Mathematical Problem Solving with Transformer

Christian Altrichter

January 15, 2023

Question 1 - Problem Setups and Preliminaries [8/100 points]

Question 1.1

Here the following statistics:

1. train.x
   (a) Number of sentences: 1.999.998
   (b) Number of characters: 76.429.949
   (c) Avg lenght: 38.34002857240042
2. train.y
   (a) Number of sentences: 1.999.998
   (b) Number of characters: 1.999.998
   (c) Avg lenght: 1
3. interpolate.x
   (a) Number of sentences: 10.000
   (b) Number of characters: 403.219
   (c) Avg lenght: 40.3219
4. interpolate.y
   (a) Number of sentences: 10.000
   (b) Number of characters: 10.000
   (c) Avg lenght: 1

Question 2 - Dataloader [2/100 points]

Question 2.1

No, by default they use two different vocabularies.

Question 2.2

The unknown token will be placed iff the vocabulary may not be extended. Thus, placing any additional vocabulary with an unknown token.

Question 3 - Model [5/100 points]

Question 3.1

The function "forward seperate" has been implemented in the source code.

Question 4 - Greedy Search [25/100 points]

Question 4.1

The greedy search has been implemented in the source code.

Question 4.2 (Bonus)

The challenge with transformers and search algorithms is that the search algorithms favor the outcome with the highest probability. Thus, in the first instance, this creates a problem in terms of diversity of the outcome. The transformer includes and connects self attention to the outcomes probabilities, and therefore learns the distribution for extracting elements by itself.

Question 4.3

Here, the stopping criteria is the token <eos> which has the token value "2" in our application. Second, we can expect always a one token answer. Thus, here we can stop after a length of three composed of the following elements in consecutive order: <sos>, token (e.g. the answer), <eos>.

Question 4.4

I have not fully understood how to implement this, thus, it has not been implemented.

Question 5 - Accuracy Computation [5/100 points]

Question 5.1

The function has been implemented in the source code.

Question 6 - Training [22/100 points]

Question 6.1

I have used the cross entropy loss function as this is a widely used common practice when dealing with classification predictions using single labels and multiple classes.

Question 6.2

The training code has been implemented in the source code. Here, the following functions were used to help in the training:

- "remove eos()": To adjust the target and replace all eos id's with padding indices (in particular needed to train effectively for question 7.4 and 7.5 as the output is not in a standardized output length anymore).
- "cal accuracy()": As requested by question 5 to aid with the computations of accuracy for both training and validation.

Question 6.3

The gradient accumulation has been implemented in the source code.

Question 7 - Experiments [33/100 points]

Question 7.1

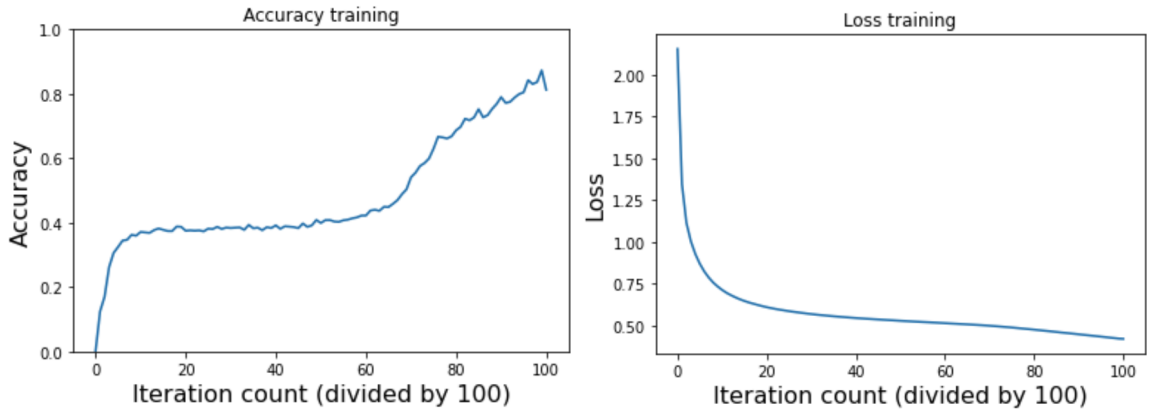
Here, the stopping criteria for the model was once the validation accuracy reaches more than 90.0% accuracy.

