Deep Learning Lab: Report Assignment #4

Marco Latella
marco.latella@usi.ch

January 13, 2022

1 Problem Setups and Preliminaries

1.1 From the dataset Number_Place Dataset, I have gathered the following properties:

Number_Place Dataset		
	Train	**Valid**
Questions	1999998	10000
Answers	1999998	10000
Questions mean len	38.22	40.32
Answers mean len	1.0	1.0
Quest Chars	76429949	403219
Ans Chars	1999998	10000

2 Dataloader

2.1 The implementation uses different vocabularies for the source and target.

2.2 When using the get_idx() method, with the extend_vocab parameter equal to False, to return the id of the string passed, it will return the unknown ID if the vocabulary does not contain the ID representation of the string.

3 Model

3.1 The model has been implemented as required.

4 Greedy Search

4.1 The Greedy Search algorithm has been implemented as required.

4.2 Taking as an example another type of transformers implementation such as GPT-2, it offers several methods to perform search tasks, giving the possibility to specify all the needed parameters to execute the desired algorithm. It allows performing Greedy search, Beam search, Top-K Sampling, etc. An implementation of nn.Transformers that provide the same functions offered by GPT-2 would have been better.

4.3 The stopping criteria have been implemented as required.

4.4 The Bach Mode Evaluation has been implemented as required.

5 Accuracy Computation

5.1 The method to compute the accuracy has been implemented as required.

6 Training

6.1 Has been used Cross-Entropy Loss since it allows to measure the distance between the output probability distribution calculated by the softmax and the truth value represented by the one-hot encoded vector of the target.

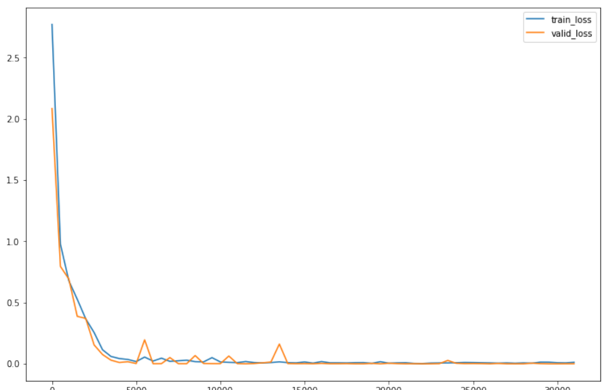
6.2 The training code has been implemented as required. The losses and the accuracies have been monitored every 500 steps for the Number Place Value dataset and every 1000 steps for the other two.

6.3 The Gradient Accumulation has been implemented as required obtaining an effective batch size of 640.

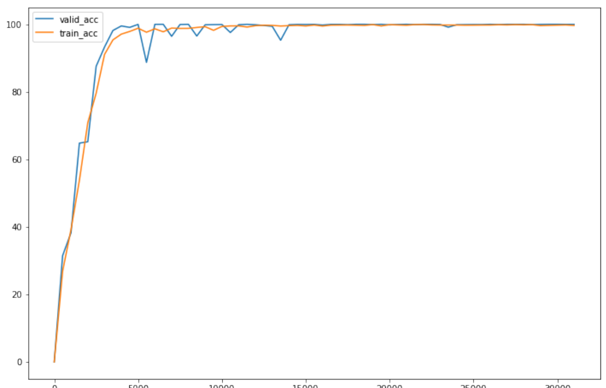
7 Experiments

7.1 The training pipeline has been implemented as required.

7.2 I run the training process for an entire epoch, even though the accuracy reached 100% first.
The validation accuracy and loss have been sampled every 500 steps.



(a) Losses



(b) Accuracies

Figure 1: Results obtained on Number Place dataset

Looking at the figure you can see that the learning curves go down quickly. In the same way, the accuracy rises just as fast. This means that the model can easily learn the needed notions to solve this task. Moreover, considering the figures, you can see that the model behaves very well, without overfit or underfit. This also indicates that the complexity of the model is not unbalanced.
Below are reported the predictions of the model at different training steps, to highlight how the learning improves.

Train step 1000 - Valid Acc: 34.29%

Questions
What is the ten thousands digit of 92815389?
What is the millions digit of 48188461?
What is the hundred thousands digit of 29115212?

