Assignment2Report

February 19, 2021

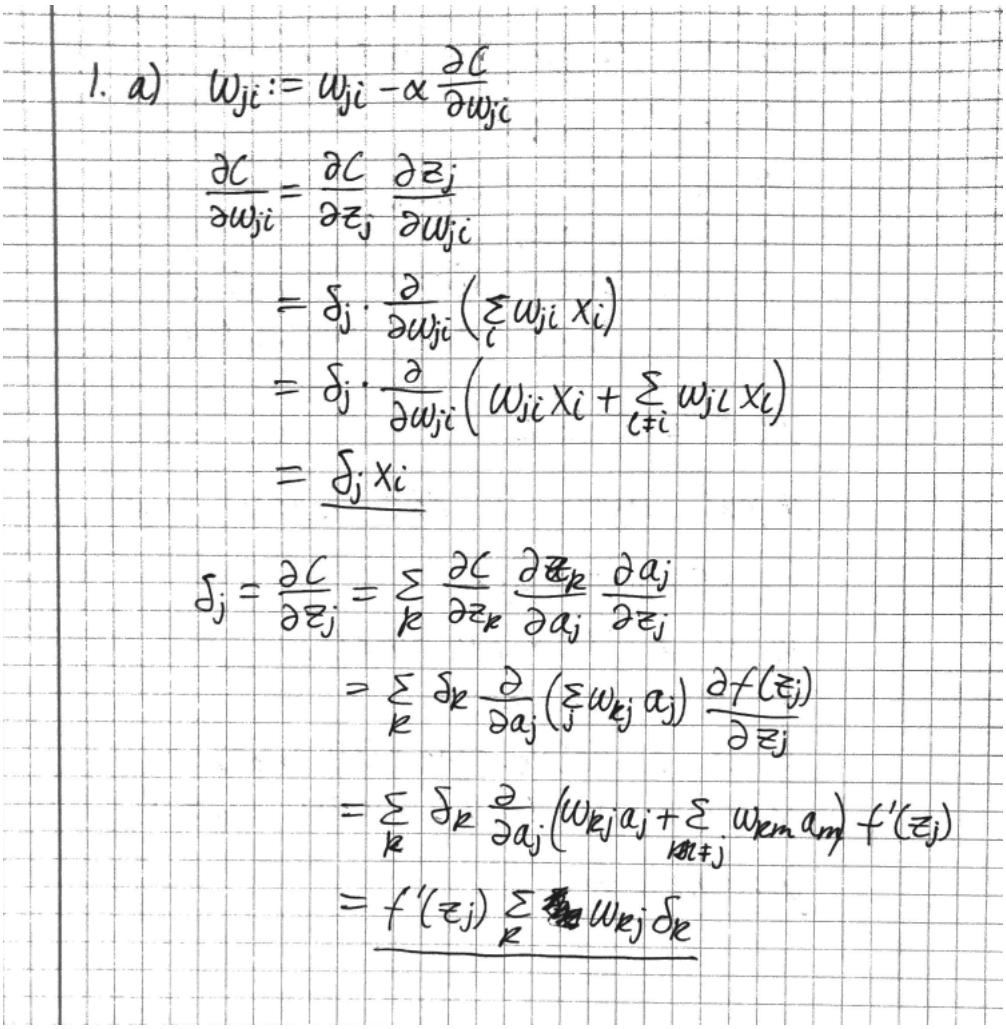
1 Assignment 2 Report

This is an outline for your report to ease the amount of work required to create your report. Jupyter notebook supports markdown, and I recommend you to check out this cheat sheet. If you are not familiar with markdown.

Before delivery, **remember to convert this file to PDF**. You can do it in two ways: 1. Print the webpage (ctrl+P or cmd+P) 2. Export with latex. This is somewhat more difficult, but you'll get somehwat of a "prettier" PDF. Go to File -> Download as -> PDF via LaTeX. You might have to install nbconvert and pandoc through conda; `conda install nbconvert pandoc`.

