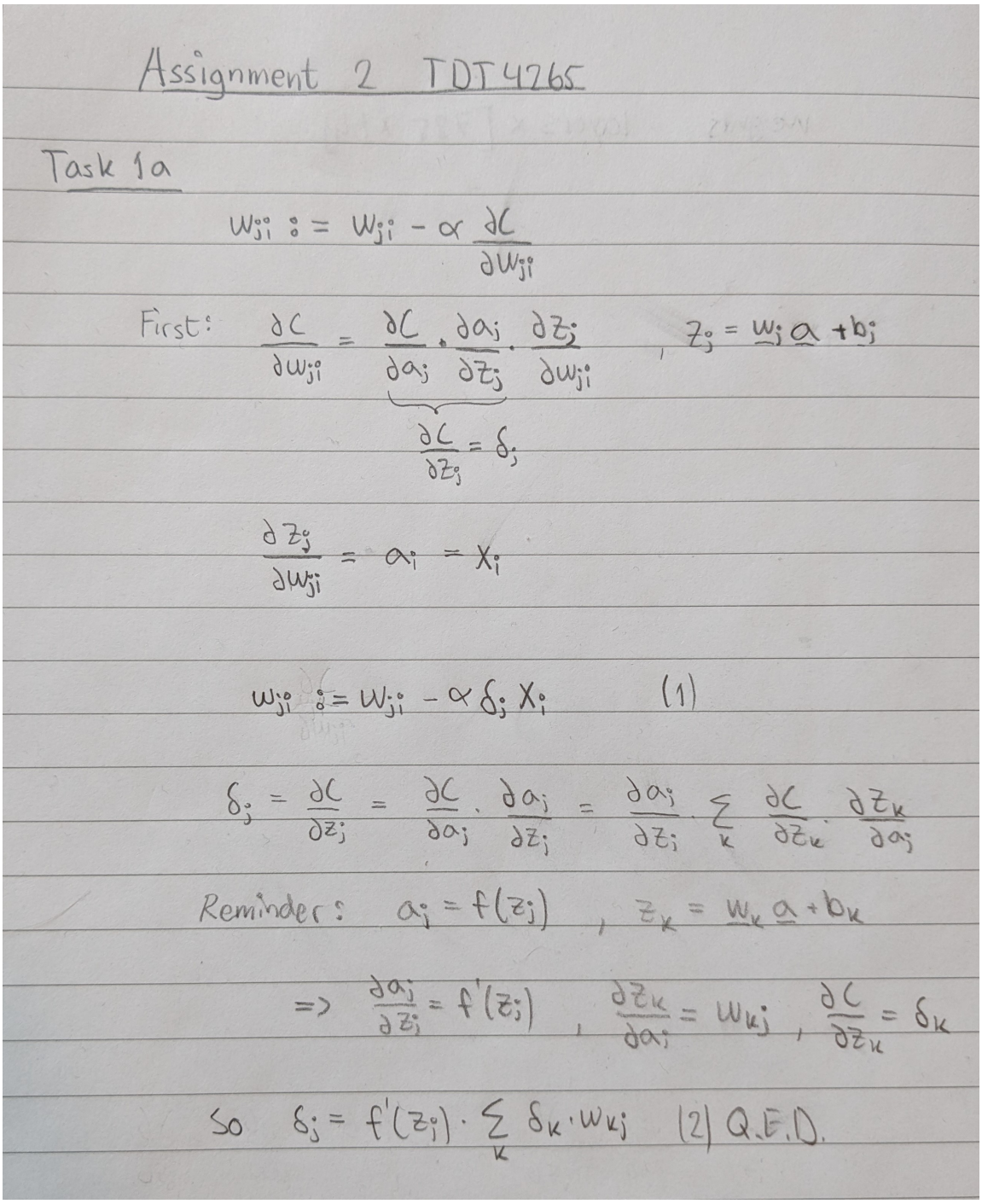
Assignment 2 Report Olof Ljunggren 2024-02-22

Task 1

Task 1a)



