# 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. 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)

$$\frac{\partial C^n(w)}{\partial w_i} = \frac{\partial C^n}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w_i}$$

Where:

$$\begin{aligned}\frac{\partial C^n}{\partial \hat{y}} &= -\frac{y^n}{\hat{y}^n} - \frac{1}{1-\hat{y}^n} \cdot (-1) + \frac{y^n}{1-\hat{y}^n} \cdot (-1) \\ &= \frac{1-y^n}{1-\hat{y}^n} - \frac{y^n}{\hat{y}^n} \\ \frac{\partial \hat{y}}{\partial w_i} &= \frac{\partial f(x^n)}{\partial w_i} = x_i^n \hat{y}^n (1-\hat{y}^n)\end{aligned}$$

Inserted in the original equation:

$$\begin{aligned}\frac{\partial C^n(w)}{\partial w_i} &= \left(\frac{1-y^n}{1-\hat{y}^n} - \frac{y^n}{\hat{y}^n}\right) x_i^n \hat{y}^n (1-\hat{y}^n) \\ &= (1-y^n) x_i^n \hat{y}^n - x_i^n y^n (1-\hat{y}^n) \\ &= x_i^n \hat{y^n} - x_i^n y^n \\ &= (\hat{y}^n - y^n) x_i^n \\ &= -(y^n - \hat{y}^n) x_i^n\end{aligned}$$

## task 1b)

Use the rule $ln(\frac{a}{b}) = ln(a) - ln(b)$ on the expression for $C^n(w)$:

$$\begin{aligned}\frac{\partial C^n(w)}{\partial w_{k,j}} &= -\frac{\partial}{\partial w_{k,j}} \sum_{p=1}^{K} y_p^n ln(\hat{y}_p^n) \\ &= -\frac{\partial}{\partial w_{k,j}} \sum_{p=1}^{K} y_p^n \left( ln(e^{z_p}) - ln(\sum_{p'}^{K} e^{z_{p'}}) \right) \\ &= -\sum_{p=1}^{K} \frac{\partial}{\partial w_{k,j}} y_p^n z_p + \sum_{p=1}^{K} \frac{\partial}{\partial w_{k,j}} y_p^n ln \left( \sum_{p'}^{K} e^{z_{p'}} \right)\end{aligned}$$

Find the derivative for the first term, which can be split into the case of $k = p$ and $k \neq p$:

$$\begin{aligned}\frac{\partial}{\partial w_{k,j}} y_p^n z_p &= \frac{\partial}{\partial w_{k,j}} y_p^n \left( w_p^T x \right) \\ &= y_p^n \frac{\partial}{\partial w_{k,j}} \sum_{j}^{I} w_{p,j} \cdot x_j = \begin{cases} 0, & k \neq p \\ y_k^n x_j^n, & k = p \end{cases}\end{aligned}$$

Substituting in for the first term and using the chain rule on the last term, we get:

$$\frac{\partial C^n(w)}{\partial w_{k,j}} = -y_k^n x_j^n + \sum_{p=1}^{K} y_p^n \frac{1}{\sum_{p'}^{K} e^{z_{p'}}} \frac{\partial}{\partial w_{k,j}} \left( \sum_{p'}^{K} e^{z_{p'}} \right)$$

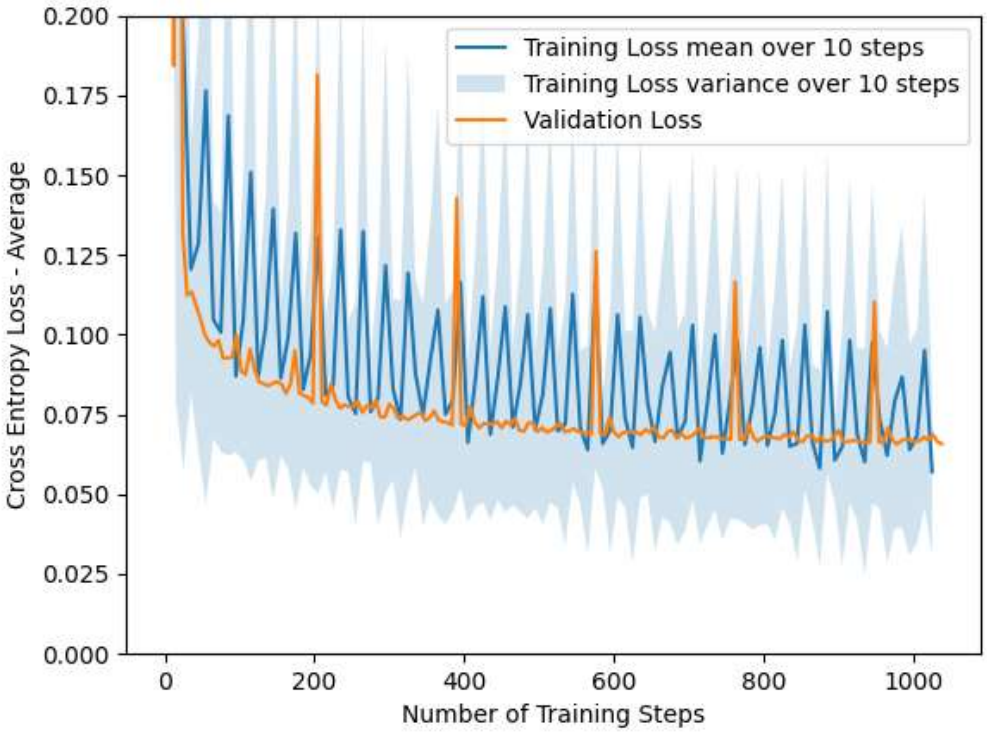
Again, we can split the last term into two cases, $k = p'$ and $k \neq p'$:

$$\frac{\partial}{\partial w_{k,j}} e^{z_{p'}} = \begin{cases} 0 & , \quad k \neq p' \\ e^{z_k} \cdot \frac{\partial}{\partial w_{k,j}} w_k x = e^{z_k} x_j & , \quad k = p' \end{cases}$$

