Assignment 1 Report

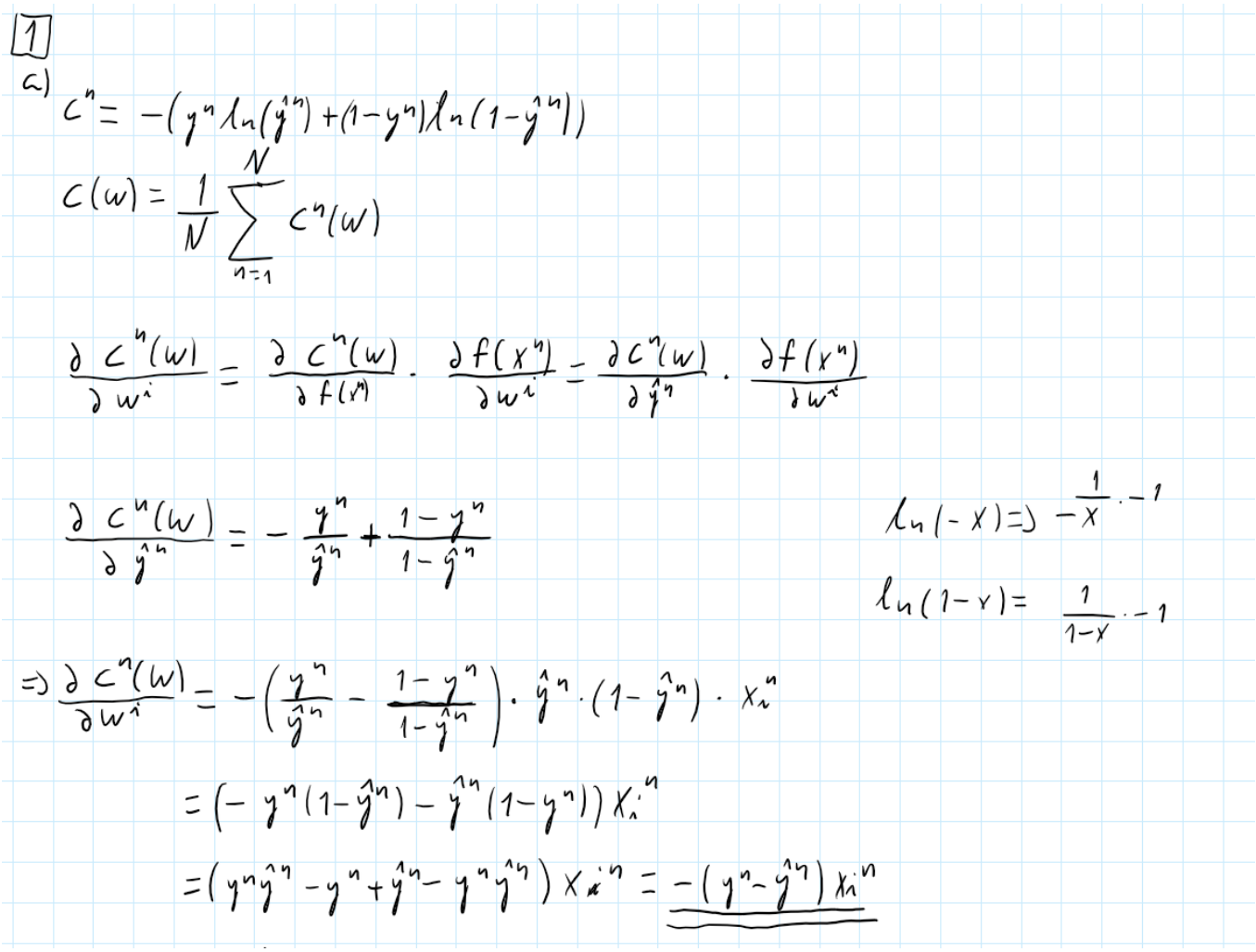
This is an outline for your report to ease the amount of work required to create your report. Jupyter notebook supports markdown, and I recommend you to check out this cheat sheet (https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet). If you are not familiar with markdown.

