Assignment 2 Report

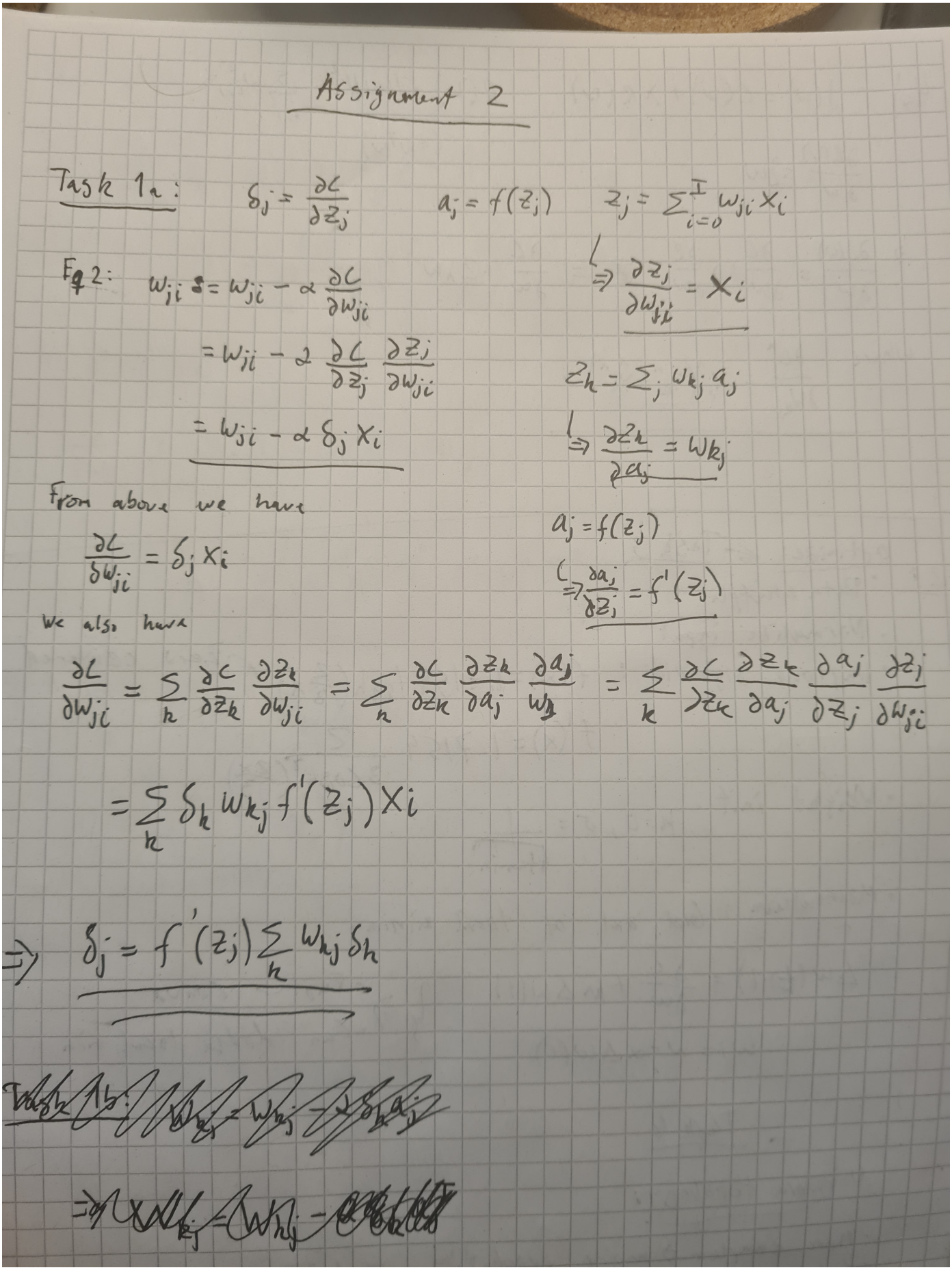
This is an outline for your report to ease the amount of work required to create your report. Jupyter notebook supports markdown, and I recommend you to check out this cheat sheet (https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet). If you are not familiar with markdown.

