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` .

Task 1

Task 1a)

Show that

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

can be written as

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

where $\delta_j = f'(z_j) \sum_k w_{kj} \delta_k$.

We apply the chain rule to $\alpha \frac{\partial C}{\partial w_{ji}}$ and get the following expression:

$$\alpha \frac{\partial C}{\partial w_{ij}} = \alpha \sum_k \frac{\partial C}{\partial z_k} \frac{\partial z_k}{\partial a_j} \frac{\partial a_j}{\partial z_j} \frac{\partial z_j}{\partial w_{ji}}$$

We then take a look at the different factors in the above expression and solve them independently.

$$\frac{\partial C}{\partial z_k} = \delta_k$$

From previous assignment.

$$\frac{\partial z_k}{\partial a_j} = \frac{\partial}{\partial a_j} \left(\sum_{j'} w_{kj'} a_{j'} \right) = w_{kj}$$

$$\frac{\partial a_j}{\partial z_j} = \frac{\partial f(z_j)}{\partial z_j} = f'(z_j)$$

$$\frac{\partial z_j}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}} \left(\sum_{i'=0}^{d} w j i' x_{i'} \right) = x_i$$

By putting all the above calculations together we get the following expression for $\alpha \frac{\partial C}{\partial w_{ji}}$:

$$\alpha \frac{\partial C}{\partial w_{ji}} = f'(z_j) \sum_k w_{kj} \delta_k$$

task 1b)

In order to write the update rule in matrix notataion we will go through the network, from the input layer to the output layer, and note the dimensions of the different matrices and vectors. The network is defined with I as the number of nodes in the input layer, J as the number of nodes in the hidden layer and K as the number of nodes in the output layer.

Define size of input vector and weight matrix between input layer and hidden layer:

$X : [I \times 1]$

$W_{ji} : [J \times I]$

Feed the input vector through the input layer:

$z_j = W_{ji} \cdot X \quad [J \times 1]$

$a_j = f(z_j) \quad [J \times 1]$

a_j will be the input for the hidden layer. And the weight matrix from the hidden layer to the output layer will be:

$W_{kj} : [K \times J]$

Feed a_j through the hidden layer:

$z_k = W_{kj} \cdot a_j \quad [K \times 1]$

$a_k = f(z_k) \quad [K \times 1]$

The update rules for the weights are

$W_{ji} := W_{ji} - \alpha \delta_1^T x_i \quad [J \times I]$

$W_{kj} := W_{kj} - \alpha \delta_2^T a_j \quad [K \times J]$

Since the dimensions of the last part of the update rule needs to be the same as for the weight matrices to be updated, the dimensions of δ_1 and δ_2 needs to be:

$\delta_1 : [J \times 1]$

$\delta_2 : [K \times 1]$

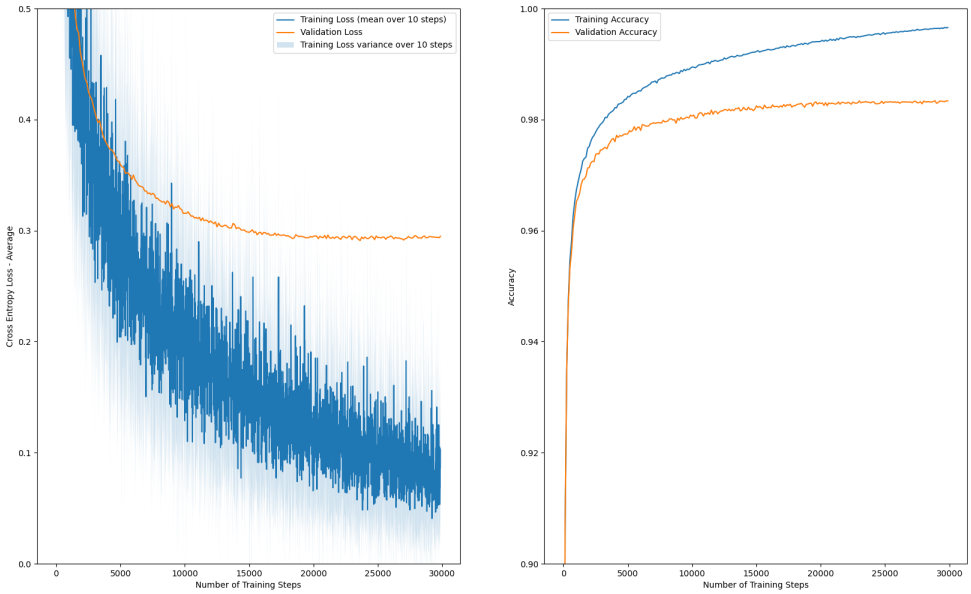
Task 2

Task 2a)

The mean and std are calculated as 33.55274553571429 and 78.87550070784701, respectivly. The are saved as variables in the pre_process_images function so that they are the same for training, validation and test.

Task 2c)

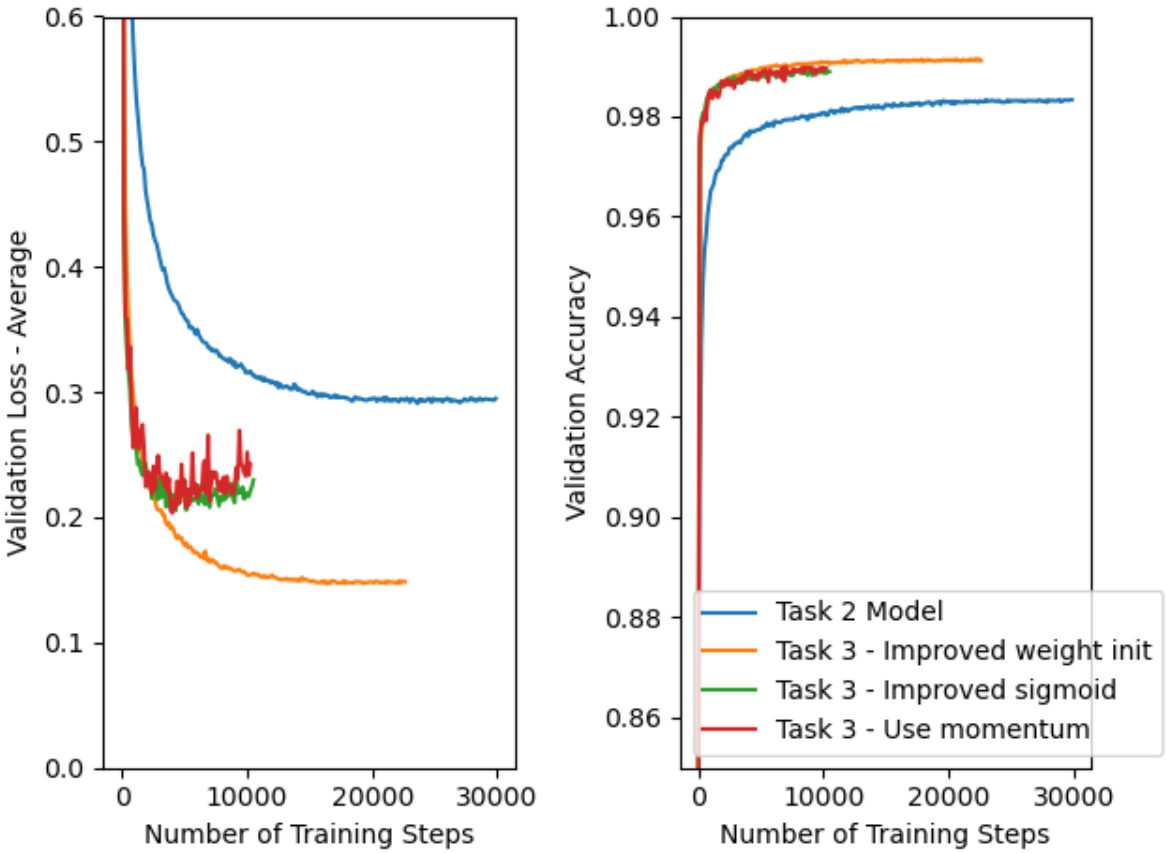
Here you have a plot of the training and validation loss and a plot shwoing the training and validation accuracy. It looks like the model overfits a bit ti the training data given the difference in the performance on the training and validation data.