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

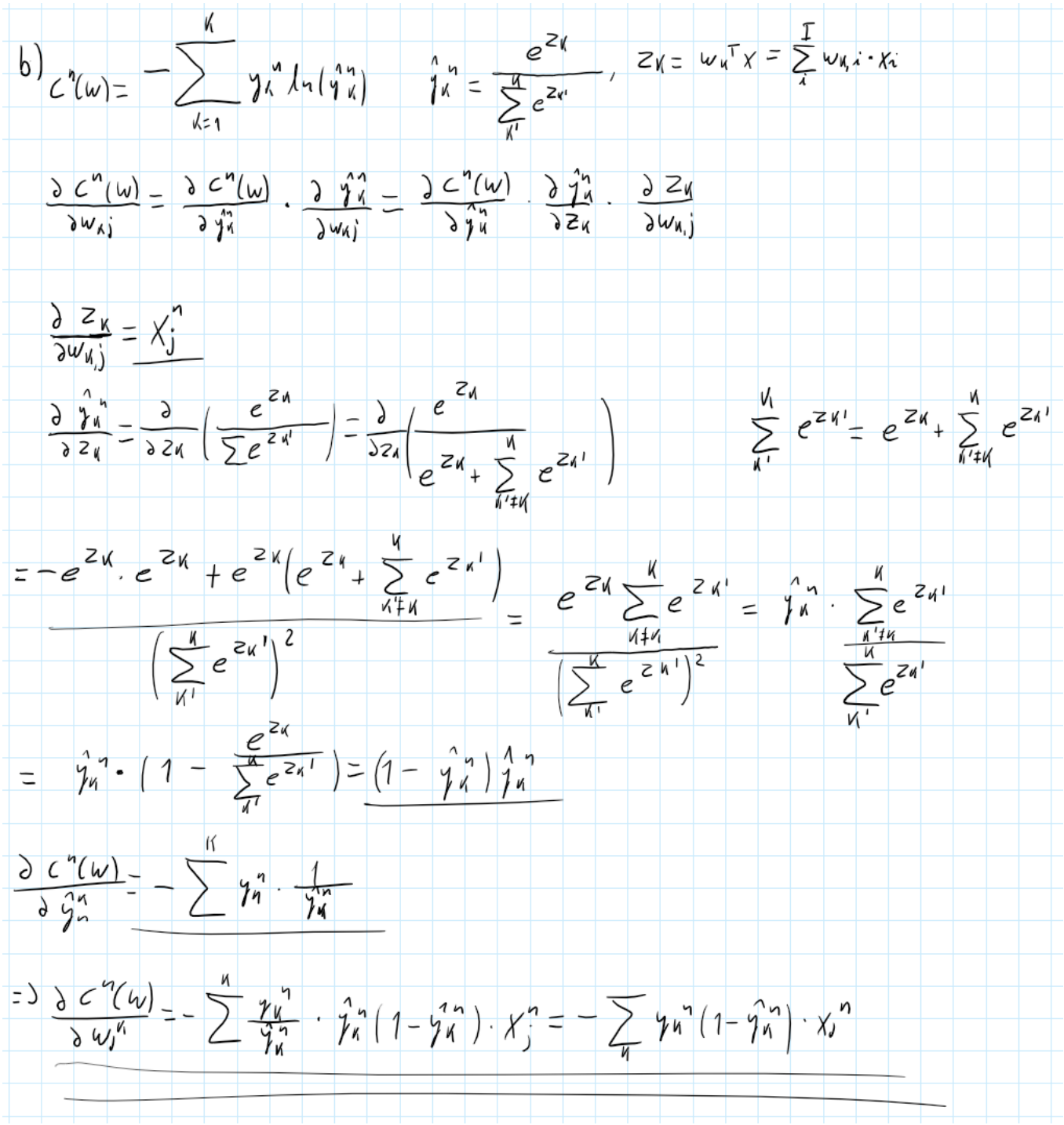
1. Print the webpage (ctrl+P or cmd+P)
2. Export with latex. This is somewhat more difficult, but you'll get somehwat of a "prettier" PDF. Go to File -> Download as -> PDF via LaTeX. You might have to install nbconvert and pandoc through conda; `conda install nbconvert pandoc` .

Task 1

task 1a)



task 1b)



This was the closest I could get to the answer, I got the same answer when calculating dC/dz^k

$$\frac{\partial C^n(w)}{\partial z_k} = \frac{\partial}{\partial z_k}\left(-\sum_{k=1}^{K} y_k^n \cdot \ln\left(\frac{e^{z_k}}{\sum_{k'} e^{z_{k'}}}\right)\right)$$

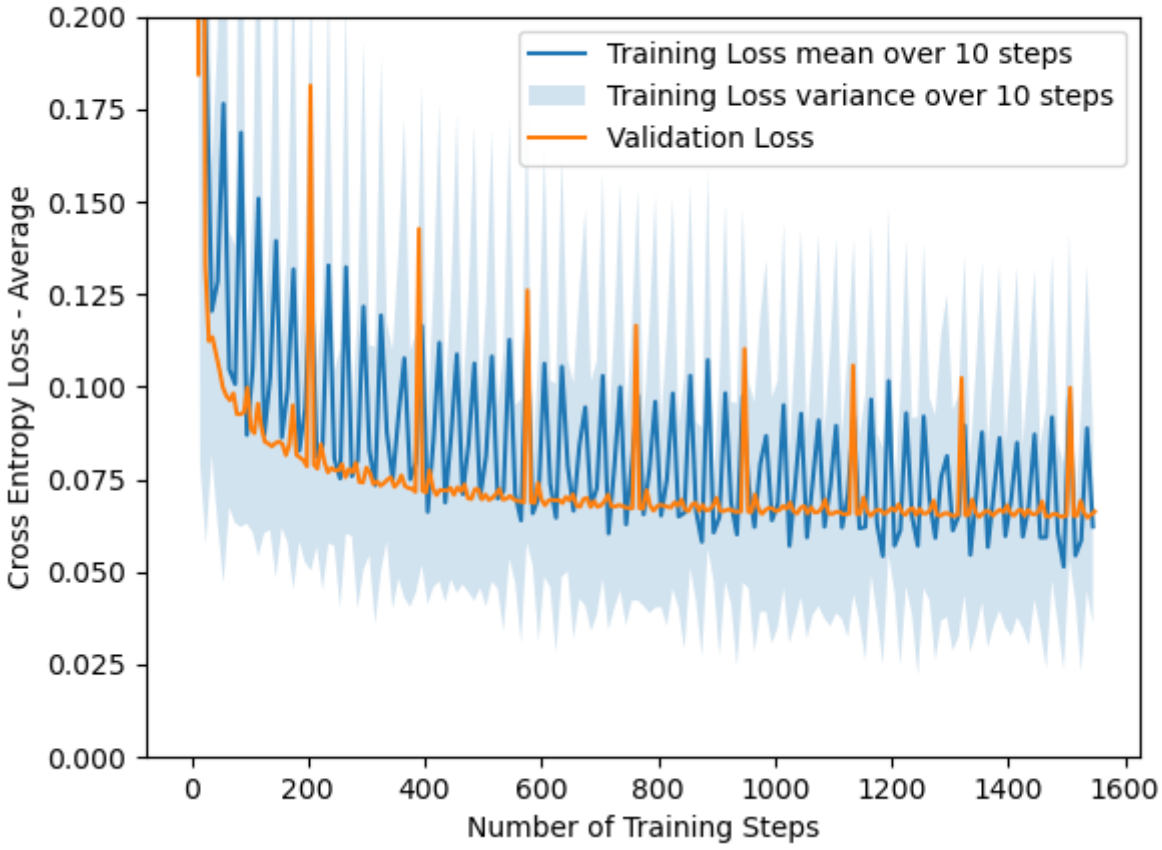
$$= -\frac{\partial}{\partial z_k}\left(\sum y_k^n \cdot \left(\ln(e^{z_k}) - \ln\left(\sum e^{z_{k'}}\right)\right)\right),$$