Before delivery, **remember to convert this file to PDF**. You can do it in two ways:

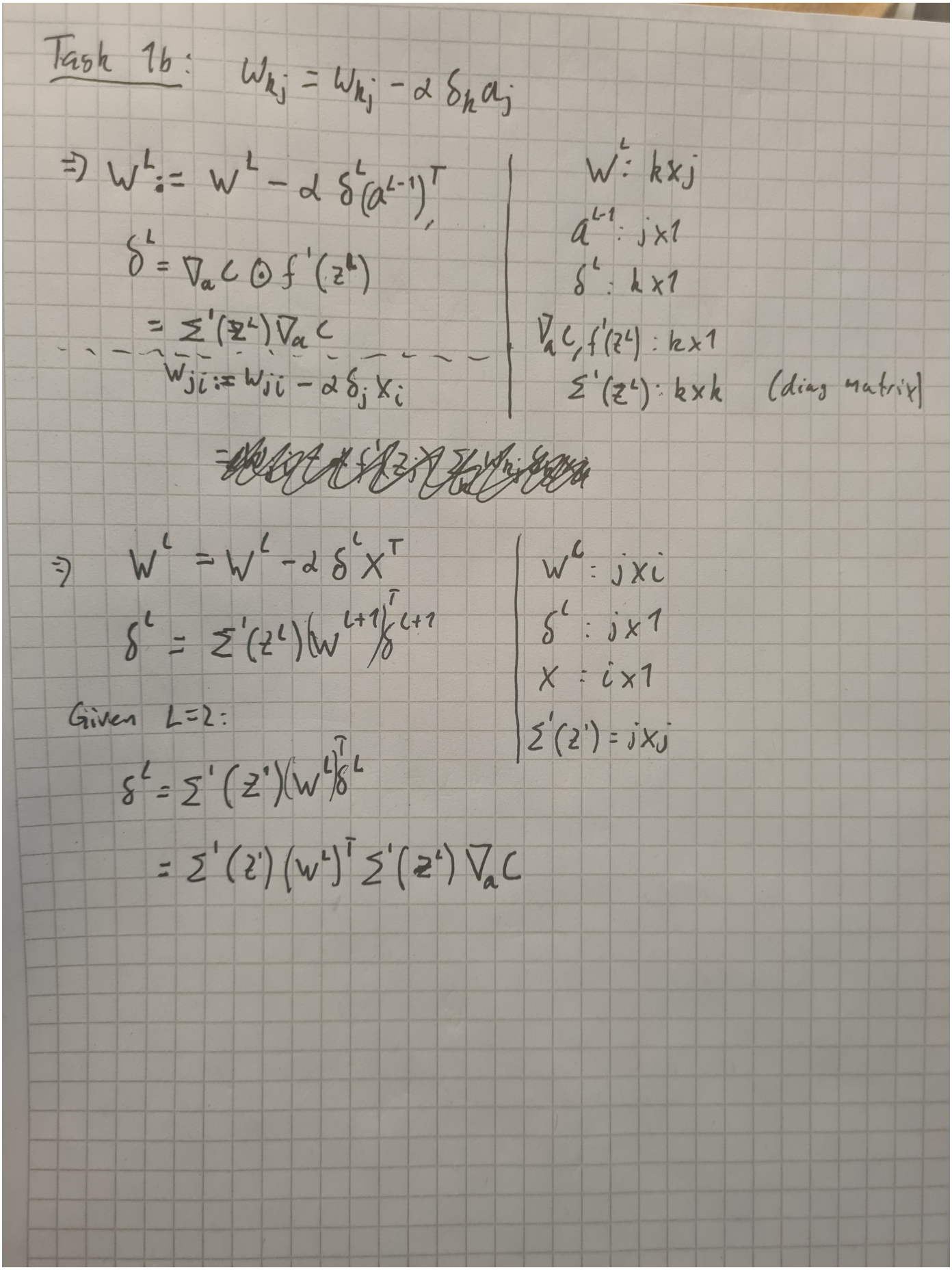
1. Print the webpage (ctrl+P or cmd+P)
2. Export with latex. This is somewhat more difficult, but you'll get somehwat of a "prettier" PDF. Go to File -> Download as -> PDF via LaTeX. You might have to install nbconvert and pandoc through conda; `conda install nbconvert pandoc` .