2 Task 1

2.1 task 1a)



1. a) $w_{ji} := w_{ji} - \alpha \frac{\partial C}{\partial w_{ji}}$

$$\frac{\partial C}{\partial w_{ji}} = \frac{\partial C}{\partial z_j} \frac{\partial z_j}{\partial w_{ji}}$$

$$= \delta_j \cdot \frac{\partial}{\partial w_{ji}} \left(\sum_i w_{ji} x_i \right)$$

$$= \delta_j \cdot \frac{\partial}{\partial w_{ji}} \left(w_{ji} x_i + \sum_{l \neq i} w_{jl} x_l \right)$$

$$= \underline{\delta_j x_i}$$

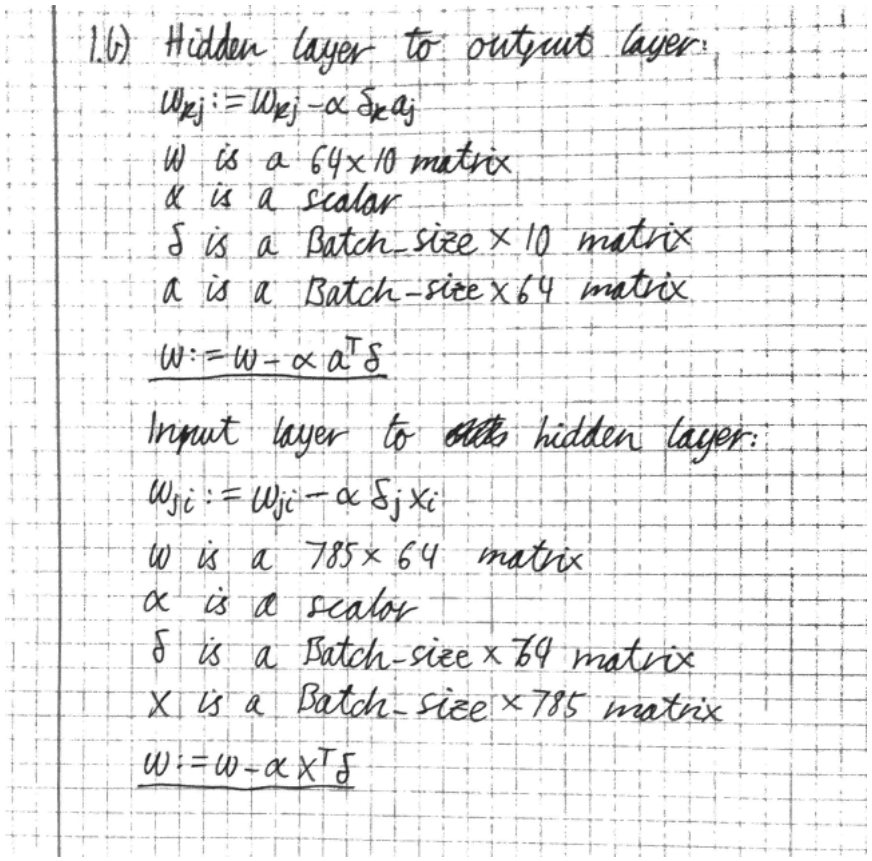
$$\delta_j = \frac{\partial C}{\partial z_j} = \sum_k \frac{\partial C}{\partial z_k} \frac{\partial z_k}{\partial a_j} \frac{\partial a_j}{\partial z_j}$$

$$= \sum_k \delta_k \frac{\partial}{\partial a_j} \left(\sum_j w_{kj} a_j \right) \frac{\partial f(z_j)}{\partial z_j}$$

$$= \sum_k \delta_k \frac{\partial}{\partial a_j} \left(w_{kj} a_j + \sum_{m \neq j} w_{km} a_m \right) f'(z_j)$$

$$= \underline{f'(z_j) \sum_k w_{kj} \delta_k}$$

2.2 task 1b)



1.b) Hidden layer to output layer:

$w_{kj} := w_{kj} - \alpha \delta_k a_j$

w is a 64×10 matrix

α is a scalar

δ is a Batch_size $\times 10$ matrix

a is a Batch-size $\times 64$ matrix

$w := w - \alpha a^T \delta$

Input layer to ~~old~~ hidden layer:

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

w is a 785×64 matrix

α is a scalor

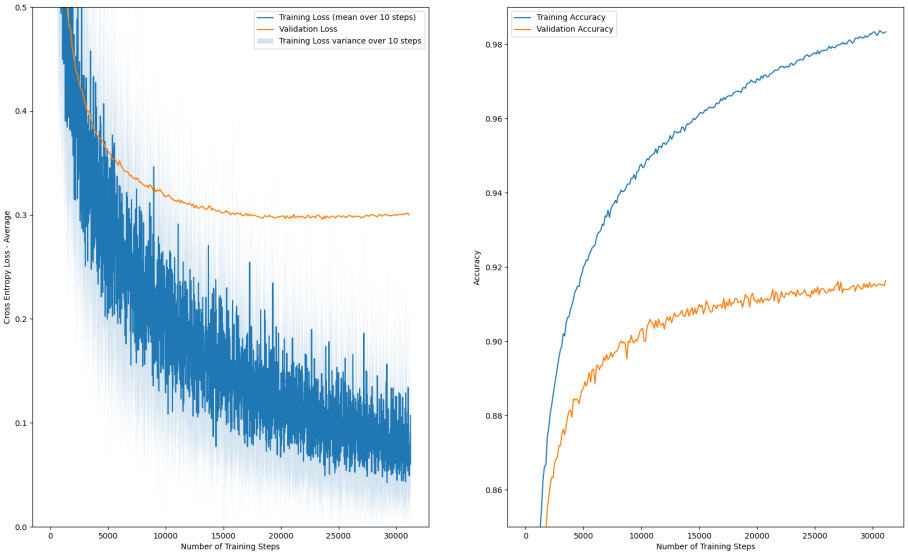
δ is a Batch-size $\times 64$ matrix

X is a Batch_size $\times 785$ matrix

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

3 Task 2

3.1 Task 2c)



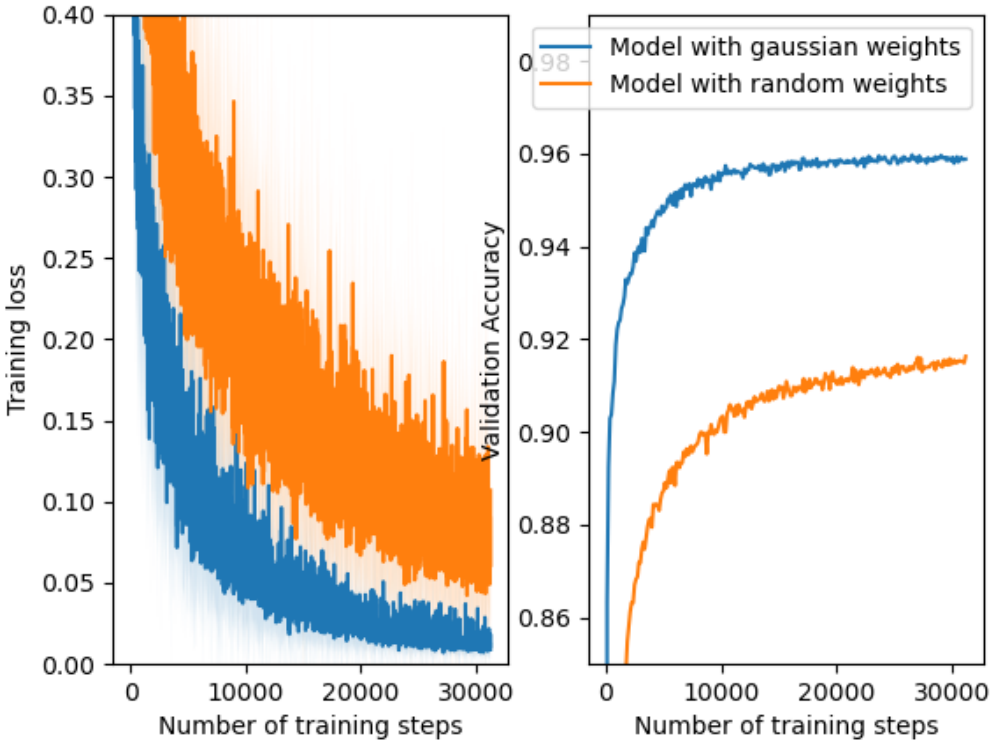
3.2 Task 2d)

There are $785 * 64 + 64x10 = 50880$ parameters in the network. The biases come from adding one extra input node.