$$= -\frac{\partial}{\partial z_k}\left(\sum y_k^n \left(z_k - \ln\left(\sum e^{z_{k'}}\right)\right)\right) = -\sum y_k^n \left(1 - \frac{e^{z_k}}{\sum e^{z_{k'}}}\right) = \sum \left(y_k^n \frac{e^{z_k}}{\sum z^{k'}}\right) - 1$$

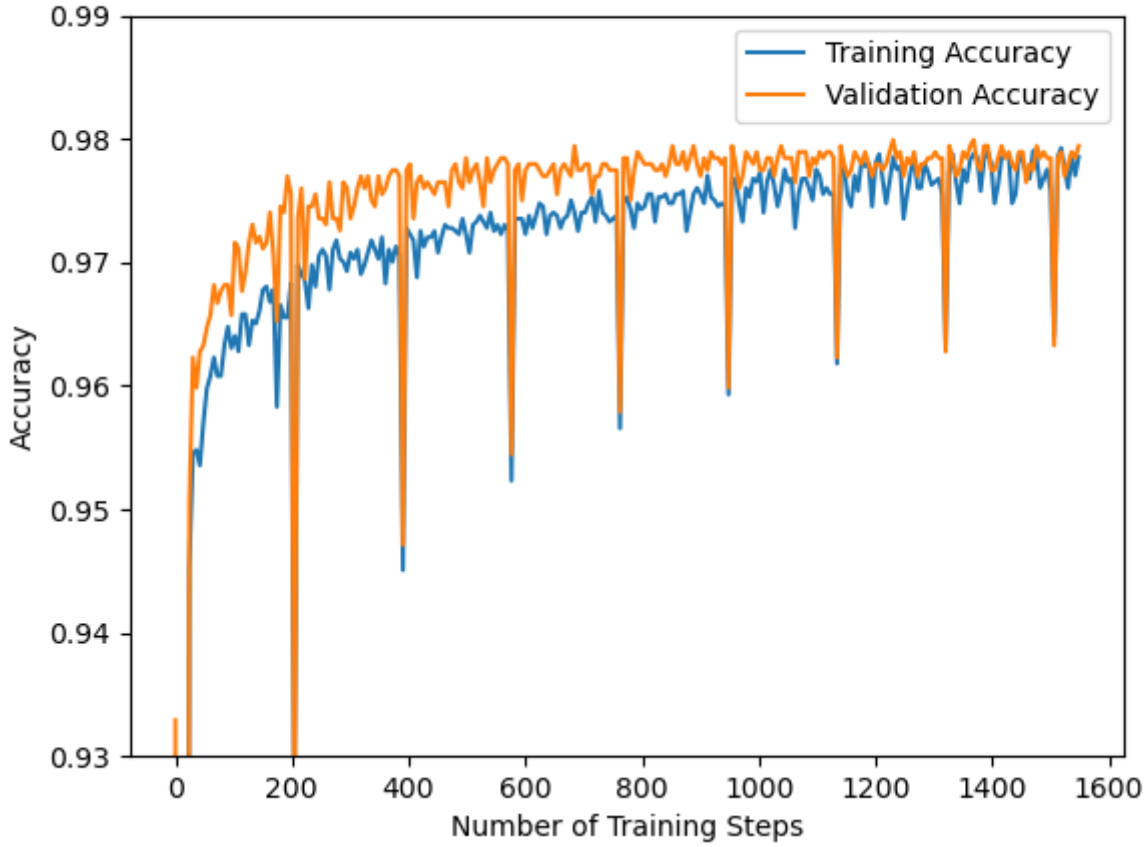
$$= \sum y_k^n \hat{y}_k^n - y_k^n$$

Task 2

Task 2b)



Task 2c)