Task 1

task 1a)

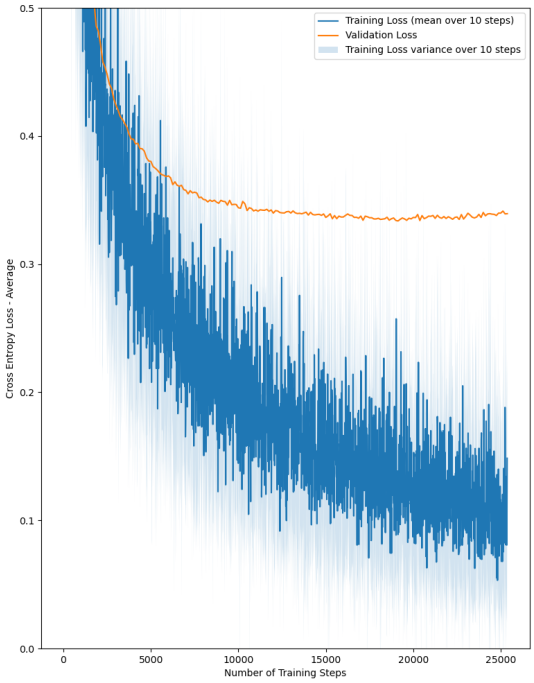

Assignment 2
Task 1a:
$\delta_j = \frac{\partial C}{\partial z_j}$
$a_j = f(z_j)$
$z_j = \sum_{i=0}^{I} w_{ji} x_i$
$\Rightarrow \frac{\partial z_j}{\partial w_{ji}} = x_i$
Eq 2: $w_{ji} = w_{ji} - \alpha \frac{\partial C}{\partial w_{ji}}$
$= w_{ji} - \alpha \frac{\partial C}{\partial z_j} \frac{\partial z_j}{\partial w_{ji}}$
$= w_{ji} - \alpha \delta_j x_i$
$z_k = \sum_j w_{kj} a_j$
$\Rightarrow \frac{\partial z_k}{\partial a_j} = w_{kj}$
From above we have
$\frac{\partial C}{\partial w_{ji}} = \delta_j x_i$
$a_j = f(z_j)$
$\Rightarrow \frac{\partial a_j}{\partial z_j} = f'(z_j)$
We also have
$\frac{\partial C}{\partial w_{ji}} = \sum_k \frac{\partial C}{\partial z_k} \frac{\partial z_k}{\partial w_{ji}} = \sum_k \frac{\partial C}{\partial z_k} \frac{\partial z_k}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}} = \sum_k \frac{\partial C}{\partial z_k} \frac{\partial z_k}{\partial a_j} \frac{\partial a_j}{\partial z_j} \frac{\partial z_j}{\partial w_{ji}}$
$= \sum_k \delta_k w_{kj} f'(z_j) x_i$
$\Rightarrow \delta_j = f'(z_j) \sum_k w_{kj} \delta_k$