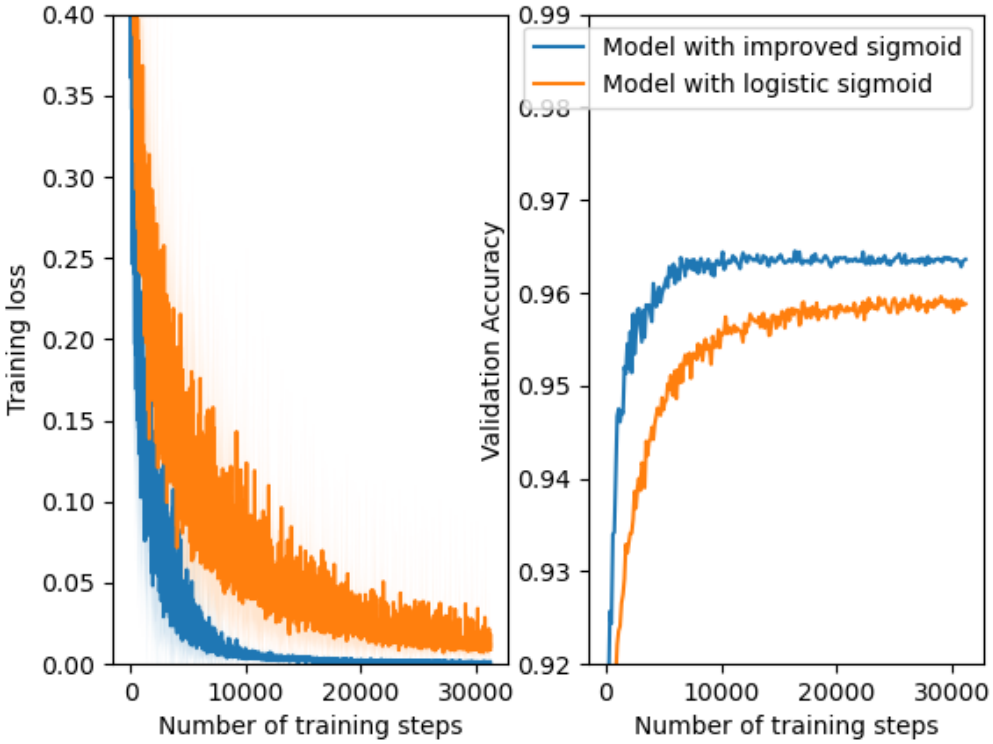
4 Task 3

All the changes were added incrementally and on top of eachother, e.g. the plot comparing sigmoid functions also has improved initial weights in both models.