Task 2d)

The network consists of a input layer with 785 neurons, including the bias, a hidden layer of 64 neurons and an output layer of 10 neurons. Thus the number of weights are $785 * 64 + 64 * 10 = 50880$.

Task 3



Above you can see a plot showing the validation loss and accuracy given the different "tricks of the trade".

Convergence speed:

As you can see from the plots the convergence speed of all three imoproved models are much faster than the model from task 2. Especially for the models with imporved sigmoid and momentum there is a great increase. Also, since these models converges much faster, the early stopping kicks in earlier, saves both time and computation.

Generalization/overfitting:

As earlier stated in task 2 the first models looks to overfit quite a lot given the much better performance on the training data compared to the validation data. For all the improved models this does not happen. All the improved models performs very good on the validation data, implying no overfitting.

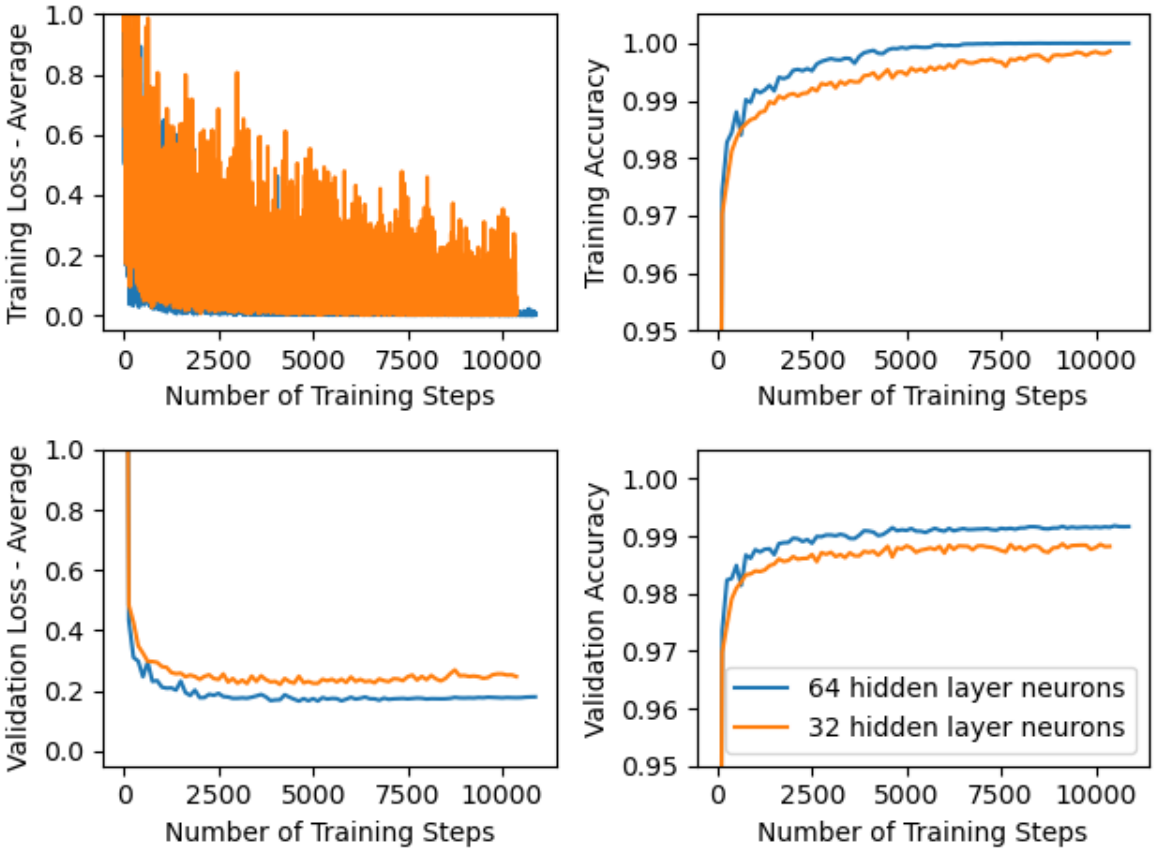
Final accuracy/Validation loss:

The final accuracy and validation loss is much better in all three imporved models compared to the original one. All three models has a loss that is almost half the one in the original model and 0.01 better validation accuracy.

Task 4

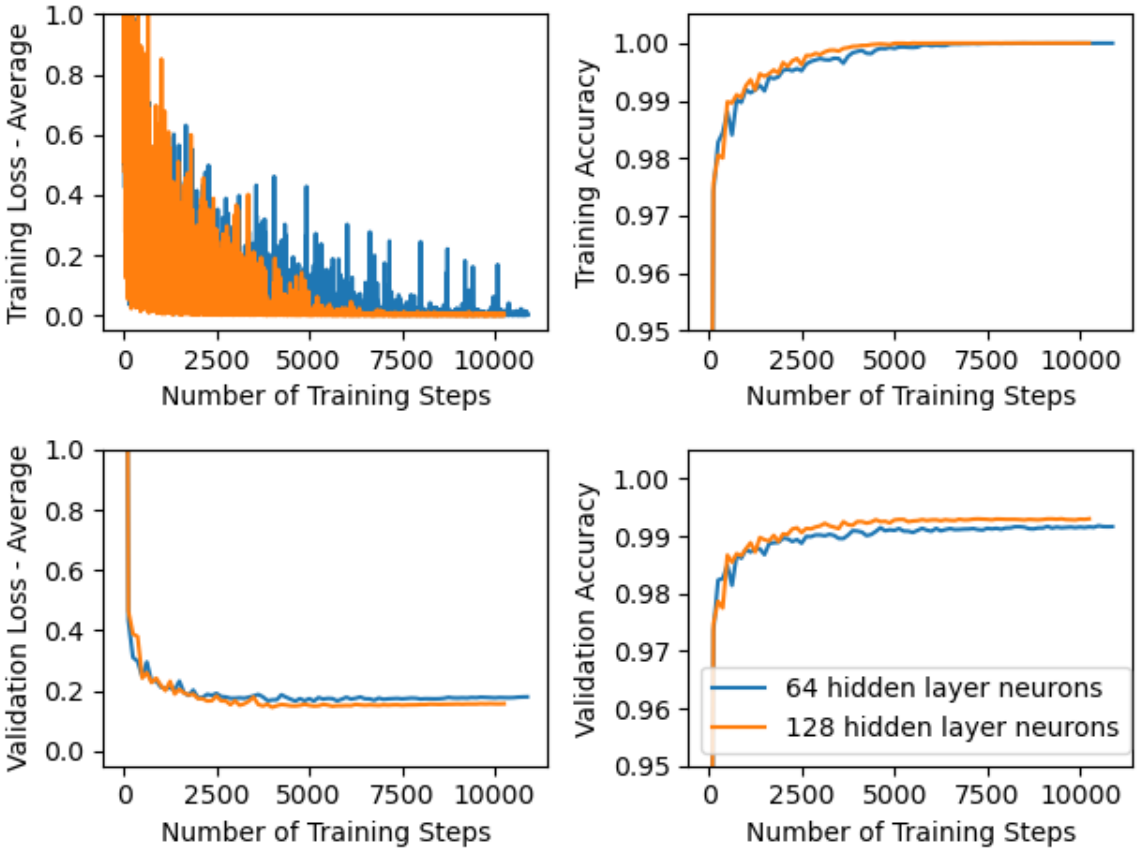
Task 4a)

Lowering the number of neurons in the hidden layer from 64 to 32 makes the network perform worse. The validation loss increases by 0.05 and the validation accuracy decreases by approximatley 0.01. The reason behind this worsen performance is probably that the model with 32 neurons in the hidden layer does not catch the features of the numbers as good as the model with 64 neurons, and thus do not manage to classify as good.