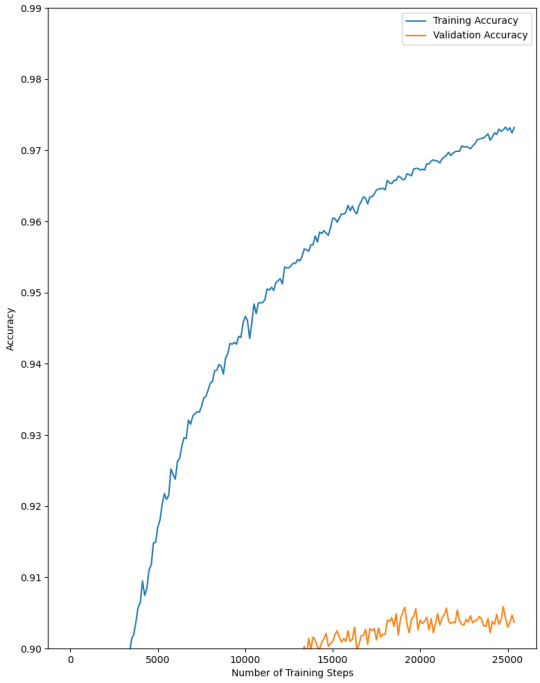
task 1b)


Task 1b: $w_{kj} = w_{kj} - \alpha \delta_k a_j$
$\Rightarrow w^L := w^L - \alpha \delta^L (a^{L-1})^T$
$\delta^L = \nabla_a C \odot f'(z^L)$
$= \Sigma'(z^L) \nabla_a C$
$w_{ji} := w_{ji} - \alpha \delta_j x_i$
W^L: kxj
a^{L-1}: jx1
δ^L: kx1
$\nabla_a C, f'(z^L)$: kx1
$\Sigma'(z^L)$: kxk (diag matrix)
$\Rightarrow W^L = W^L - \alpha \delta^L x^T$
$\delta^L = \Sigma'(z^L)(w^{L+1})^T \delta^{L+1}$
w^L: jxi
δ^L: jx1
x: ix1
$\Sigma'(z^1)$: jxj
Given L=2:
$\delta^L = \Sigma'(z^1)(w^L)^T \delta^L$
$= \Sigma'(z^1)(w^L)^T \Sigma'(z^L) \nabla_a C$

Task 2

Task 2c)