Assignment 2 TDT4265

Task 1a

$$w_{ji}^{\circ} := w_{ji} - \alpha \frac{\partial C}{\partial w_{ji}}$$

First: $\frac{\partial C}{\partial w_{ji}} = \frac{\partial C}{\partial a_j} \cdot \frac{\partial a_j}{\partial z_j} \cdot \frac{\partial z_j}{\partial w_{ji}}$, $z_j = w_j a + b_j$

$$\frac{\partial C}{\partial z_j} = \delta_j$$

$$\frac{\partial z_j}{\partial w_{ji}} = a_i = x_i$$

$$w_{ji}^{\circ} := w_{ji} - \alpha \delta_j x_i \quad (1)$$

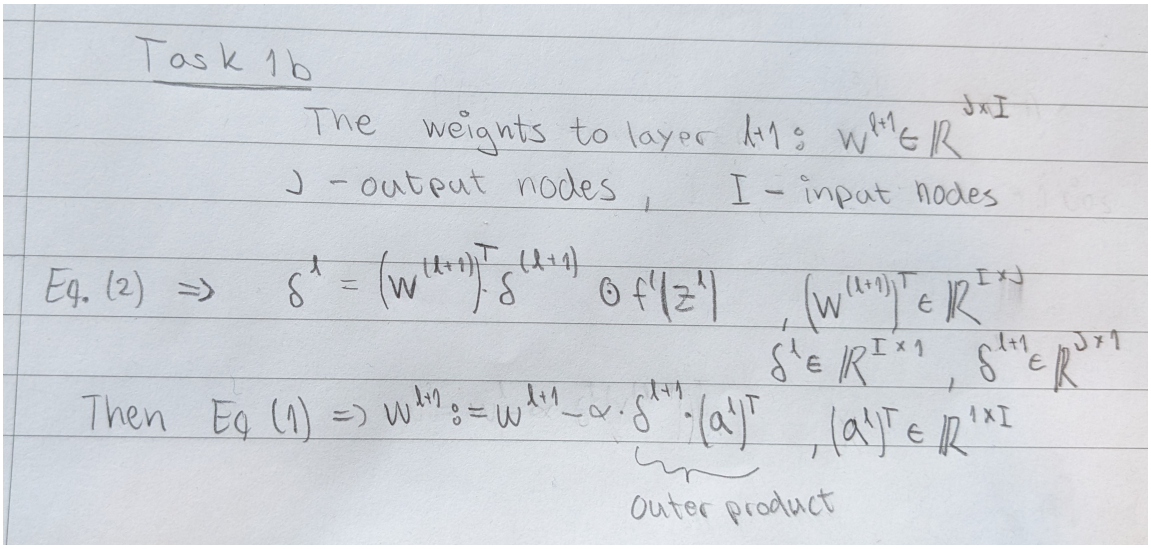
$$\delta_j = \frac{\partial C}{\partial z_j} = \frac{\partial C}{\partial a_j} \cdot \frac{\partial a_j}{\partial z_j} = \frac{\partial a_j}{\partial z_j} \sum_k \frac{\partial C}{\partial z_k} \cdot \frac{\partial z_k}{\partial a_j}$$

Reminder: $a_j = f(z_j)$, $z_k = w_k a + b_k$

$$\Rightarrow \frac{\partial a_j}{\partial z_j} = f'(z_j), \quad \frac{\partial z_k}{\partial a_j} = w_{kj}, \quad \frac{\partial C}{\partial z_k} = \delta_k$$

So $\delta_j = f'(z_j) \cdot \sum_k \delta_k \cdot w_{kj}$ (2) Q.E.D.

Task 1b)



Task 2

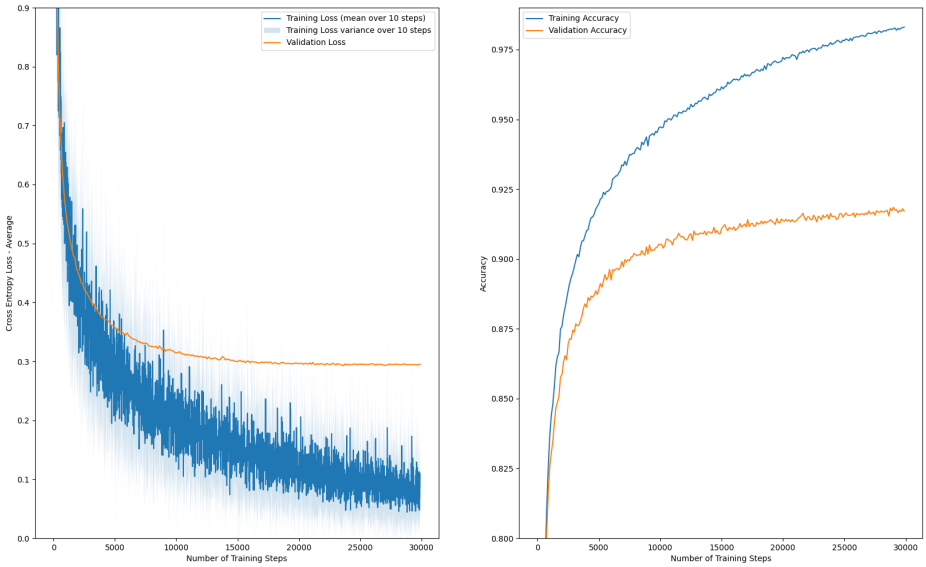
Task 2a)

A mean and standard deviation for the entire dataset used was calculated in the file normalization.py. There we found that mean = 33.55274553571429 and std = 78.87550070784701.