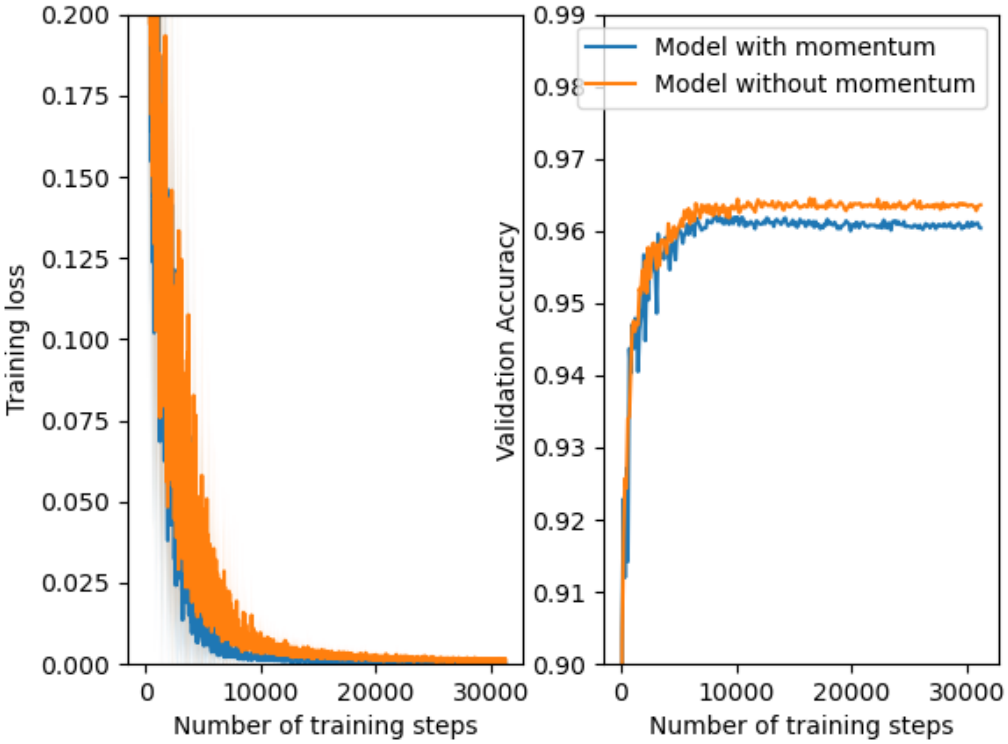
First i added improved initial weights which resulted in a much higher convergence speed, final accuracy and lower loss.



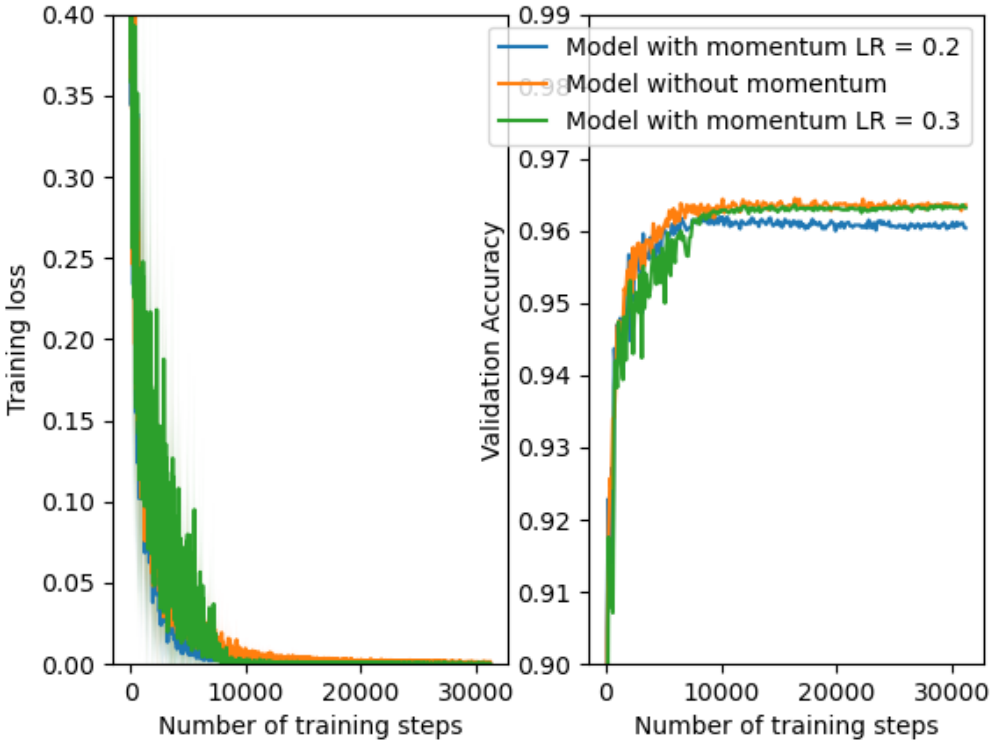
Next i added the improved sigmoid function which again gave a higher convergence speed, validation accuracy and lower training loss. The change was not as large as when adding the first trick but still a significant improvement.