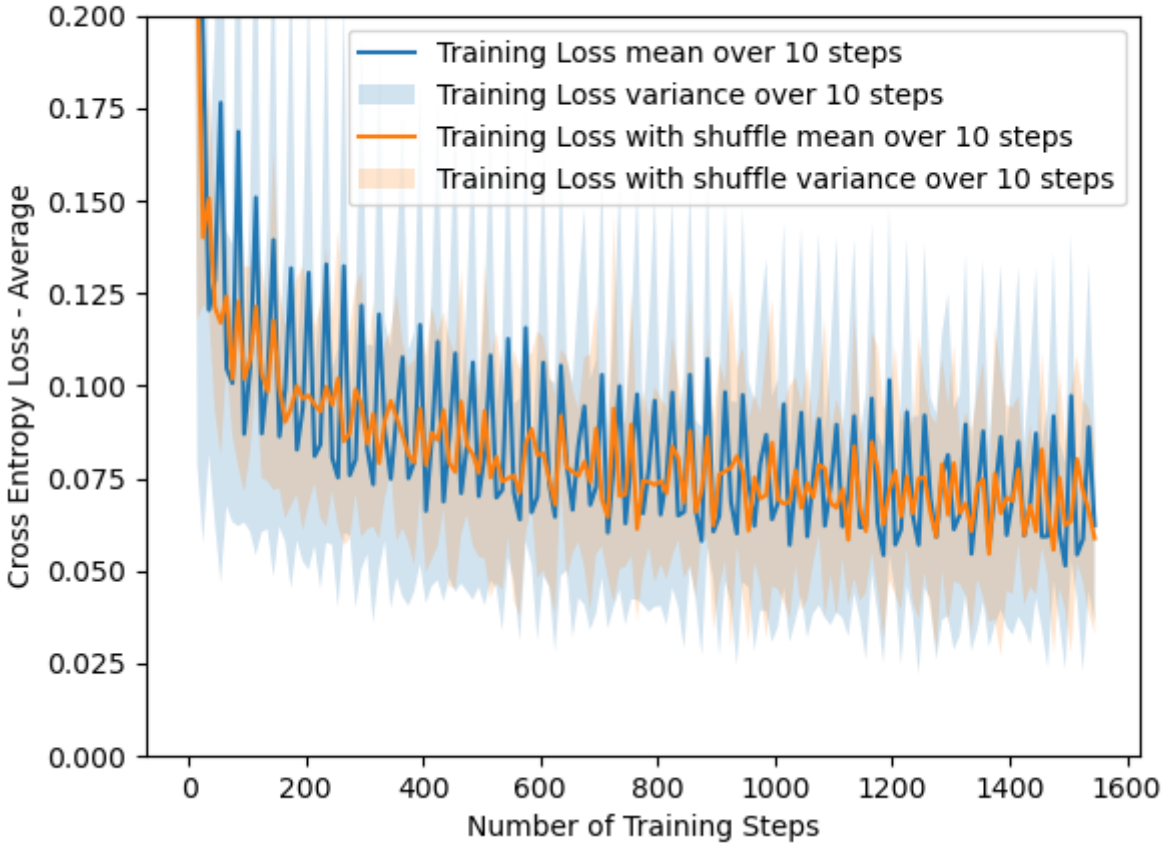


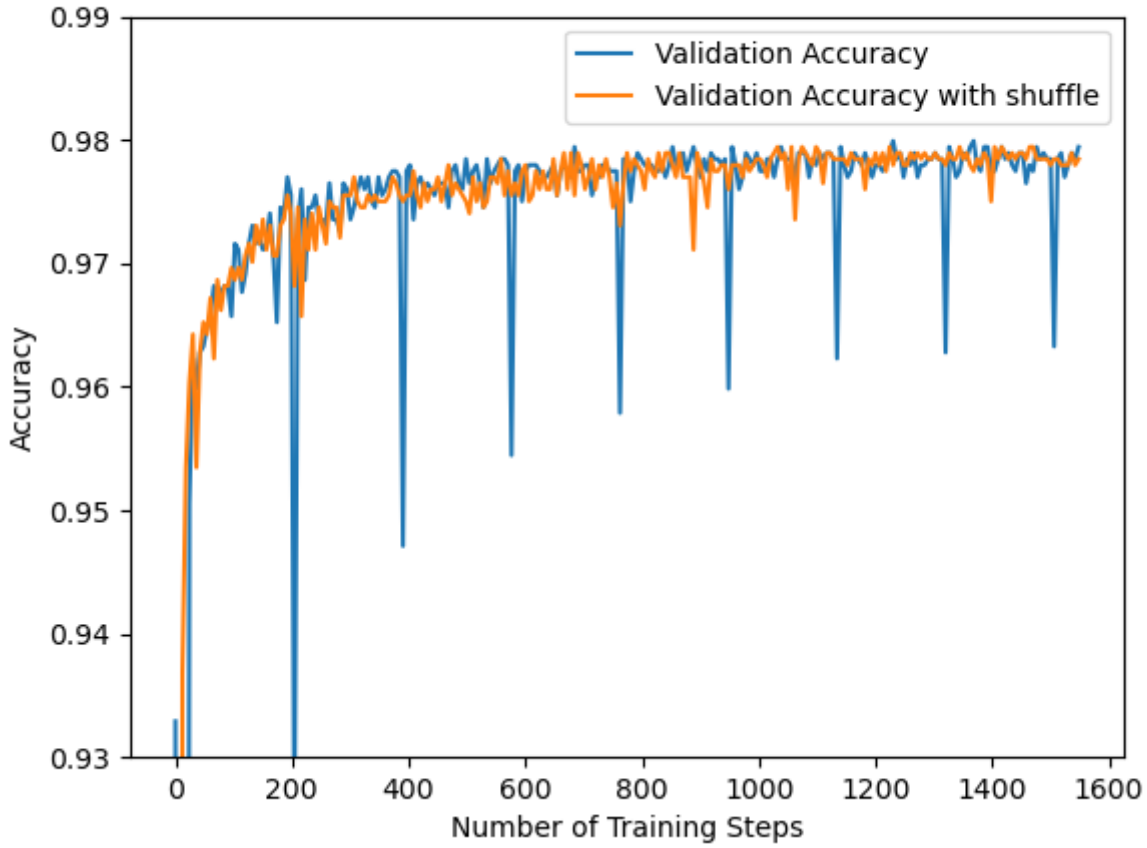
Task 2d)