Task 2b)

Forward and backward functions were implemented. This was in a way that could manage multiple nodes.

Task 2c)



Task 2d)

In total there should be 285 *64* + *64* 10 weights = 18880. We also saved results from intermediate results for backward propagation, this is 32 *285* + *32* 64, minibatch with 32 numbers and 285 inputs to hidden layer and 64 inputs to output layer. But as learnable parameters we have 18880.

Task 3

In this third task I implemented better weight initialization, an improved sigmoid and finally gradient momentum.

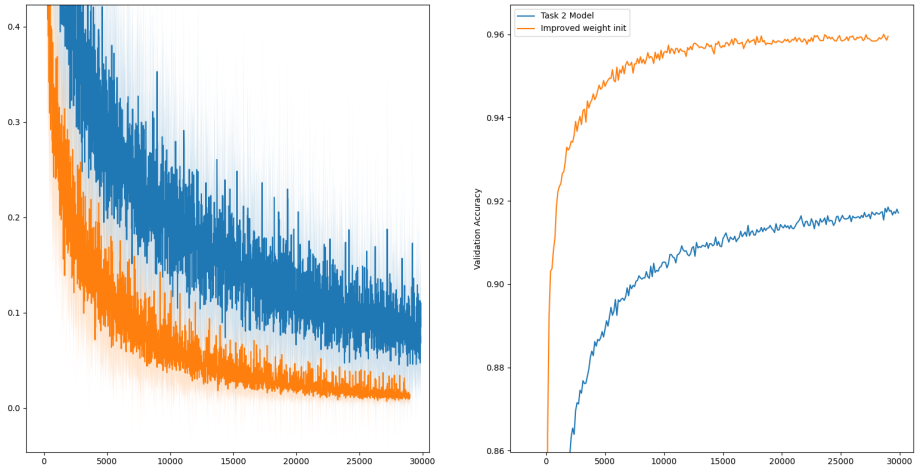
1. Improved weight initialization: as we can see in the plot the gaussian distributed weights gave quite better results. We can notice that the validation seems to stabilize around 10000 training steps instead of still improving around 30000 as in task 2c. Also its seems like the weights converged to a little bit better result, more accurate classification. We reached 95% accuracy in ~5000 repetitions.

2. Improved sigmoid: also here the convergence speed seemed to improved quite significantly. Now we reached 95% accuracy in ~2000 repetitions compared to ~5000 repetitions in the previous subtask. Also here the final accuracy seems to be a little bit better.

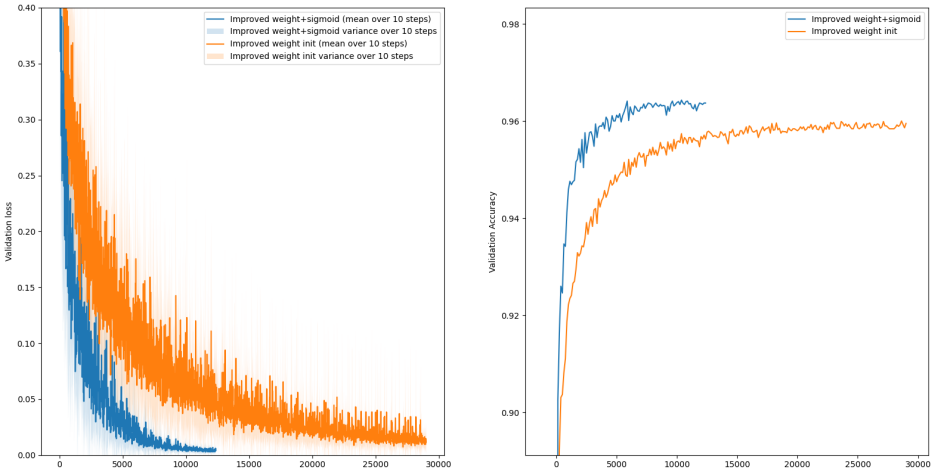
3. Momentum: the momentum further more improvec the convergence speed. Now we reached 95% accuracy in a little less than ~2000 repetitions.Here the final accuracy does not seem to improve. Almost the opposite, which kind of makes sense since we are not only using the correct gradient. This also gives a jumpy behavior. Though, in total the convergence were faster.