Task 4b)

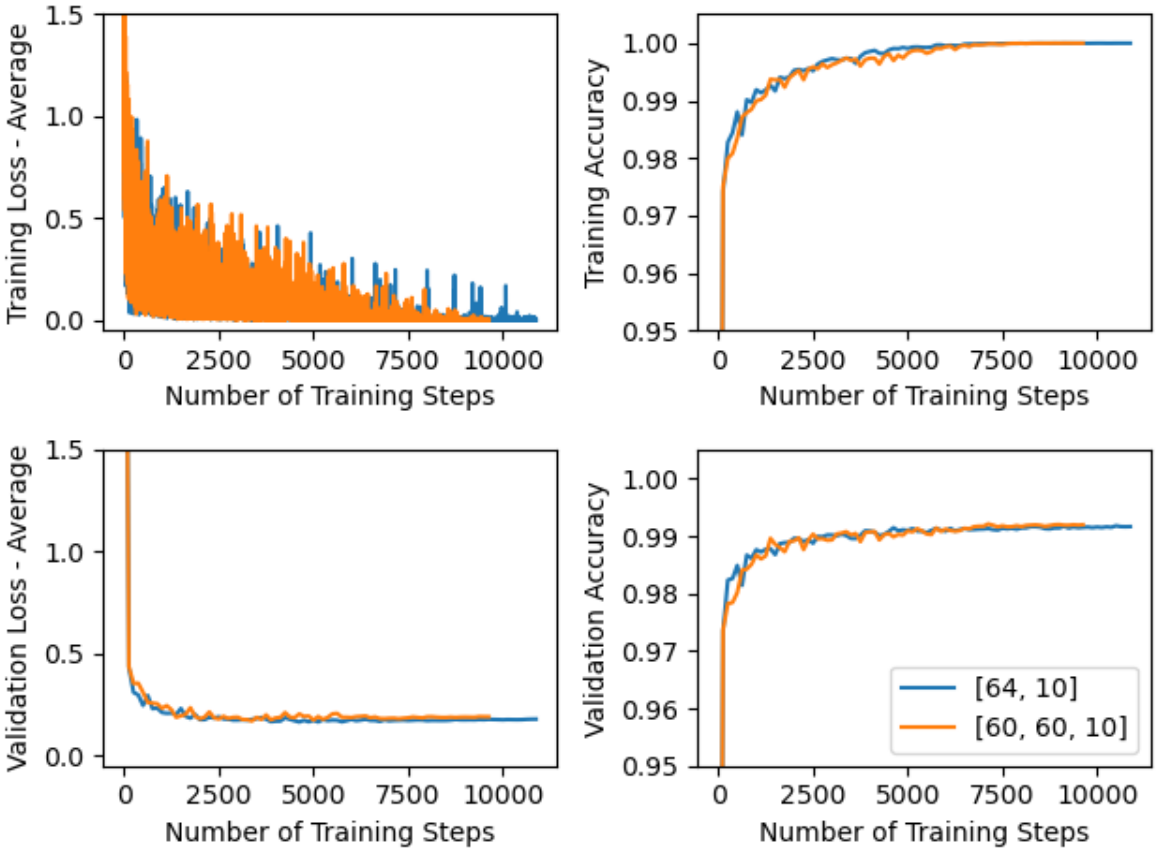
Increasing the number of neurons in the hidden layer from 64 to 128 makes the model perform better. This is probably beacuse the model with the higher number of neurons manges to catch more of the features seperating the different digits, and thus manages to classify better.



