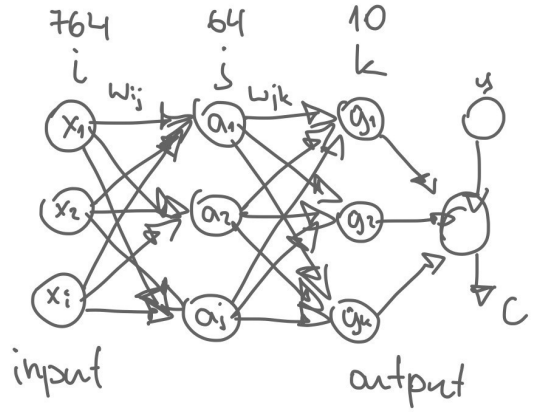
Assignment 2 Report

Task 1

task 1a)



sigmoid

$a_j = f(z_j)$ $\quad z_j = \sum_{i=0}^{I} w_{ji} x_i$

$\hat{y}_k = f(z_k)$ $\quad z_k = \sum_{j=0}^{J} w_{kj} a_j$

$f_{softmax}$

$$w_{kj} := w_{kj} - \alpha \frac{\partial C}{\partial w_{kj}} = w_{kj} - \alpha \delta_k a_j \qquad \delta_k = \frac{\partial C}{\partial z_k} = -(y_k - \hat{y}_k)$$

$$w_{ji} := w_{ji} - \alpha \frac{\partial C}{\partial w_{ji}} = w_{ji} - \alpha \delta_j x_i \qquad \delta_j = \frac{\partial C}{\partial z_j}$$

Task 1a

$$\frac{\partial C}{\partial w_{ji}} = \frac{\partial C}{\partial z_j} \cdot \frac{\partial z_j}{\partial w_{ji}} = \delta_j \cdot \frac{\partial z_j}{\partial w_{ji}} = \delta_j \cdot x_i$$

$$\Rightarrow w_{ji} := w_{ji} - \alpha \delta_j x_i$$

need sum since one j can influence all k

$$\delta_j = \frac{\partial C}{\partial z_j} = \sum_{k=0}^{K} \frac{\partial C}{\partial z_k} \cdot \frac{\partial z_k}{a_j} \cdot \frac{a_j}{z_j} = \sum_{k=0}^{K} \delta_k w_{kj} f'(z_j)$$

$$\delta_j = f'(z_j) \sum_{k=0}^{K} \delta_k w_{kj}$$

task 1b)

Task 1b

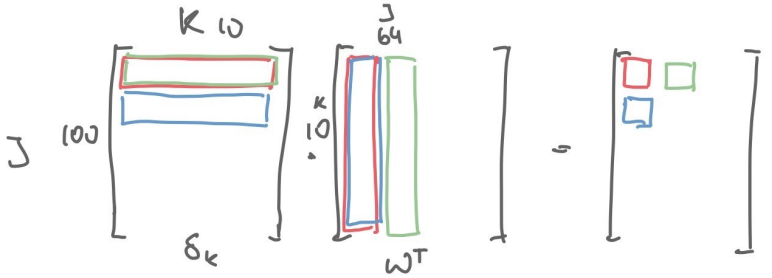
for batch size N=1

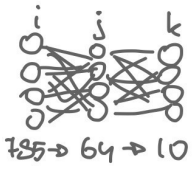
$$W := w - \alpha \cdot \delta \cdot X$$

$[I \times J] := [I \times J] - [1 \times 1] \cdot [I \times 1] \cdot [1 \times J]$ ← matricies sizes for i-j

$[J \times K]$ $[J \times K] - [1 \times 1] \cdot [J \times 1] \cdot [1 \times K]$ ← —"— j-k

$$\Rightarrow w := w - \alpha \cdot X^T \cdot \delta$$





$$\frac{\partial C}{\partial w_{kj}} = \frac{1}{N}\left(-A^T \cdot (y-\hat{y})\right)$$

$[64 \times 100] \cdot [100 \times 10] \rightarrow [64 \times 10]$

$\delta_k = -(y-\hat{y}) \rightarrow [100 \times 10]$

↑ batch size

$$w_{kj} := w_{kj} + \alpha \cdot \frac{\partial C}{\partial w_{kj}}$$

$[64 \times 10]$ $[64 \times 10]$

$$\delta_j = f'(z_j) \odot \delta_k \cdot w_{kj}^T$$

$[100 \times 64] \odot [100 \times 10] \cdot [10 \times 64] \rightarrow [100 \times 64]$