Predicted Answer
2
1
2

Train step 3500 - Train Acc: 60.07%

Questions
What is the ten thousands digit of 92815389?
What is the millions digit of 48188461?
What is the hundred thousands digit of 29115212?
Predicted Answer
9
8
1

Train step 10500 - Valid Acc: 100.00%

Questions
What is the ten thousands digit of 92815389?
What is the millions digit of 48188461?
What is the hundred thousands digit of 29115212?
Predicted Answer
1
8
1

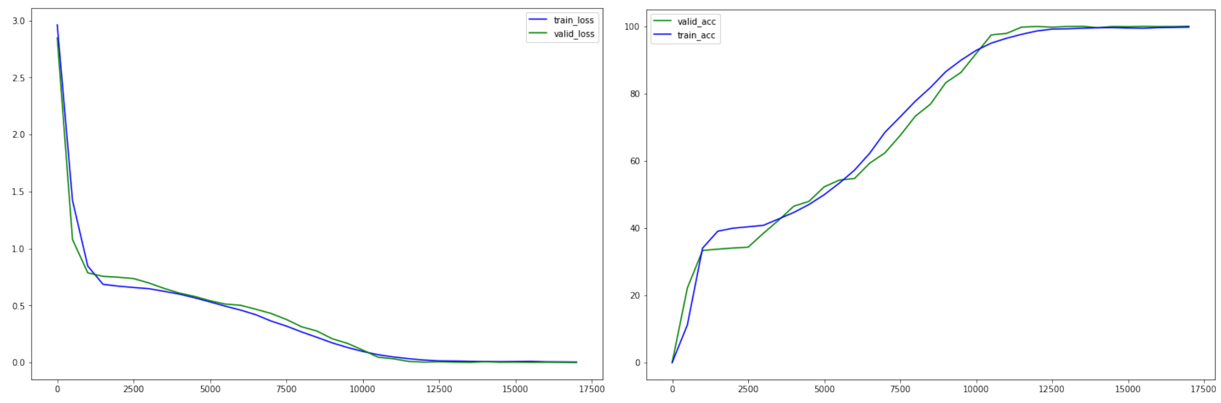
Reasoning on the Questions and Predicted Answers reported, you can see that the model's learning is very quick. After only 3500 training steps it can predict well the 60% of the answers, reaching rapidly the 100% of accuracy.

7.3 As required has been tried different combinations of hyperparameters.

Combination 1

- Learning rate: 0.0001
- Hidden layer size: 256
- Number heads self attention: 8
- Dimension Feed Forward: 512
- Number layers encoder: 1
- Number layers decoder: 1

Running the test on the **Number place value** task, has been obtained the following results.



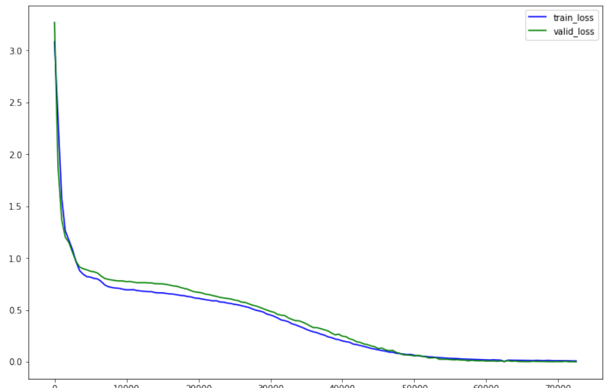
(a) Losses (b) Accuracies

Figure 2: Results obtained on Number place value task

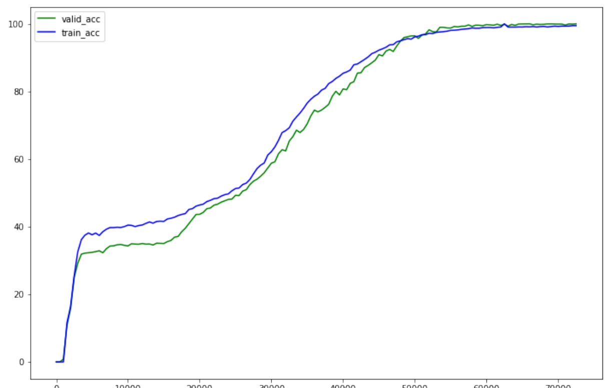
Using combination 1, no degradation has been noticed. The only difference encountered, unlike the default combination, regards the number of steps to reach 100% of accuracy on the validation set. It reaches 100% in about 17000 steps, instead of 10500.