Finally i added momentum, which resulted in a lower training loss, but it reduced the validation accuracy by a tiny bit. This trick might have a better effect if the learning rate was set higher or if it was implemented without the improved weights and the improved sigmoid.



I tried increasing the learning rate to 0.03 while using momentum and the validation accuracy improved almost to the point of the network without momentum, but still not as good.

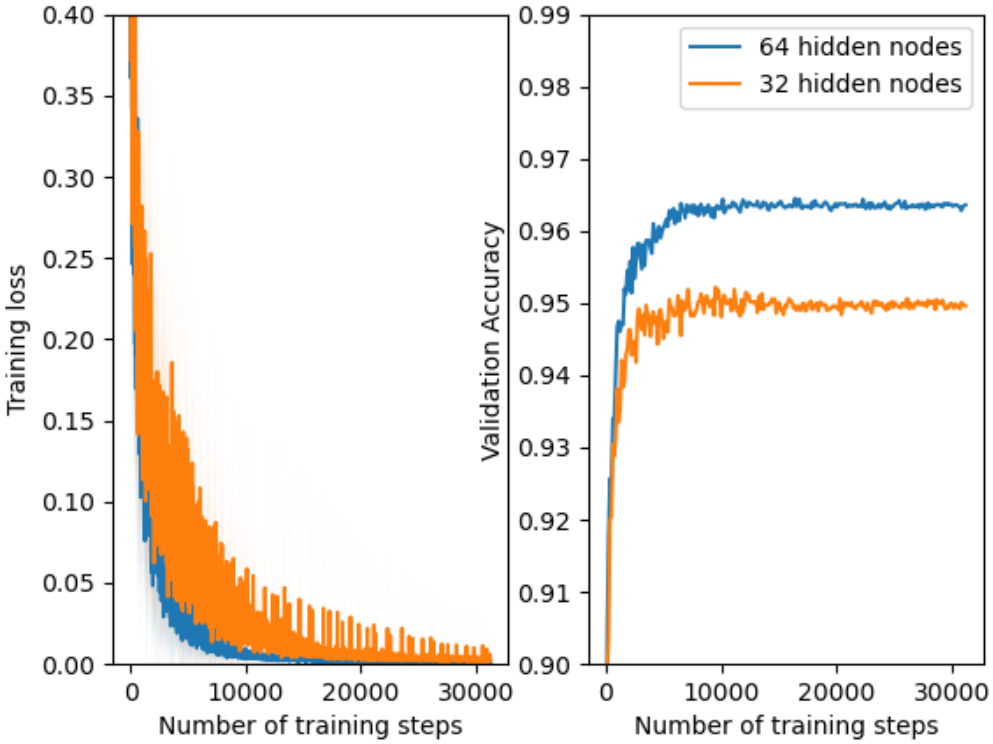


5 Task 4

5.1 Task 4a)

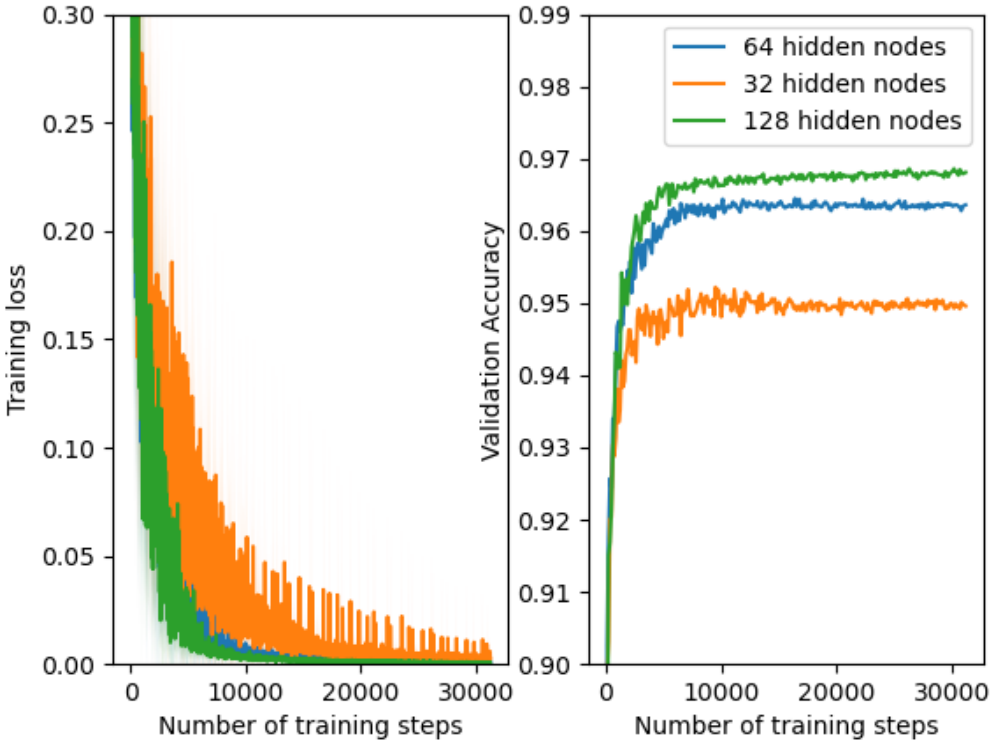