The training has been absolved accordingly with a validation accuracy of 0.9166666666666666.

Question 7.2

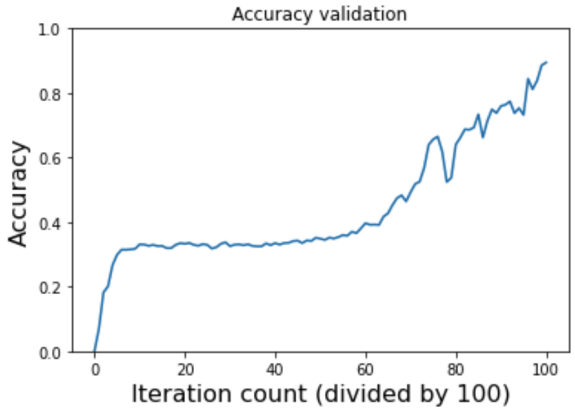
The following depicts the training loop metrics. Here, the model has completed with:

- Training accuracy: 0.8125
- Training loss: 0.4198666865959555
- **Validation accuracy: 9166666666666666**
- Validation loss: 0.0971173751526154

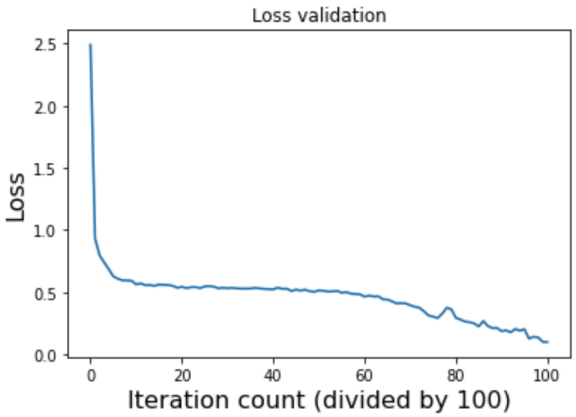


(a) Training accuracy

(b) Training loss



(c) Validation accuracy



(d) Validation loss

Figure 1: Training and validation metrics

Here the following question, answers and predictions for of three validation entries. As it can be seen, the output matches the predictions in 2 out of 3 cases:

```
Question  0 : What is the hundred thousands digit of 22225209?

Answer  0 : 2
<eos>
Prediction 0  : 2
<eos>
Question  1 : What is the ten millions digit of 82446846?
<pad><pad><pad><pad><pad>
Answer  1 : 8
<eos>
Prediction 1  : 8
<eos>
Question  2 : What is the millions digit of 24377448?
<pad><pad><pad><pad><pad><pad><pad><pad><pad>
Answer  2 : 4
<eos>
Prediction 2  : 3
<eos>
```

Figure 2: Output for number place with the trained model

Question 7.3

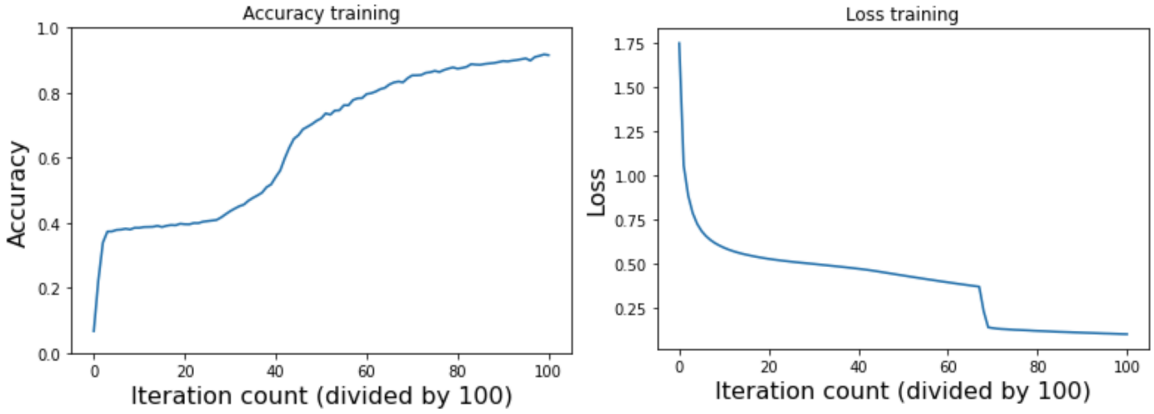
Question 7.3.1 - first configuration

The following hyper parameters have been changed / reduced in the first configuration:

- Nr. of encoding layers: changed from 3 to 1.
- Nr. of decoding layers: changed from 2 to 1.

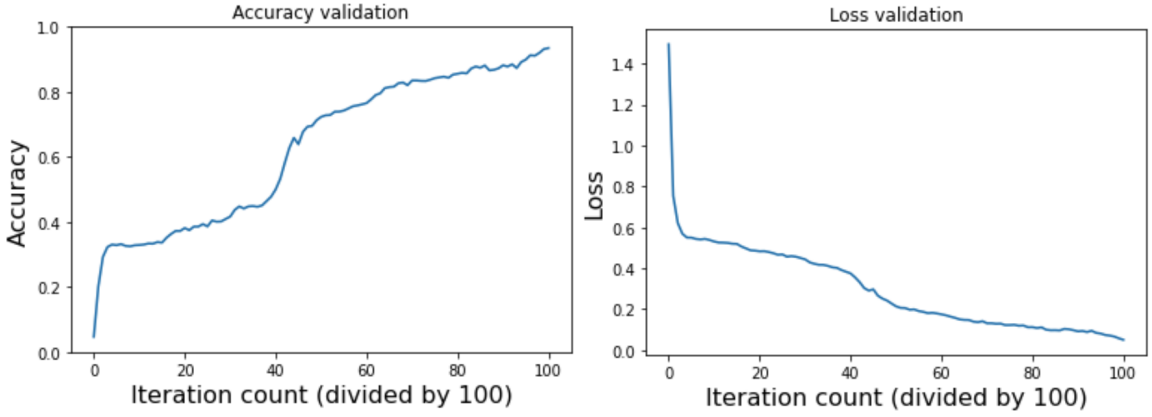
The training has been absolved in a slightly longer time:
The final values are as follows:

- Training accuracy: 0.8645833333333334
- Training loss: 0.4417515738974541
- **Validation accuracy: 0.94958**
- Validation loss: 0.33394190745475966



(a) Training accuracy

(b) Training loss



(c) Validation accuracy

(d) Validation loss

Figure 3: Training and validation metrics

Question 7.3.2 - second configuration

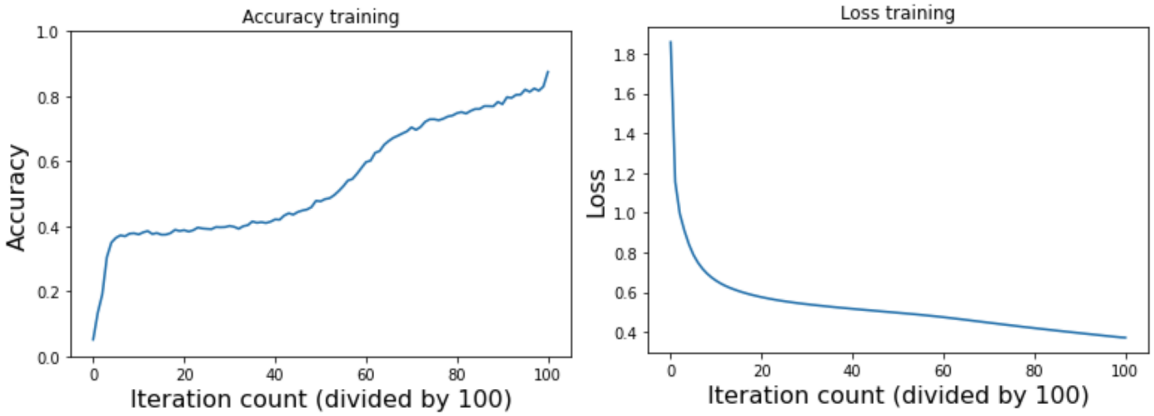
The following hyper parameters have been changed / reduced in the second configuration:

- Dim feed forward changed from 1024 to 512.