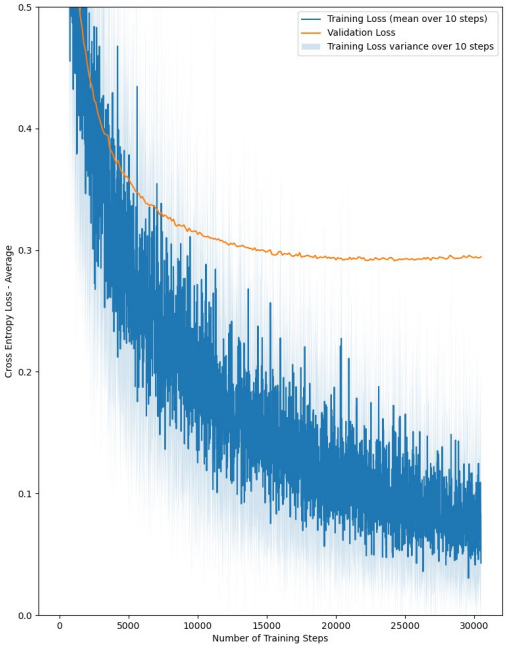
$$\frac{\partial C}{\partial w_{ji}} = \frac{1}{N}\left(X^T \cdot \delta_j\right)$$

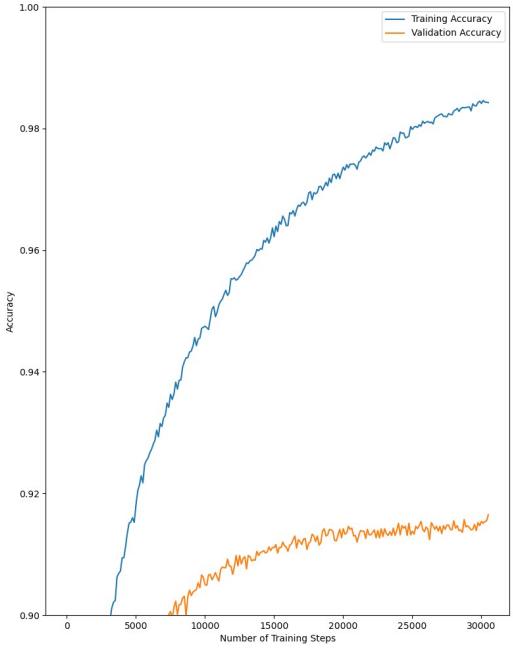
$[785 \times 100]$ $[100 \times 64] \rightarrow [785 \times 64]$

$$w_{ji} := w_{ji} + \alpha \frac{\partial C}{\partial w_{ji}}$$

Task 2

Task 2c)

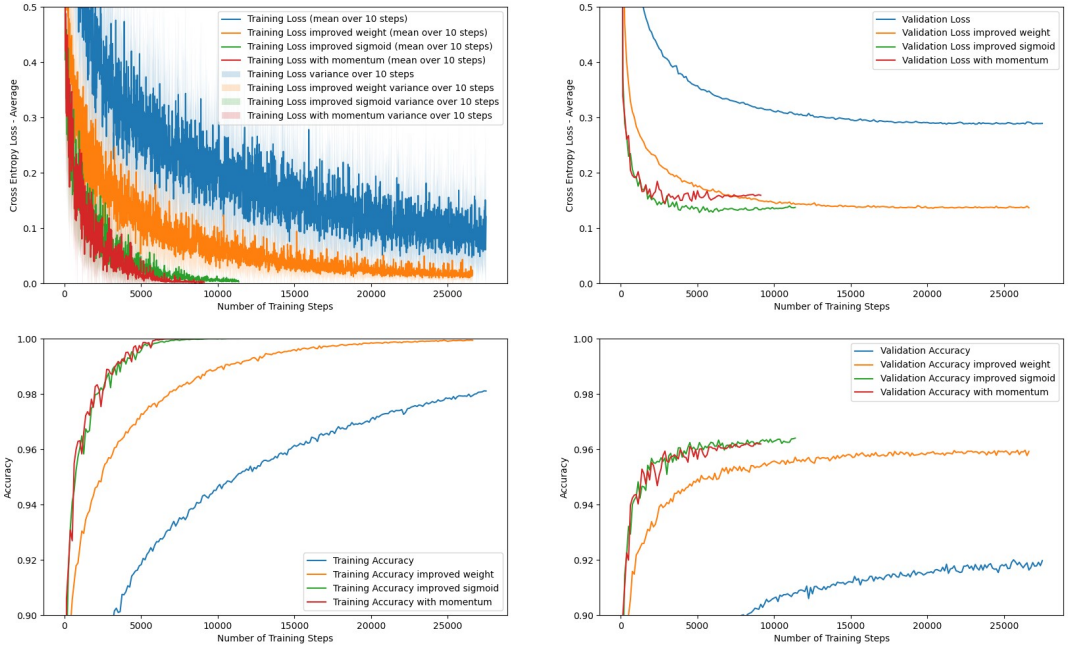




Task 2d)