For task 4 i chose to use the best performing model from task 3, which was the model with improved sigmoid but without momentum. This also reduced runtime and memory constraints which occured when trying to train multiple models, especially for the model with 128 hidden nodes which creates a lot of 128x20000 and 128x10000 arrays.

With 32 hidden units we can see that the accuracy is lower than with 64, and the loss is also higher, which indicates some underfitting.



5.2 Task 4b)

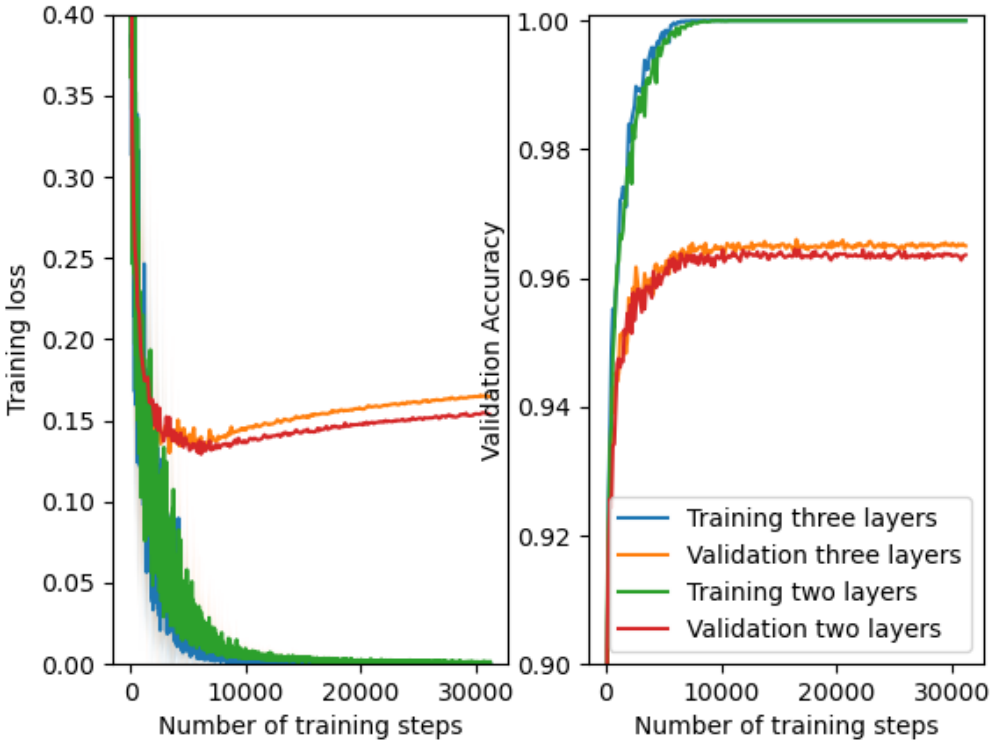
With 128 hidden units we get a small increase in accuracy, however the training time and memory needed was a lot larger than with 64 units.