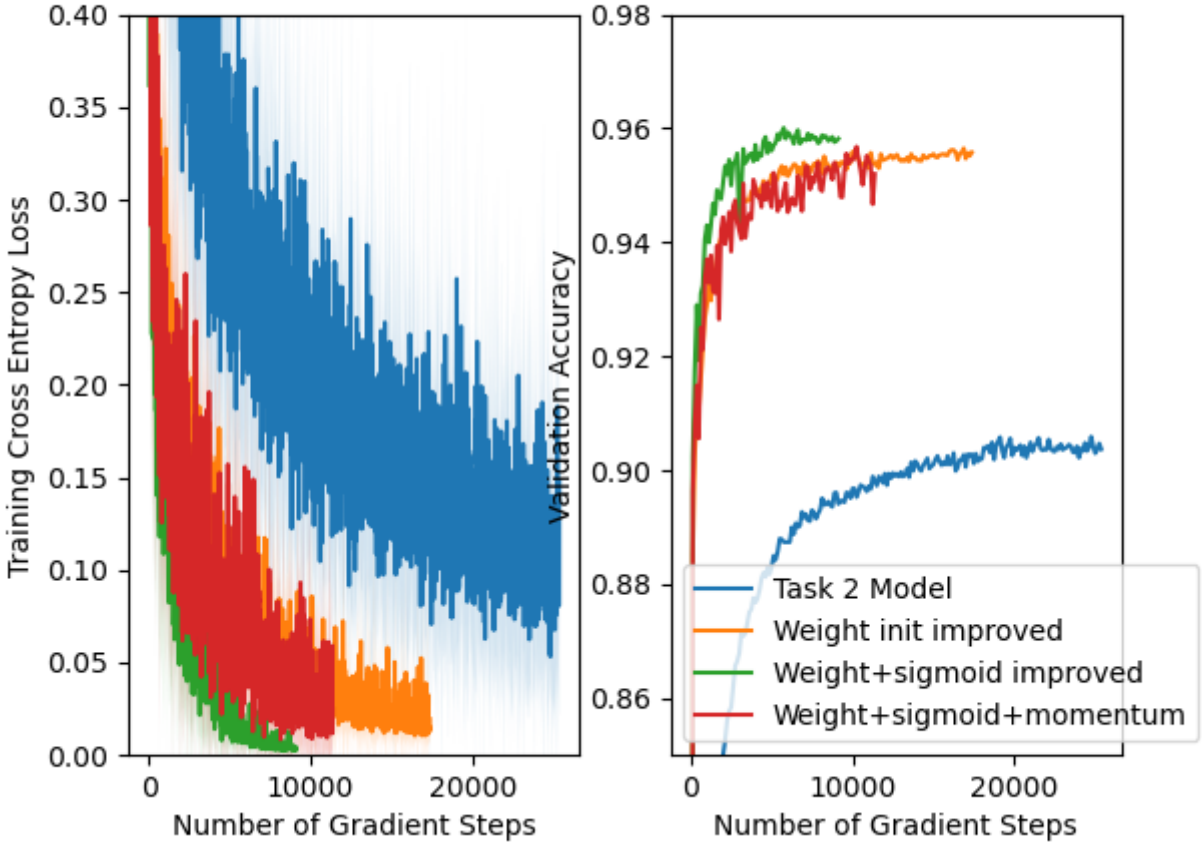


Task 2d)

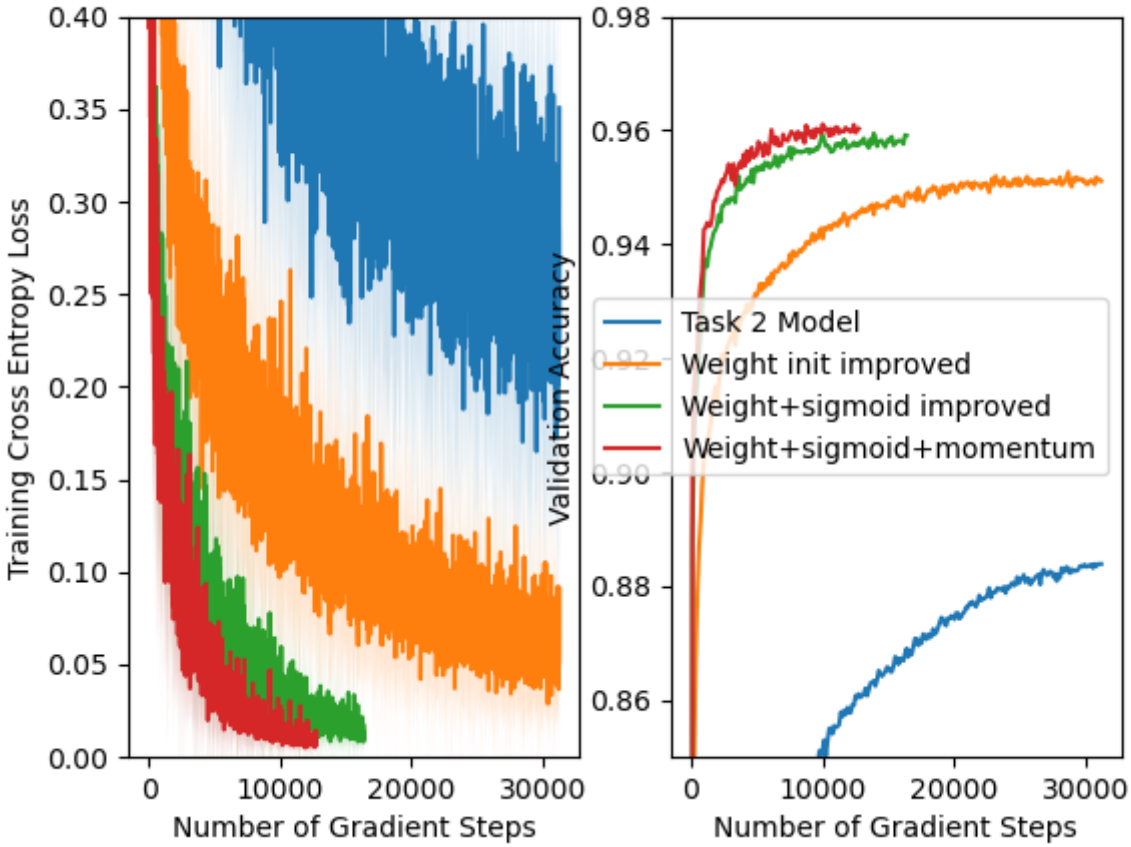
Givent that we use the bias trick, the bias is part of the weigths. Num parameters = 785*64+64*10 = 50 880