In the first layer we have 784x64 weights and 64 biases, and in the second layer we have 64x10 weights, and we are not adding biases here. It will result in 785*64+64*10=50880 parameters, which is the same as sum of all elemts in weight matricies (since we are using the bias trick)

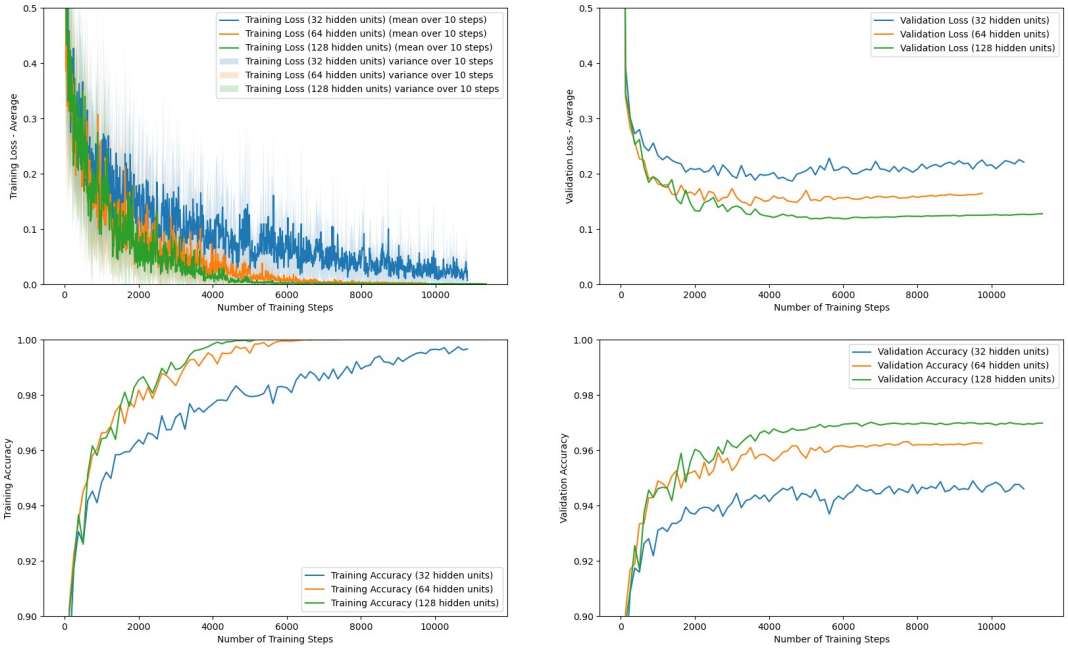
Task 3



In the image above we can see how adding the "tricks" changes the learning rate and accuracy of thge model. All changes can bee compared to the original model (blue). First I changed the weight initialization method (orange). This improved the end validation accuracy from 92% to about 95.5%. This model stopped learning at about the same epoch as the original one. Next I changed the sigmoid function (green). This lead to even higher validation accuracy at about 96% and the network stopped learning at epoch 18. We can also see that learning speed has improved aswell. We can however see that the network might overfit a bit, since the validation loss is slowly increasing after about 5000 training steps. The training accuracy of the network ends up at 100%. It might be that we should use a bigger dataset to be able to bring the validation accuracy closer to 100%. The last change was adding the momentum (red), when updating the weights. This resultet in very similiar to the previous one. The network stops learning a bit earlier at epoch 14. It also seems that the validation loss and accuracy is a bit more "noisy". Here the model also start to overfit slightly, as the validation loss increases after about 4000 training steps. However since we have early stopping the overfitting doesn't become a big problem. The end validation accuracy didn't increase this time.