5.3 Task 4d)

In task 3 the network had 50880 parameters (from 2d)). Now i want a network with two hidden layers of equal size which requires x nodes where x is determined by $x^2 + 785x + 10x = 50880$. This equation translates to $x \approx 59.5$ so i chose to make a network with two hidden layers of size 59, giving a total of $785 * 59 + 59^2 + 59 * 10 = 50386$ parameters.

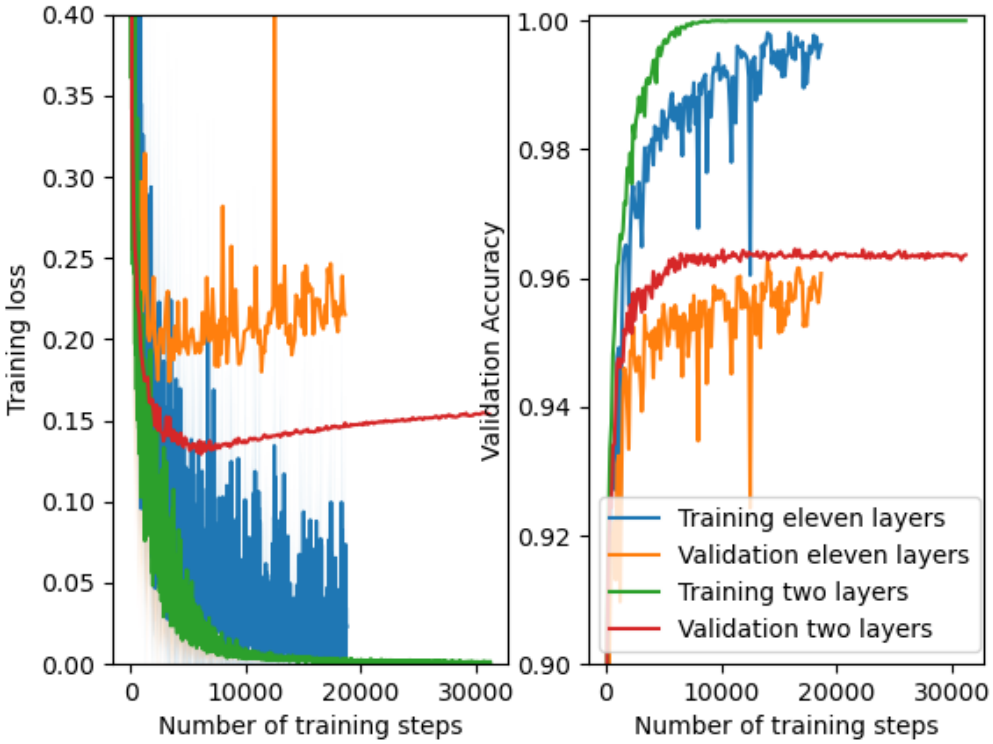
With two hidden layers we get a small increase in validation accuracy and a small increase in validation loss. The network had a higher validation accuracy with fewer parameters, but also a higher validation loss.



5.4 Task 4e)

I trained the model with ten hidden layers for 30 epochs. The plot compares the model with ten layers to the model with two layers and no momentum from Task 3.

We can see that the model with ten hidden layers has a higher loss and lower accuracy than the two layer model. There is also a lot more noise in the curves. This is caused by overfitting to the noise in the Mnist pictures instead of using only the features important to distinguishing numbers. If we trained the ten layer model with more data it might have ended up with a different result, but for classifying numbers we do not need a model that complex in order to perform very well. For this application ten hidden layers will result in a model which is too complex and includes too much variance.


Training loss
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
10000
20000
30000
Number of training steps
Validation Accuracy
1.00
0.98
0.96
0.94
0.92
0.90
0
10000
20000
30000
Number of training steps
Training eleven layers
Validation eleven layers
Training two layers
Validation two layers