The training has been absolved in a slightly longer time. The final values are as follows:

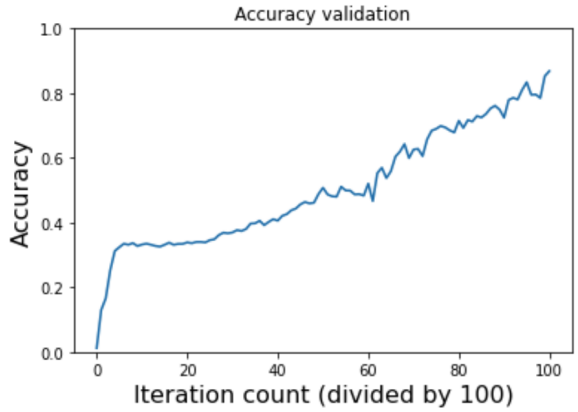
- Training accuracy: 0.875
- Training loss: 0.37212073448163113
- **Validation accuracy: 1.0**
- Validation loss: 0.12981492562744862

Despite having a stopping condition for a validation at 90.0% validation accuracy, the model incurred a massive jump at the end, leading to a validation accuracy of 100.0%.

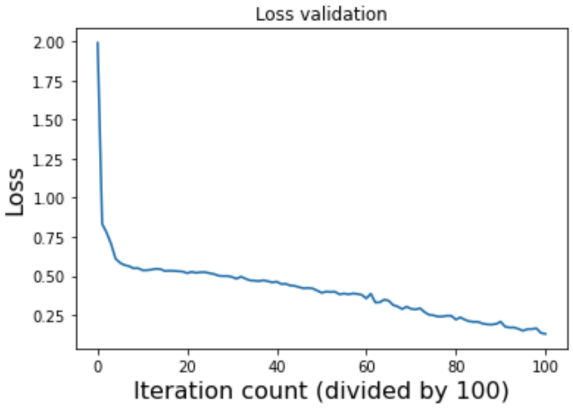


(a) Training accuracy

(b) Training loss



(c) Validation accuracy



(d) Validation loss

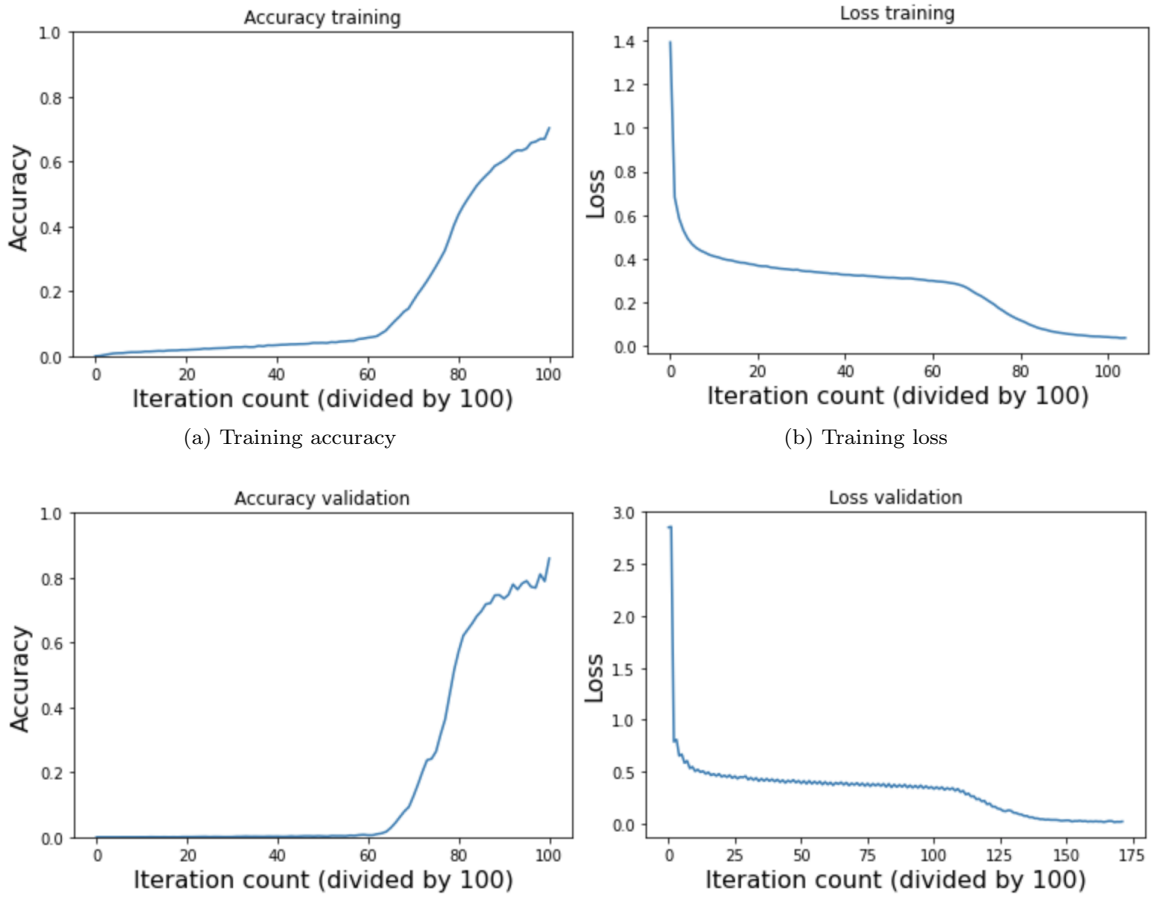
Figure 4: Training and validation metrics