Task 3

Training accuracy=0.1



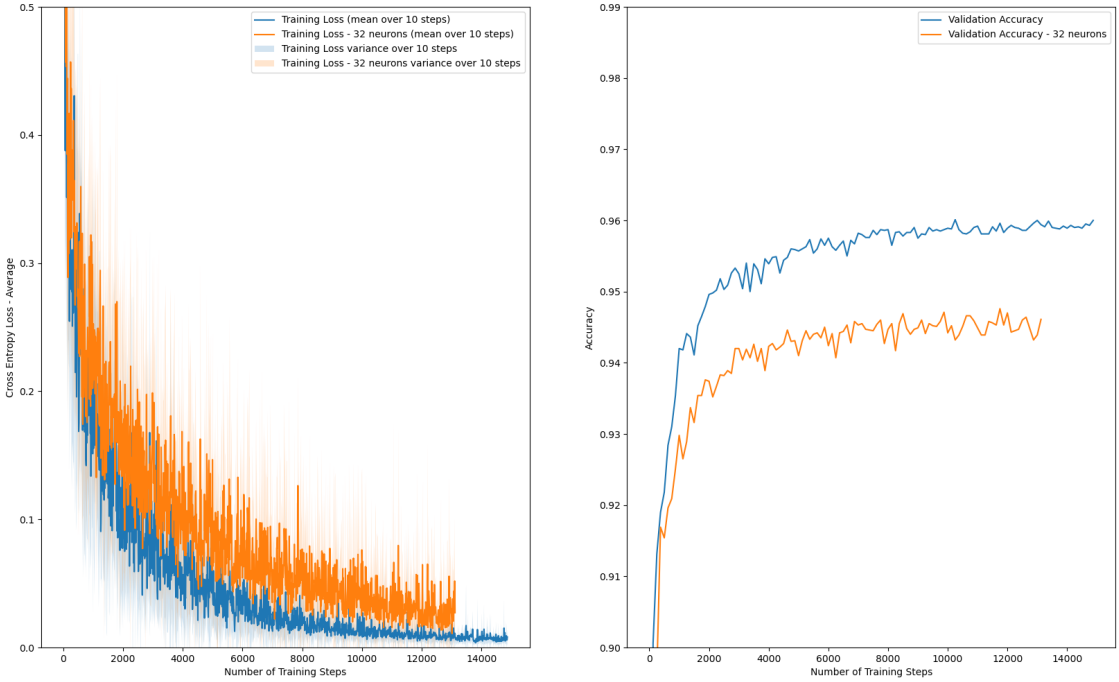
Training accuracy=0.02



From the plots we see that the addition of momentum yields a poorer performance with a higher training accuracy. This is probably due to the gradient becoming too big and pushing the result to a bigger minima. For the lower training accuracy we can see that it converges faster and yields the highest accuracy. We can also see that introducing improved weight initialization significantly increases convergence and accuracy.