Finally all of these 3 we still have tendecies of overfitting. This is seen by looking at the large diffrence between training and validation accuracy. The reason for this is probably the number of nodes in the hidden layer in this case. But it is hard to draw conclusion now when we have not tested any other network topologies.

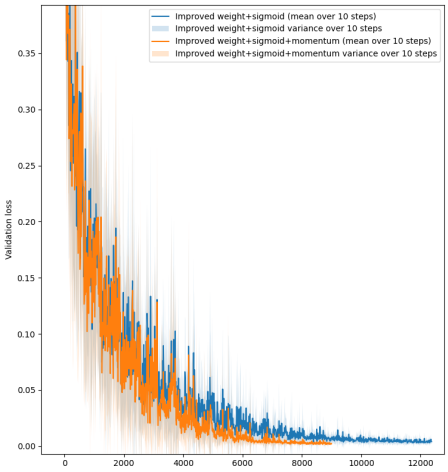
Task 3a)

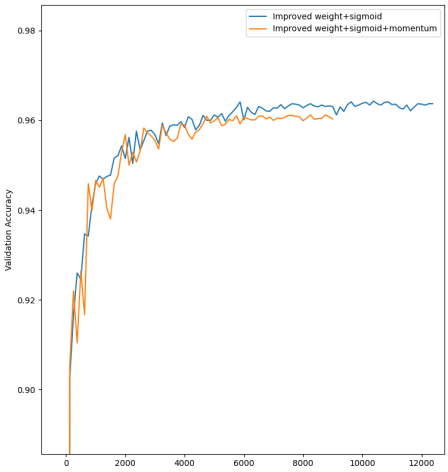

Task 2 Model
Improved weight init
Validation Accuracy

Task 3b)


Improved weight+sigmoid (mean over 10 steps)
Improved weight+sigmoid variance over 10 steps
Improved weight init (mean over 10 steps)
Improved weight init variance over 10 steps
Validation loss
Improved weight+sigmoid
Improved weight init
Validation Accuracy

Task 3c)

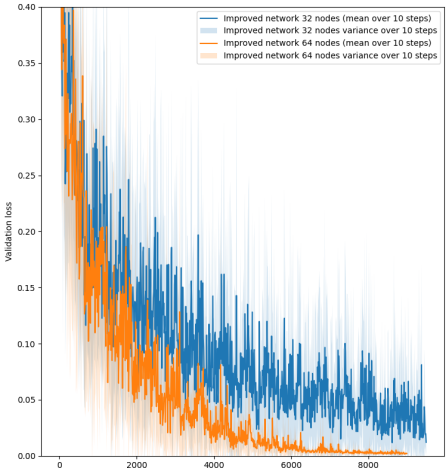


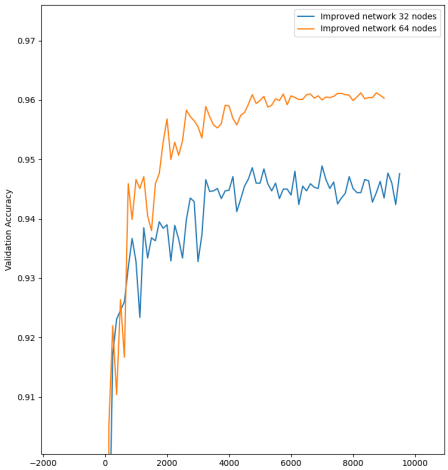


Task 4

Task 4a)

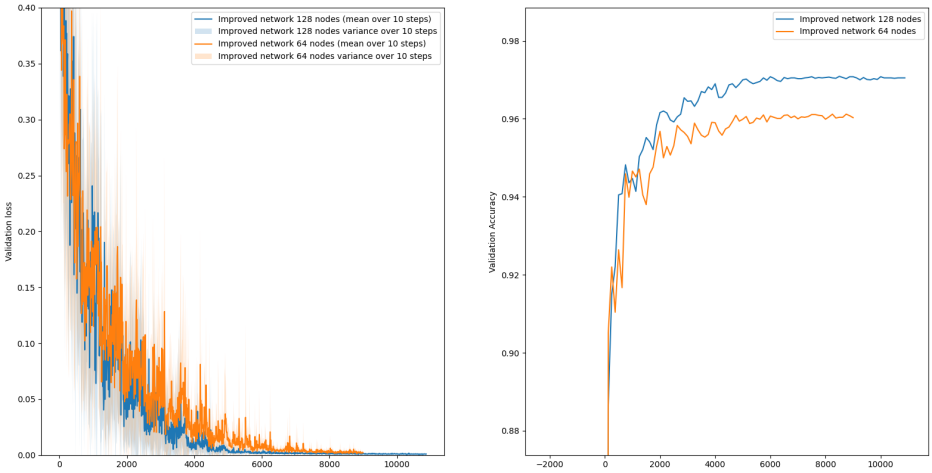
Using 32 nodes in the hidden layers gives worse result than for 64. We loose some accuracy.





