Deep Learning: Assignment 4

Jury Andrea D’Onofrio

November 2022

1 Problem Setups and Preliminaries

1.1

algebra__linear_1d

1. train.x
   - Number of sentences → 1999998
   - Number of characters → 73368093
   - Average question length in terms of number of characters → 36,68408318
2. train.y
   - Number of sentences → 1999998
   - Number of characters → 5968004
   - Average answer length in terms of number of characters → 2,984004984
3. interpolate.x
   - Number of sentences → 10000
   - Number of characters → 419669
   - Average question length in terms of number of characters → 41,9669
4. interpolate.y
   - Number of sentences → 10000
   - Number of characters → 32978
   - Average answer length in terms of number of characters → 3,2978

comparison__sort

1. train.x
   - Number of sentences → 1999998
   - Number of characters → 81588795

- Average question length in terms of number of characters → 40,79443829

2. train.y
   - Number of sentences → 1999998
   - Number of characters → 37413762
   - Average answer length in terms of number of characters → 18,70689971
3. interpolate.x
   - Number of sentences → 10000
   - Number of characters → 434938
   - Average question length in terms of number of characters → 43,4938
4. interpolate.y
   - Number of sentences → 10000
   - Number of characters → 213802
   - Average answer length in terms of number of characters → 21,3802

numbers__place_value

1. train.x
   - Number of sentences → 1999998
   - Number of characters → 78429947
   - Average question length in terms of number of characters → 39,21501272
2. train.y
   - Number of sentences → 1999998
   - Number of characters → 3999996
   - Average answer length in terms of number of characters → 2
3. interpolate.x
   - Number of sentences → 10000
   - Number of characters → 413219
   - Average question length in terms of number of characters → 41,3219
4. interpolate.y
   - Number of sentences → 10000
   - Number of characters → 20000
   - Average answer length in terms of number of characters → 2

I calculate the average question and answer lengths in terms of number of characters as number of characters divided by number of sentences

2 Dataloader

2.1

No, the implementation doesn't use the same Vocabulary for the source and the target. For example they are different in terms of lengths.

2.2

The algorithm returns this token when the word is not in the Vocabulary and we don't want to add it (`extend_vocab = False`).

3

3.1

I implement the function `forward_separate`, please see the code.

4

I implement the main structure of the greedy search and the two stopping criteria, please the code. However, since I have an error in terms of shape inside it that I was not able to overcome - shape '[64, 512, 32]' is invalid for input of size 802816 - I was not able to test the entire function.

4.1

I implement the greedy search algorithm, please see the code.

4.2

The model uses a lot of computational resources because it has an encoder/decoder structure. I also need to accumulate the gradient during the training phase because the model is very large, this increases the computational time.

4.3

I implement the two stopping criteria:

- The first is when the model outputs the end-of-sentence token `eos`.
- The second is when only if all tokens match the true answer sequence.

Please see the code.

4.4

I carry out search in batch mode, please see the code.

5

5.1

I Implement a function which computes the accuracy both on training and validation set every 1000 training steps. The model prediction is counted as correct, only when the entire output sequence (all characters) matches that of the correct answer. Please see the code.

6

6.1

I decide to use the `CrossEntropyLoss` because this is a classification problem.

6.2

I implement the training pipeline. I also keep track of both training and validation losses, as well as accuracy on the validation and training data subsets, every 1000 training steps. Please see the code.

6.3

I accumulate gradients for 10 steps. Please see the code.

7

7.1

I implement the training pipeline using the given hyper-parameters, please see the code.

7.2

I reach over 90% after 5 min when training on a GPU and 100% after 10 min. Figure 1 shows the training and validation loss, while Figure 2 shows the training and validation accuracy. Both of the graphs show good results.

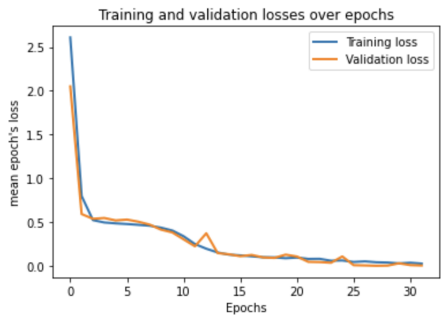

Figure 1: Training and Validation loss

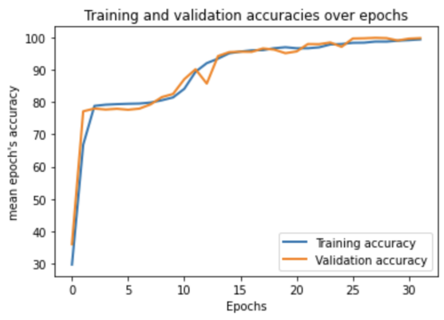

Figure 2: Training and Validation accuracy

7.3

My first configuration achieves over the 90% of accuracy in 10 min when training on GPU. I use the following hyper-parameters:

- batch size = 128
- accumulating gradients for 10 steps
- learning rate = 0.0001
- Gradient clipping rate = 0.1
- Model with 3 encoder layers, 2 decoder layers, hidden dimension of 256, feed-forward dimension of 1024, using 8 attention heads.

Figure 3 shows my results.

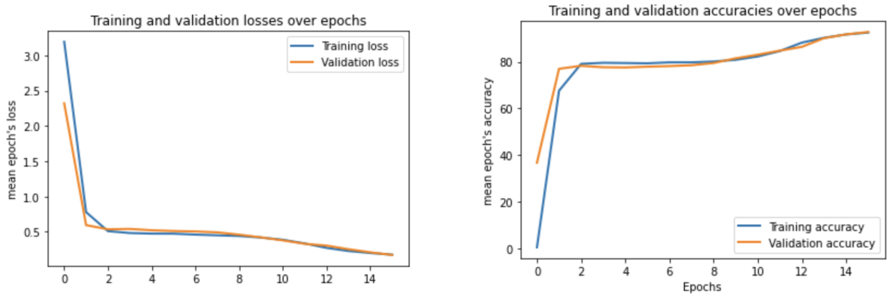

Figure 3: Training and Validation plots with first configuration

My second configuration achieves over then 95% of accuracy in 10 min when training on GPU. I use the following hyper-parameters:

- batch size = 128

- accumulating gradients for 10 steps
- learning rate = 0.0001
- Gradient clipping rate = 0.1
- Model with 2 encoder layers, 1 decoder layers, hidden dimension of 256, feed-forward dimension of 1024, using 8 attention heads.

Figure 4 shows my results.

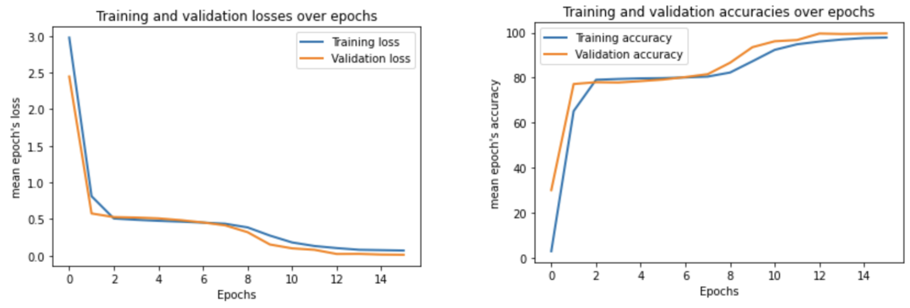

Figure 4: Training and Validation plots with second configuration

7.4

7.5