Task 4d)

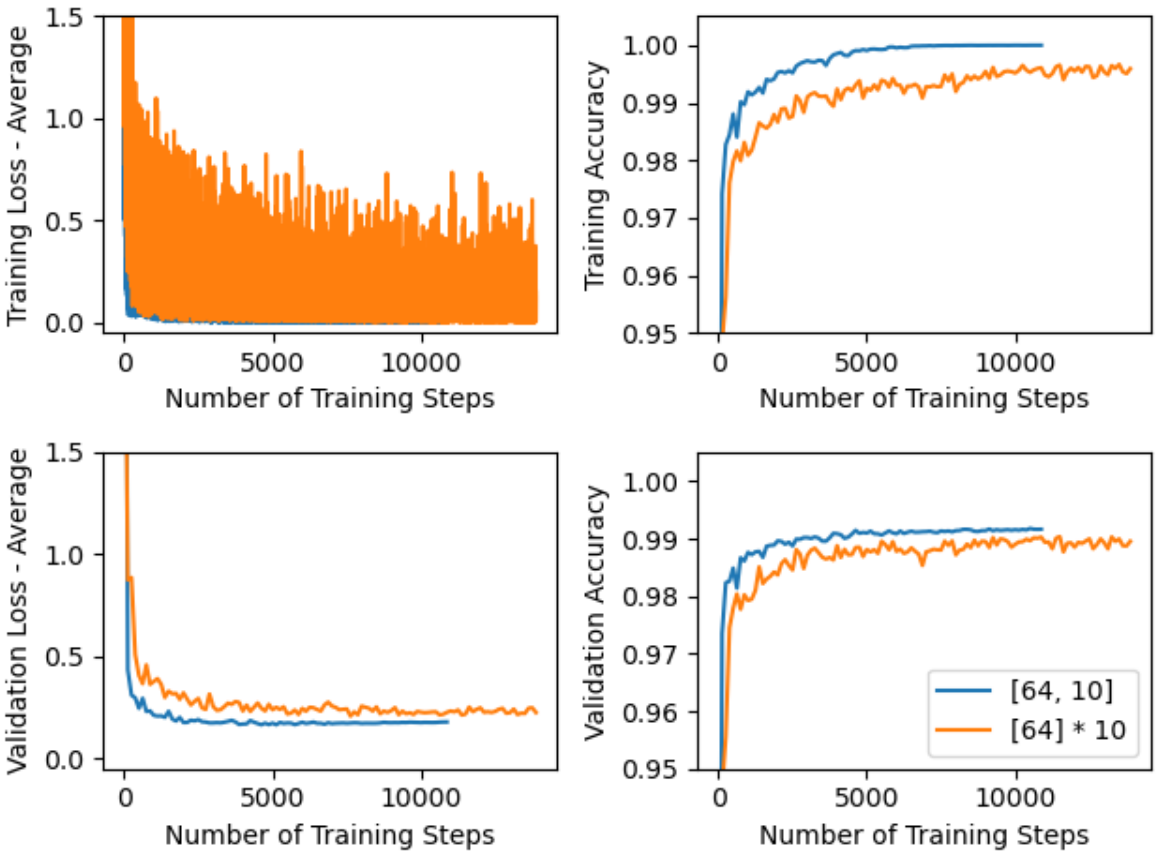
The number of parameters in the initial network was 50880. To get approximatley as many neurons we use [60, 60, 10] neurons in each layer giving
$785 * 60 + 60 * 60 + 60 * 10 = 50770$ parameters.

Below you can see the loss of training and validation.



The network with more layers, but approximatley as many parameters, performs alomst exactly as good as the original from earlier.

Task 4e)



The new model with more layers performs worse than the original model. It could be that the

training stops to early and that the model could improve even more given more iterations. Also the loss on the training data looks very noisy and that it is struggeling to find a minima.