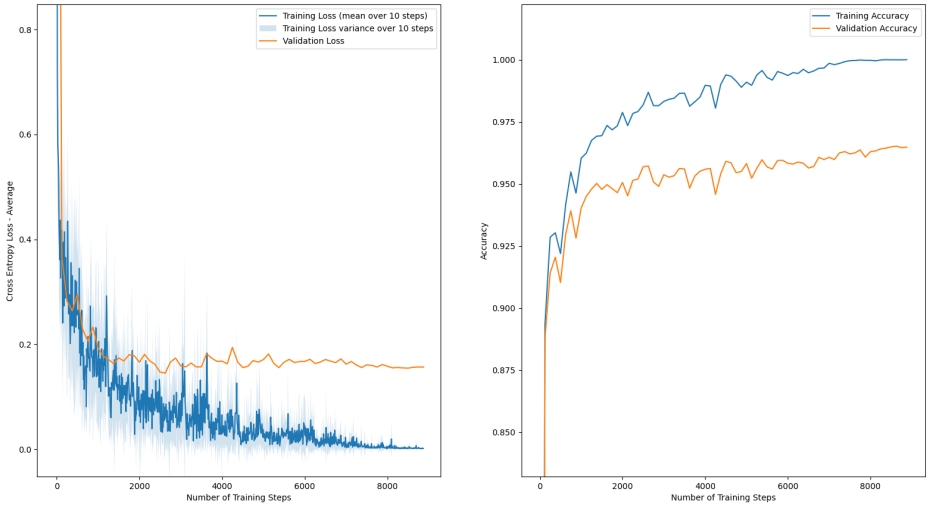
Task 4b)

Using 128 nodes in the hidden layers isntead of 64 does improve a little, but not as much as from 32 to 64. Hence, we have probably overfit the network a little bit.



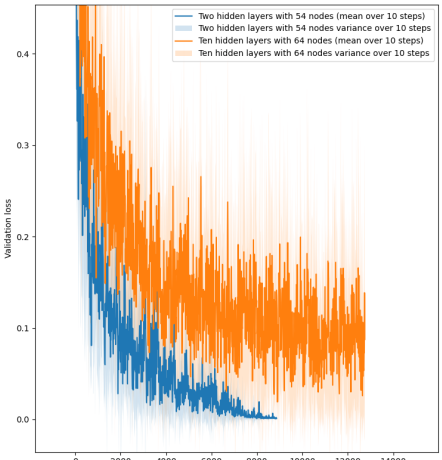
Task 4d)

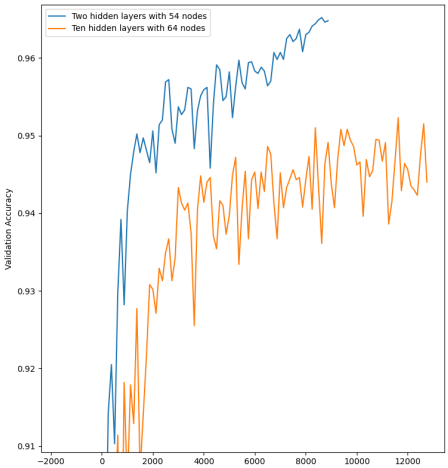
By using to equal size hidden layers with size 54 nodes gives 285 *54* + *54* 54 + 54 * 10 = 18846 which is a little fewer parameters than the task 3 case with 18880 parameters for one hidden layer with 64 nodes. In this case it seems like the results of the two networks are quite similar. Both converge to ~96% accuracy after ~8000 training steps.



Task 4e)

In this case we have a really severe case of overfitting. We fit the model alomst perfect onto our training data but we lose the generalization and therefore the performance on the validation set.


Two hidden layers with 54 nodes (mean over 10 steps)
Two hidden layers with 54 nodes variance over 10 steps
Ten hidden layers with 64 nodes (mean over 10 steps)
Ten hidden layers with 64 nodes variance over 10 steps
Validation loss
0.4
0.3
0.2
0.1
0.0
0
2000
4000
6000
8000
10000
12000
14000


Two hidden layers with 54 nodes
Ten hidden layers with 64 nodes
Validation Accuracy
0.96
0.95
0.94
0.93
0.92
0.91
−2000
0
2000
4000
6000
8000
10000
12000