It stops after 33 epochs, with shuffling after 16.

Task 2e)

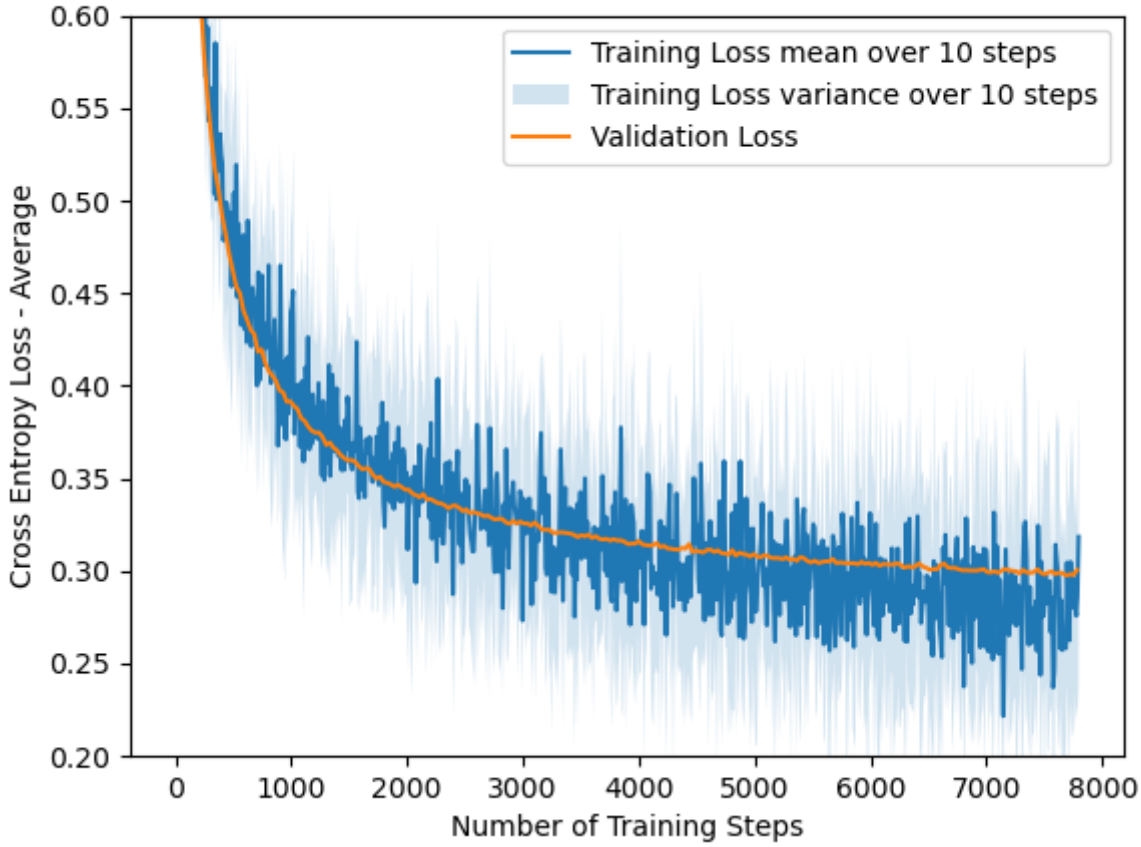
My best guess would be that every 200-ish training steps there is a validation set that is particularly difficult for the neural net to classify, the shuffling of the data set will therefore spread out the difficult to classify data points. Generally the random shuffling will remove any deterministic bias of the data set.