Combination 2

- Learning rate: 0.0001
- Hidden layer size: 64
- Number heads self attention: 4
- Dimension Feed Forward: 128
- Number layers encoder: 1
- Number layers decoder: 1



(a) Losses



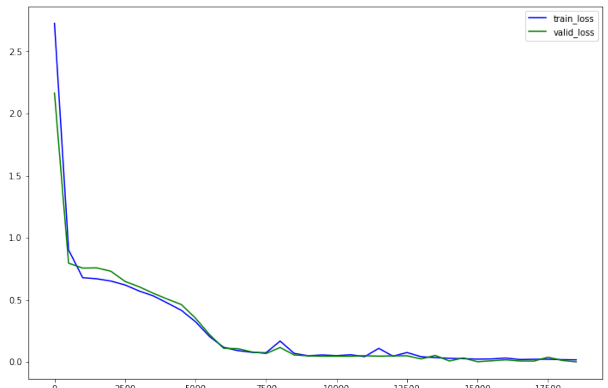
(b) Accuracies

Figure 3: Results obtained on Number place value task

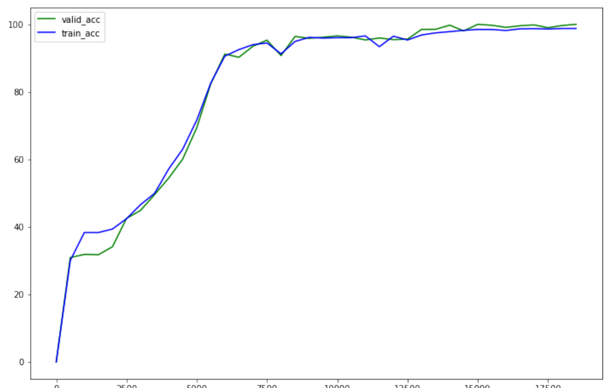
Using combination 2, as in the previous combination, there was no degradation in terms of accuracy. However, this time the model took a lot of steps more to reach 100% of accuracy, about 3 epochs.

Combination 3

- Learning rate: 0.0001
- Hidden layer size: 1024
- Number heads self attention: 8
- Dimension Feed Forward: 10204
- Number layers encoder: 1
- Number layers decoder: 1



(a) Losses

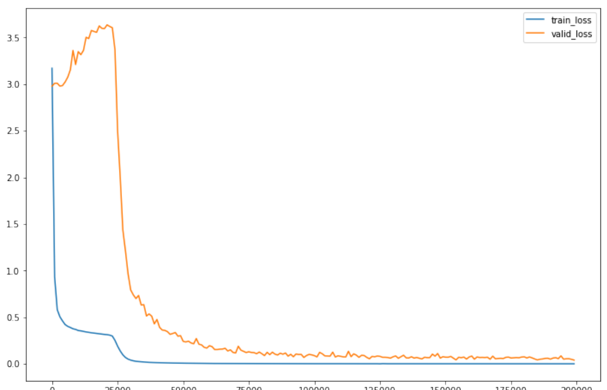


(b) Accuracies

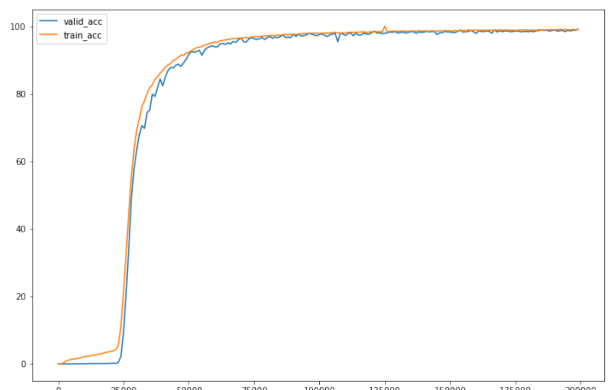
Figure 4: Results obtained on Number place value task

Using combination 3 there was no degradation in Accuracy. Once again the only aspect affected was the number of steps to reach 100%.

7.4 The figures below show the results obtained on **Compare Sort** task. It has been trained until the Validation accuracy reached 99%.