Question 7.4

Comparison sort differs to the previous number place document in terms of input complexity and variety.

The training has completed after approximately 3 hours and 20 minutes with the final values as follows:

- Training accuracy: 0.703125
- Training loss: 0.03884751064702868
- **Validation accuracy: 0.90185**
- Validation loss: 0.021611989559605718



(a) Training accuracy

(b) Training loss

(c) Validation accuracy

(d) Validation loss

Figure 5: Training and validation metrics

Here the following question, answers and predictions for of three validation entries. As it can be seen, the output matches the predictions in 2 out of 3 cases. In the one case where the prediction was wrong, it came close to the solution:

```
torch.Size([32, 64])
Question  0 : Put 0.4, 5, 30, 50, -2, 16 in descending order.
<pad><pad><pad><pad><pad><pad><pad><pad><pad><pad>
Answer  0 : 50, 30, 16, 5, 0.4, -2
<eos><pad><pad><pad><pad><pad><pad><pad><pad>
Prediction 0  : 50, 10, 16, 5, 0.4, -2
<eos><pad><pad><pad><pad><pad><pad><pad><pad>
Question  1 : Put 3/4, -217, -15/2, 2/5 in descending order.
<pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad>
Answer  1 : 3/4, 2/5, -15/2, -217
<eos><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad>
Prediction 1  : 3/4, 2/5, -15/2, -217
<eos><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad>
Question  2 : Put 70, 102018, -5 in descending order.
<pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad>
Answer  2 : 102018, 70, -5
<eos><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad>
Prediction 2  : 102018, 70, -5
<eos><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad>
```

