, 70.98 MMac, 100.000% MACs,
  (rnn): GRU(67.97 k, 100.000% Params, 70.98 MMac, 100.000% MACs, 28, 64, num_layers=3)
)
```



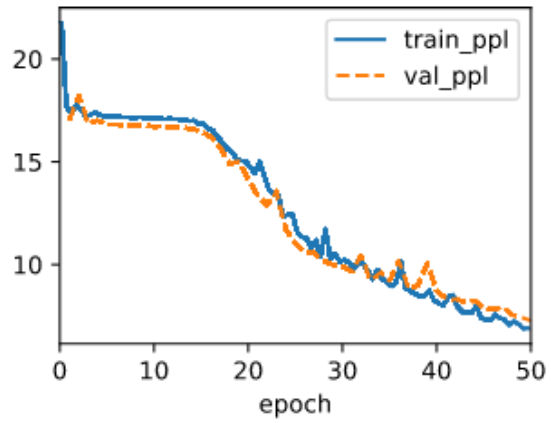
The most complex Deep GRU model that was ran. It didn't perform that well and took about 25 minutes alone just to train on my laptop.

B) Using the same hyperparameters and number of models as part A, here are a few observations and a few of the more interesting models for the Deep LSTMs:
- Number of MACs and parameters scaled with the number of layers roughly linearly.
- Overfitting did not occur even after 50 epochs
- Small words that repeated were more common for these models than the Deep GRUs
- Good models generally tended to be under a training and validation error of around 10.
- Training was much more gradual and sporadic compared to the single layer GRUs.
- The Deep LSTM models would plateau before beginning to learn more at about the 15 epoch mark.

Deep LSTM – 32 Hiddens, 50 Epochs, 2 Layers

```
train_nlp(f[f'hiddens{hiddens[2]}_epochs{epochs[3]}_layer{layers[0]}'],
          'deep_lstm',
          data,
          chars,
          phrases,
          epochs[3],
          bs)
```

```
Input Phrase : 'study to show that you are'
        Model prediction out to 50 characters :
        study to show that you are and the the the the the the the the the the t
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)
)
```



Though the quality of the model's predictions are less than stellar, this graph does a very good job of demonstrating that plateau and drop at around epoch 15 as well as the very low training-validation loss gap.

C) Like the single layer GRUs and LSTMs, Deep GRUs just seem to be better in every regard for this application except in regards to the validation gap. The Deep GRU models were smaller, faster, and generally produced better sentences. I don't think that, for this application anyway, that having more than 2 layers would be beneficial (at least, I saw no real benefit to using 3 layers

compared to 2 as many of my 2 layer models outperformed the 3 layer models and the 3 layer models take much longer to train) and that would be if I even used more than 1 layer. The one layer models worked just about as well as the 2 and 3 layer models did and they were much smaller and quicker to train.