(a) Losses



(b) Accuracies

Figure 5: Results obtained on compare - sort task

Compared to the **Number Place Value** task, this one looks harder to solve for the model. To reach a reasonable accuracy value it took more time and more epochs. This behavior can be explained by reasoning on the notions it has to learn. Unlike the first task, the model has to acquire more type of information from the context, such as the difference between sorting in a decreasing or increasing way, handling the changing length of the list to sort, using the given numbers instead of others numbers, the difference between negative and positive numbers, etc...

All these types of notions require more effort from the model to be understood.

The growth of the learning can be seen in the following predictions.
After 1000 training steps, the model is only able to print a few values contained in the corresponding list.
Reaching an Accuracy of 24.00% it started to understand, with some errors, which number to use and learn a bit of notion of sorting.
At 92.48% the model started to answer correctly most of the questions.

Train step 1000 - Valid Acc: 0.00%

Questions
Sort 3, 52126, -1/5, -3.4, -0.1 in increasing order.
Put 5, -10, -1361.7, 1/5 in decreasing order.
Sort 178, 0.5, -3, 2/7, -1053, 2.
Predicted Answer
-13, -2/5, -1/5, -1/5, -1
11, 0.5, -1/1, -1/5
-17, -0.5, -0.5, 1, 2, 2/7

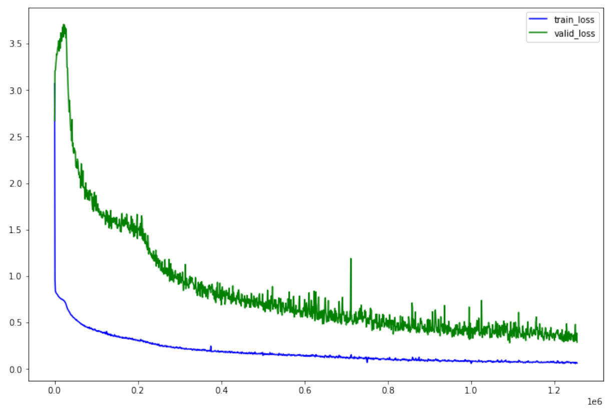
Train step 26000 - Valid Acc: 24.00%

Questions
Sort 3, 52126, -1/5, -3.4, -0.1 in increasing order.
Put 5, -10, -1361.7, 1/5 in decreasing order.
Sort 178, 0.5, -3, 2/7, -1053, 2.
Predicted Answer
-3.4, -0.1, -1/5, 3, 25126
5, 1/5, -10.671, -13
-105, -3, 0.5, 2/7, 27, 173

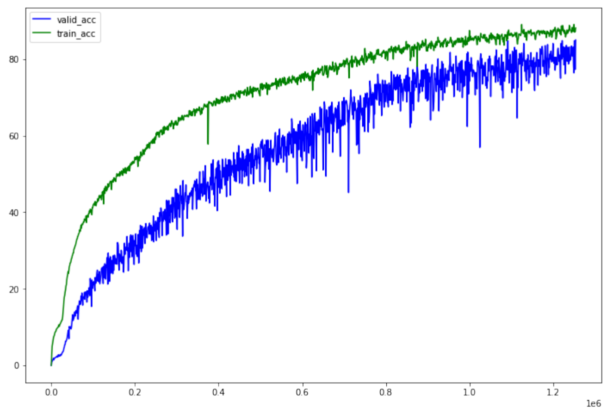
Train step 61000 - Valid Acc: 92.48%

Questions
Sort 3, 52126, -1/5, -3.4, -0.1 in increasing order.
Put 5, -10, -1361.7, 1/5 in decreasing order.
Sort 178, 0.5, -3, 2/7, -1053, 2.
Predicted Answer
-3.4, -1/5, -0.1, 3, 52126
5, 1/5, -10, -1361.7
-1053, -3, 2/7, 0.5, 2, 178

7.5 The figures below show the results obtained on the **Algebra - Linear 1d** task.



(a) Losses



(b) Accuracies

Figure 6: Results obtained on algebra - linear 1d task

The model has trained approximately for 10 hours and then was stopped, reaching an accuracy of 85%, but looking at its trend you can assume that could have reached also 90% or above.

As you can see in the figures, this task was really hard to solve, but the figures show that it could be able to obtain fairly good results. During the training there were several peaks in the validation accuracy, however, it seems to get more stable when the validation loss becomes lower.

Trying to think about what can be done to improve the performance on this task, you could increase the number of heads of the multi-head attention mechanism in order to get a richer interpretation of the information embedded in the embedding vectors.