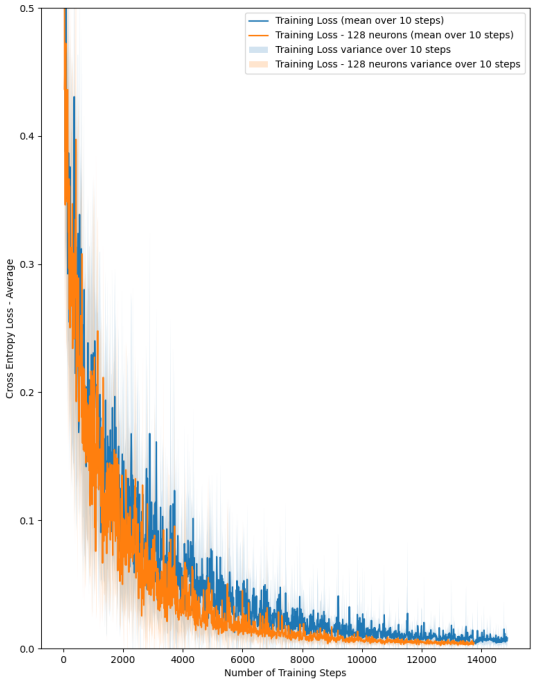
Task 4

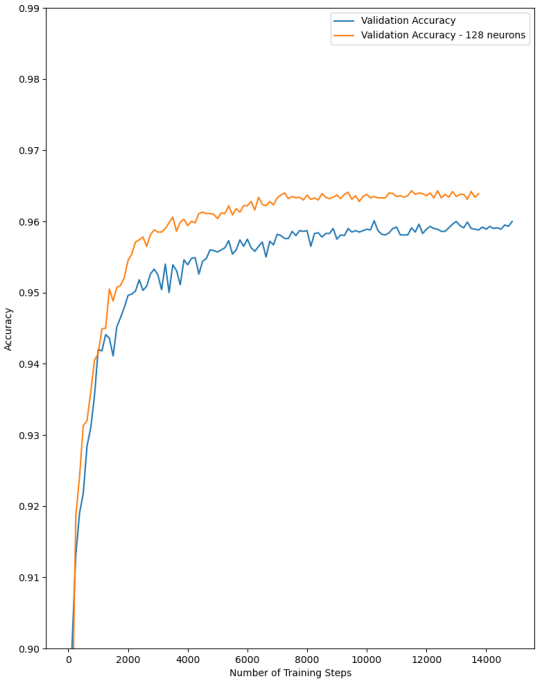
Task 4a)



As we see above the accuracy is reduced when reducing the number of neurons per layer. We however reduve the number of traing steps.

Task 4b)