Training Loss mean over 10 steps
Training Loss variance over 10 steps
Training Loss with shuffle mean over 10 steps
Training Loss with shuffle variance over 10 steps
Cross Entropy Loss - Average
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
0
200
400
600
800
1000
1200
1400
1600
Number of Training Steps

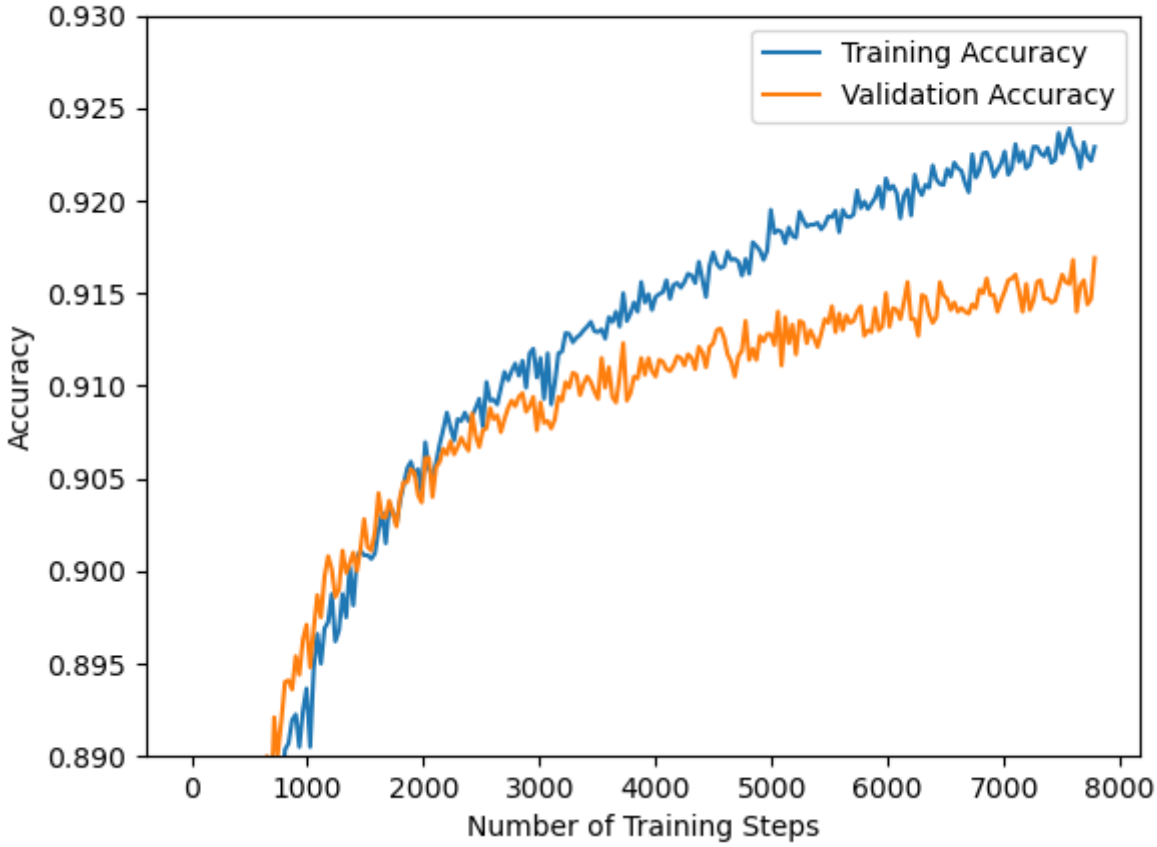

Validation Accuracy
Validation Accuracy with shuffle
Accuracy
0.99
0.98
0.97
0.96
0.95
0.94
0.93
0
200
400
600
800
1000
1200
1400
1600
Number of Training Steps

Task 3

Task 3b)



Task 3c)