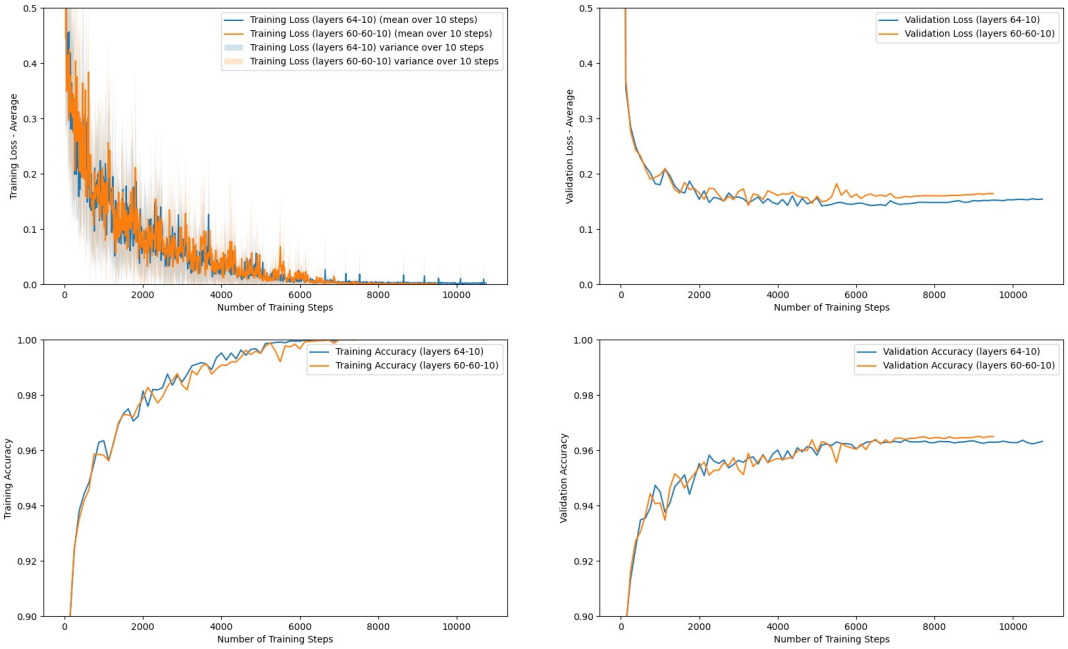
Task 4

Task 4ab)



We can see that the network with 32 hidden units, gives much worse results than the other two. Reason for that might be that the model is too simple and it is underfitting. However when we make model more complex with 128 hidden units, we can see improvement in the validation accuracy, without any obvious signs of overfitting.

Task 4d)

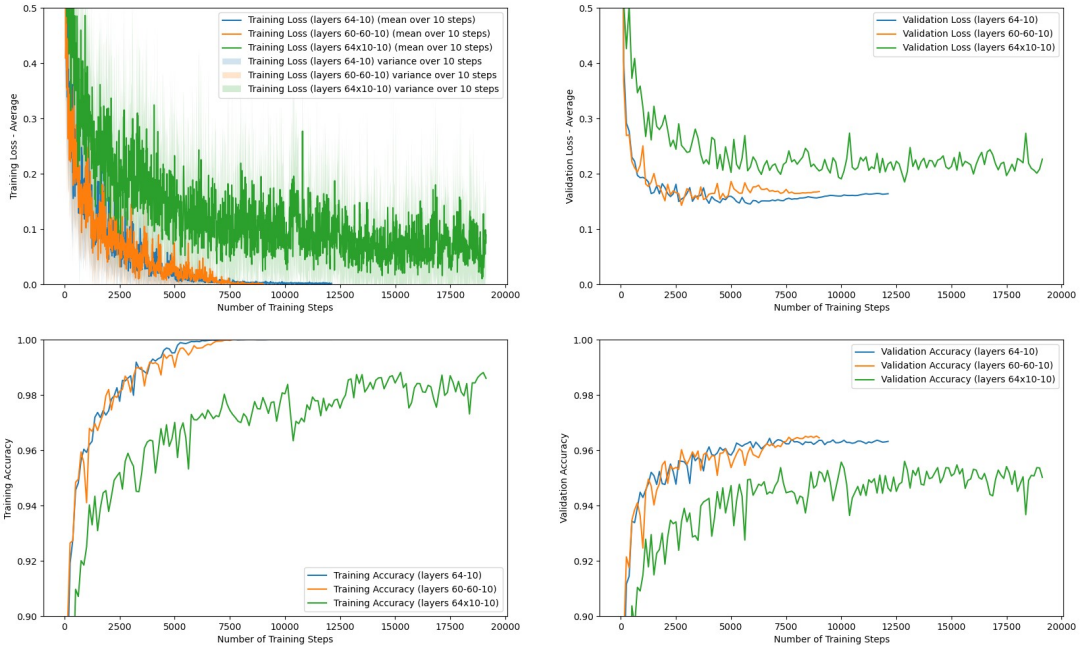


Here we have two neural networks:

- original with 64-10 units in two hidden layers, with 50880 parameters
- new network with 60-60-10 units in three layers, with 51300 parameters

As we can see the performance is very similiar, without any big benefits.

Task 4e)



Here we have three neural networks:

- original with 64-10 units in two hidden layers, with 50880 parameters
- three layer network with 60-60-10 units, with 51300 parameters
- new 11 layer network with 64x10-10 units, with 87744 parameters

As we can see the performence got clearly worse. The main reason is that our model is far too complex for this simple problem. The other reason might be "The Vanishing Gradient Problem", which is common in the network with a lot of layers. The main problem are the small derivatives of the sigmaoid function that are multiplied together in the chainrule in backpropagation to update the weights. That means that some layers will get very small updates, and the learning will not be efficient. We can see that in task 4b we increased number of nodes in a layer and got better performance, however increasing number of layers significantly had the opposite effect.