Figure 6: Output for comparison sort with the trained model

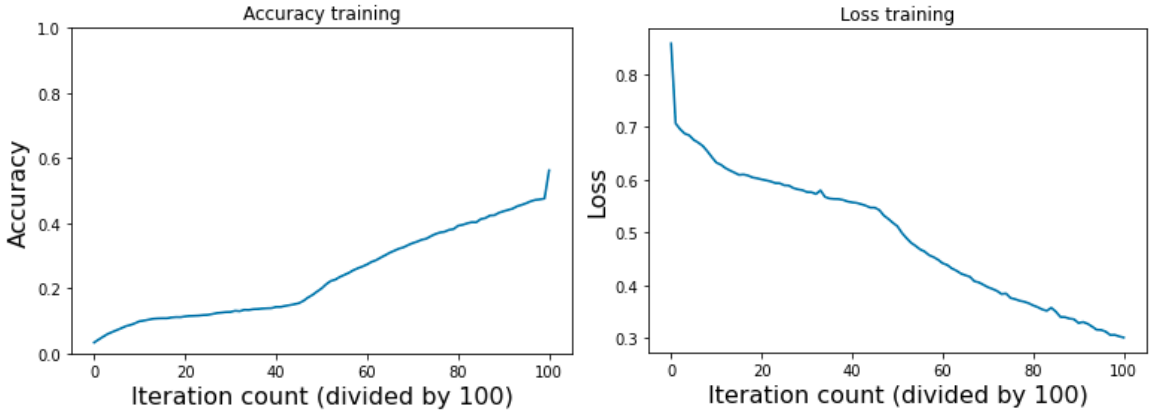
Question 7.5 (Bonus)

The training model has been adjusted to the standard transformer parameters due to higher complexity in input and output. Thus, with the following changed parameters:

- d model = from 256 to 512
- encoder layers = from 3 to 6
- decoder layers = from 2 to 6
- dim forward = from 1024 to 2048

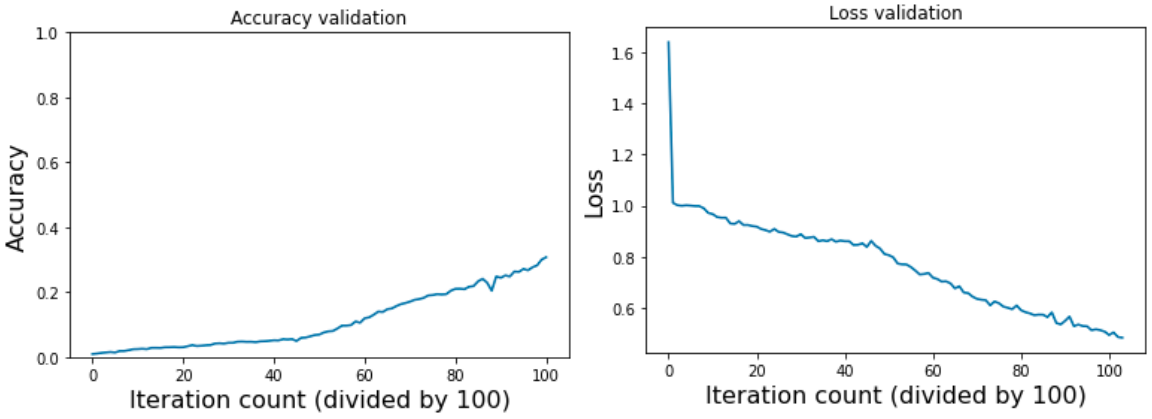
This has been done to help with learning additional complexity of the model. I was aware that this leads to a longer training of the model, however, once it grasps the structure, the learning curve is expected to steepen. Due to the lenght of the training, google colab randomly removed the GPU at its own will. Therefore, the "save state()" and "load state()" function have been implemented to resume training after its last crash.

- Training accuracy: 0.546875
- Training loss: 0.300616359859705
- **Validation accuracy: 0.3053125**
- Validation loss: 0.4529990041255951



(a) Training accuracy

(b) Training loss



(c) Validation accuracy

(d) Validation loss

Figure 7: Training and validation metrics

Here the following question, answers and predictions for of three validation entries. Despite the training not being completed, the model achieves certain predictions correctly:

```
Question  1 : Solve 0*t - 33*t - 83 + 964 = -736 for t.
<pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad>
Answer  2 : 49
<eos><pad>
Prediction 3  : 31
<eos><pad>
Question  1 : Solve -22*k = 166 + 57 - 25 for k.
<pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad>
Answer  2 : -9
<eos><pad>
Prediction 3  : -9
<eos><pad>
Question  1 : Solve -39*o - 99*o + 67 = 111*o + 814 for o.
<pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad><pad>
Answer  2 : -3
<eos><pad>
Prediction 3  : -3
<eos><pad>
```

Figure 8: Output for linear algebra with the trained model