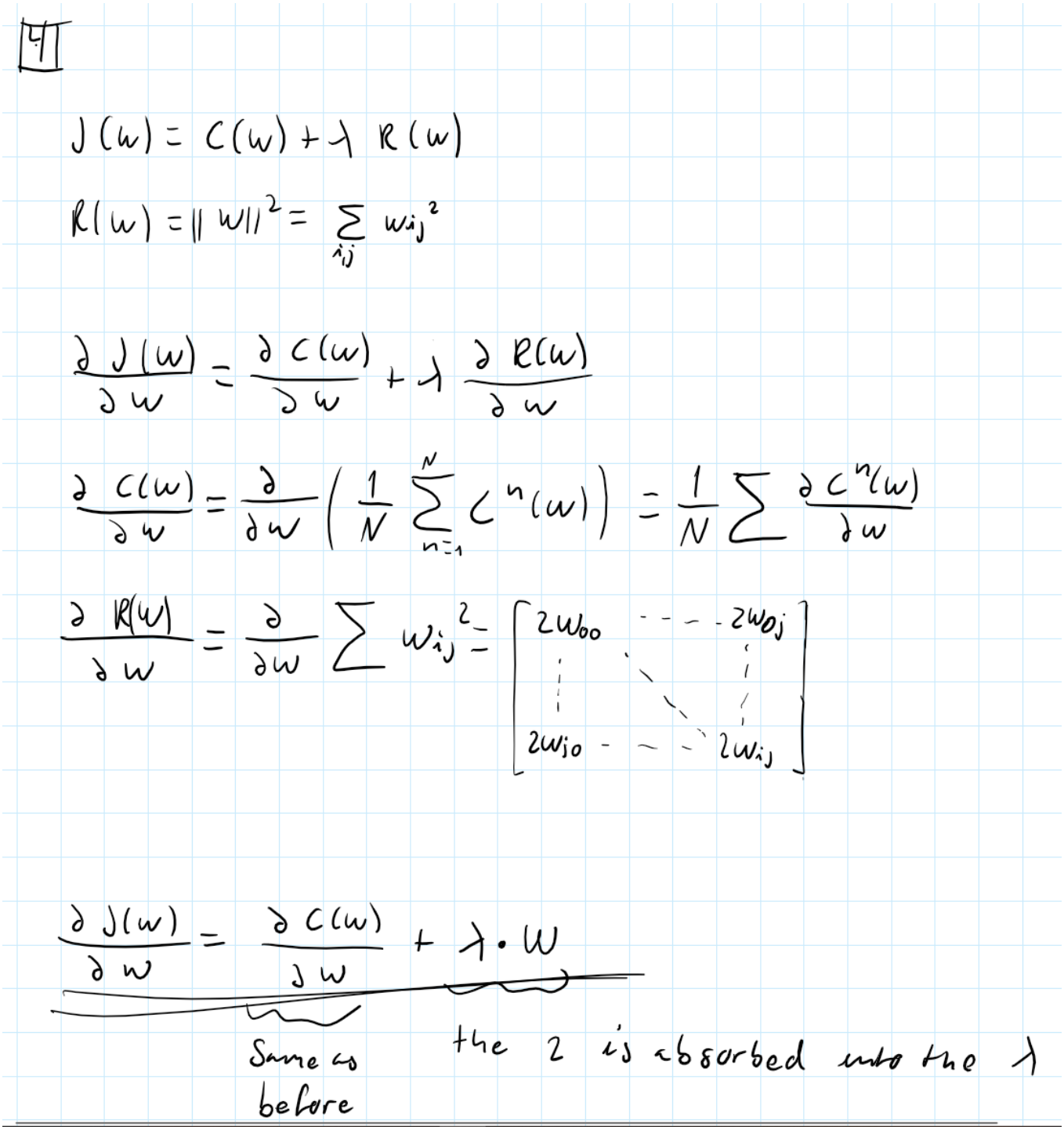


Task 3d)

The training accuracy is growing at a much faster rate than the validation accuracy. Although the validation accuracy is still rising it may start to fall off as the training accuracy continues to grow. This is because the neural net is very accurately fitting itself to the training set while not generalizing well to the validation set.

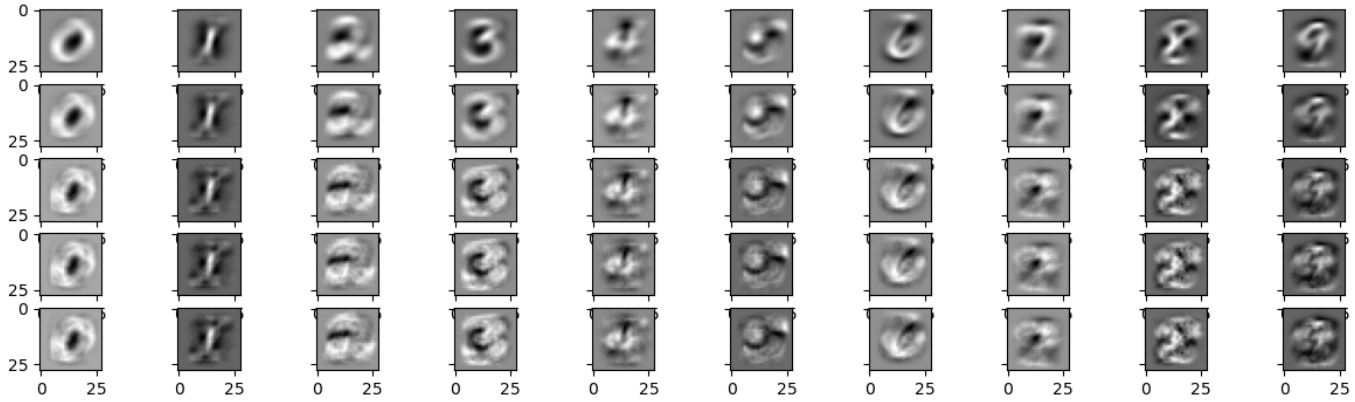
Task 4

Task 4a)