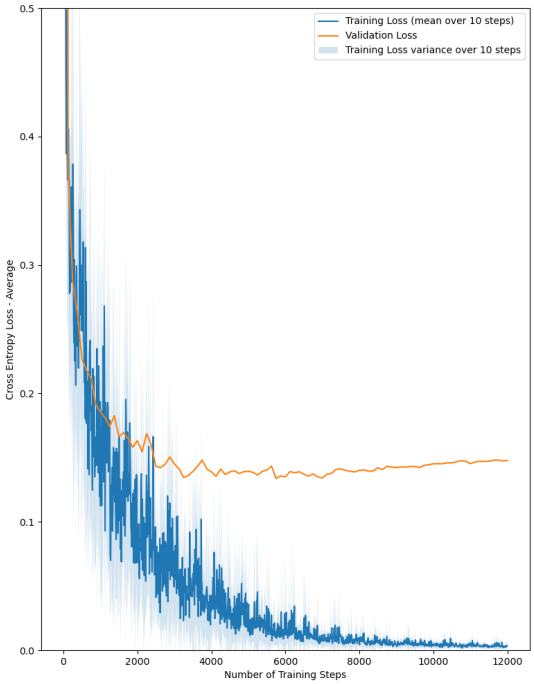


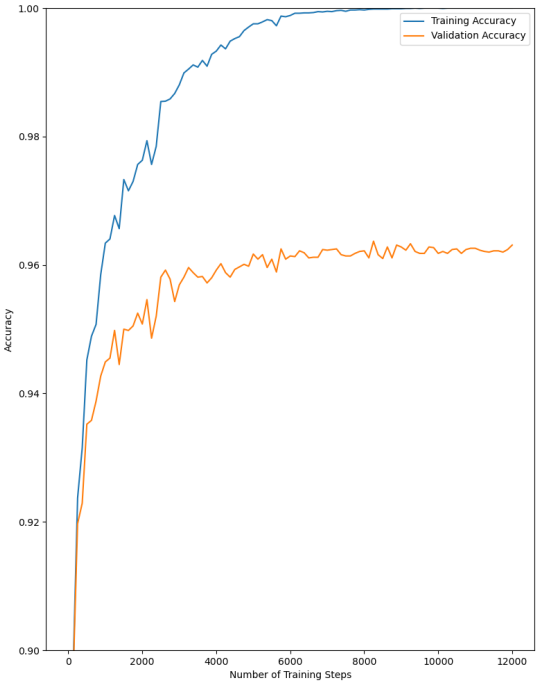


As we see above, the accuracy is increased and the traing steps are reduced when increasing the number of neurons per layer.

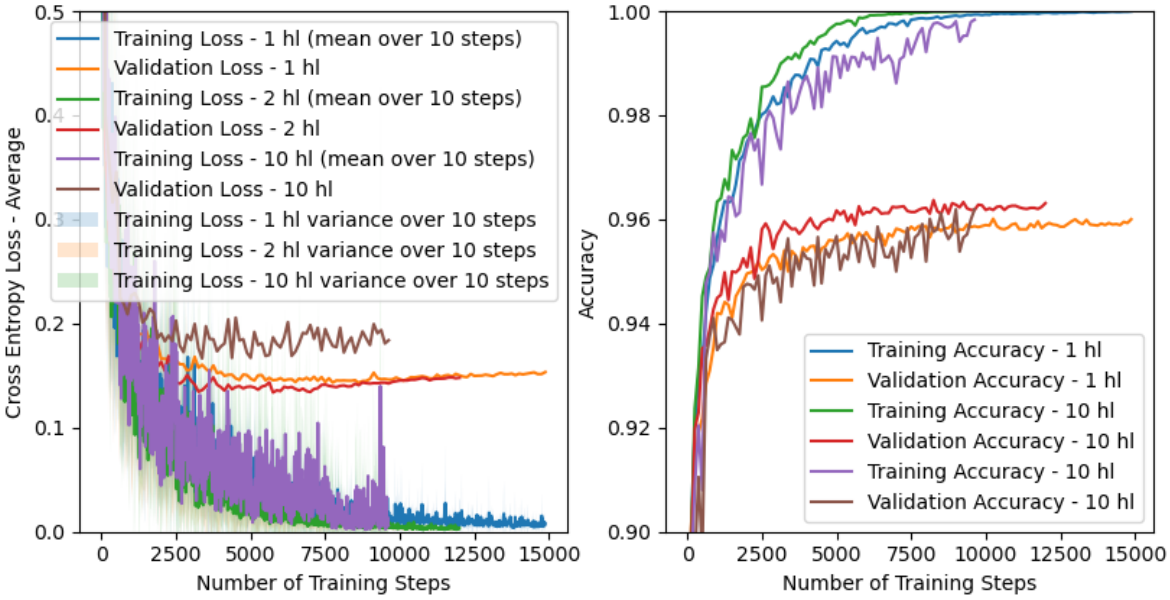
Task 4d)

In task 3 we have the same amount of parameters as we derived in 2d: 50 880 In the new network we have 1 additional layer, such that Num parameters = 785*64+64*64+64*10 = 54976 We now have 128 hidden units in our network.





Task 4e)



We see that while introducing 2 hidden layers increases the accuracy, using 10 hidden layers decreases it. We see that it aslo gets "noisy". These factors are due to overfitting, where the compexity of this model does not fit the problem.