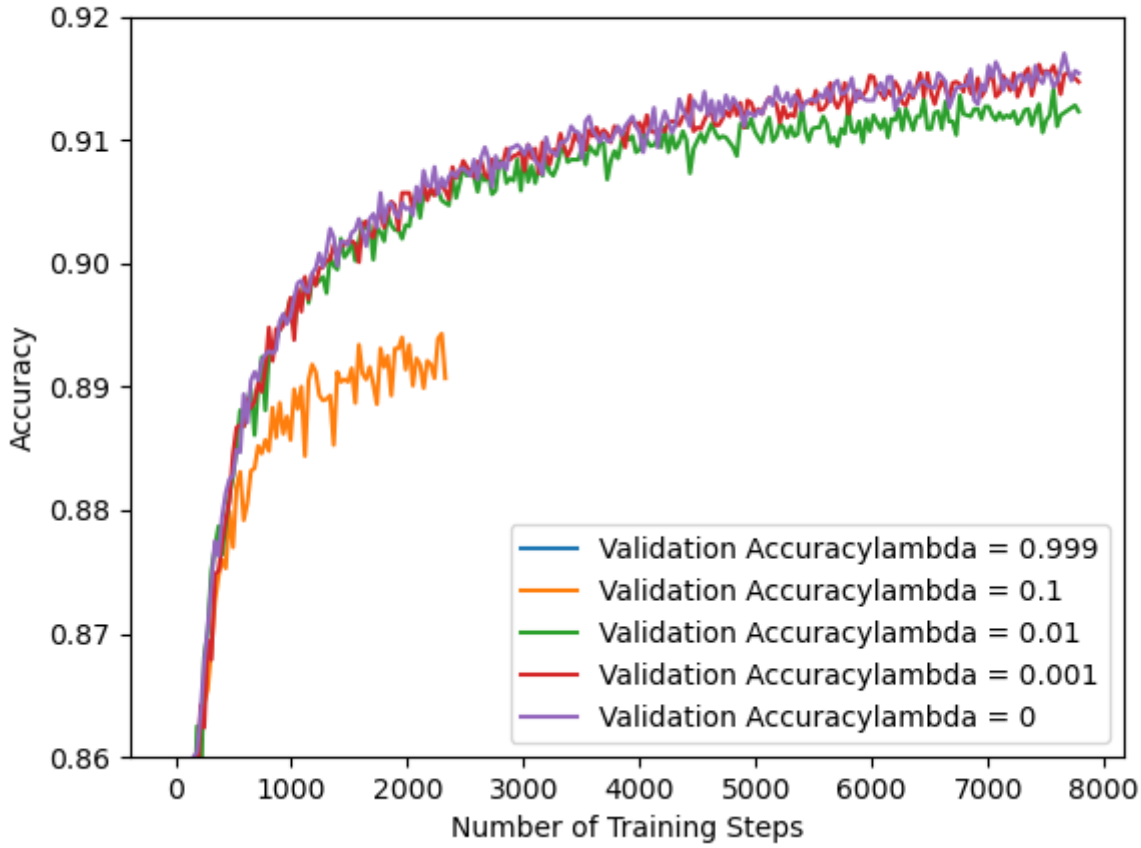


Task 4b)

Visualization of weights using lambda = [1, 0.1, 0.01, 0.001, 0]. Zero at the bottom



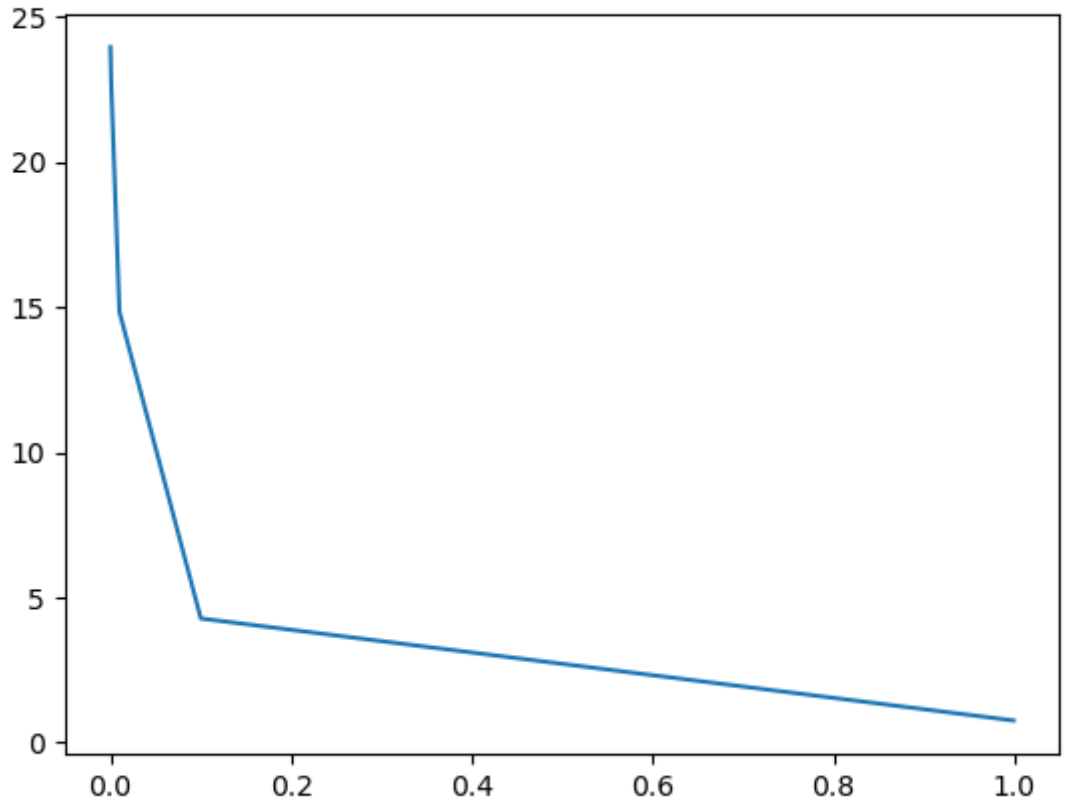
Task 4c)



Task 4d)

It makes sense that the validation accuracy drops off whenever we use l2 regularization as the weights that were trained using what is currently the validation set will diminish due to the weights having a cost.

Task 4e)



X axis is L2 and y axis is the norm of the weights squared.

As expected adding a cost for the size of the weights will reduce the size of the weights. We move in the opposite direction of the gradient which is the opposite direction of an increase in weights. Therefore high values of lambda will result in smaller weights.