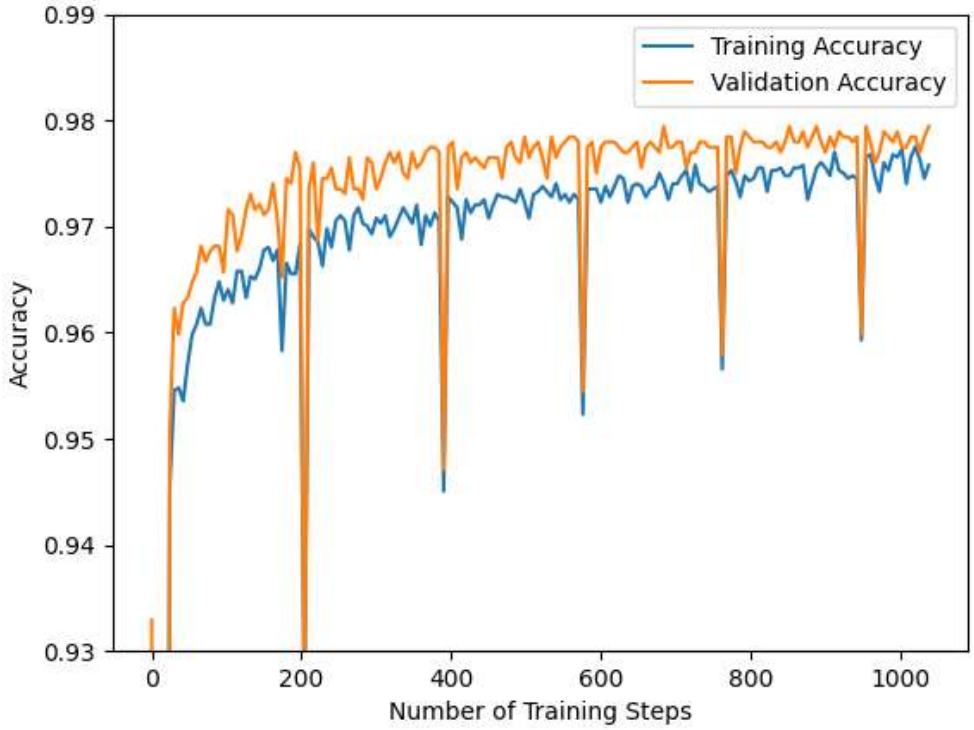
$$\begin{aligned}\frac{\partial C^n(w)}{\partial w_{k,j}} &= -y_k^n x_j^n + \sum_{p=1}^{K} y_p^n \frac{1}{\sum_{p'}^{K} e^{z_{p'}}} e^{z_k^n} x_j^n \\ &= -y_k^n x_j^n + \sum_{p=1}^{K} y_p^n \hat{y}_k^n x_j^n \\ &= -y_k^n x_j^n + \hat{y}_k^n x_j^n \\ &= -x_j^n (y_k^n - \hat{y}_k^n)\end{aligned}$$

# Task 2

## Task 2b)




## Task 2c)

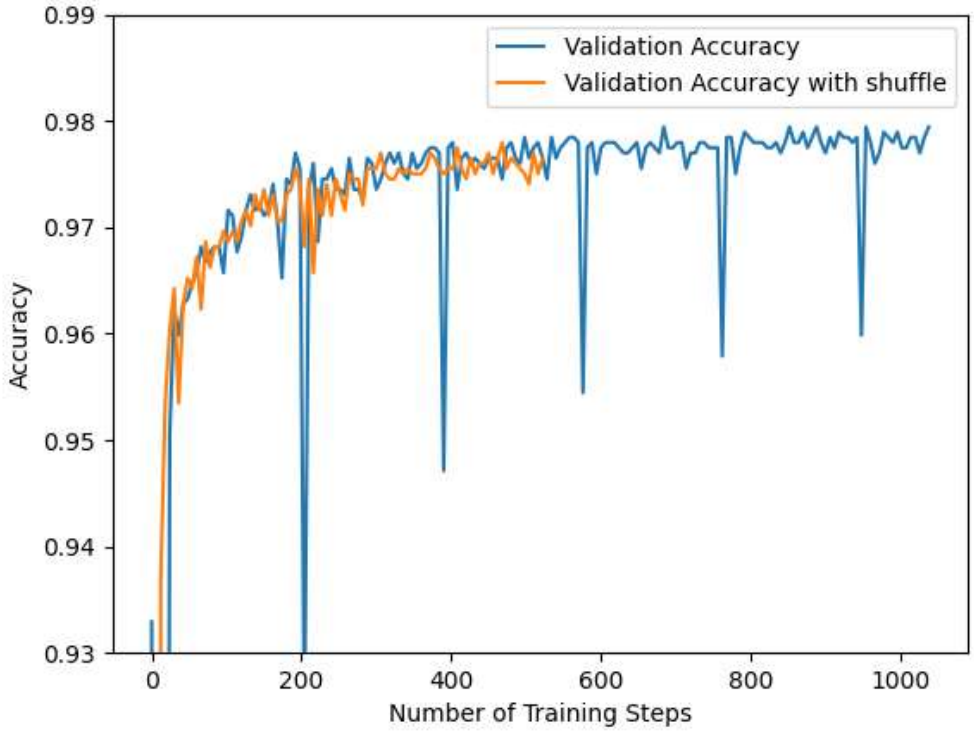



## Task 2d)

The early stopping kicks in at epoch 1038.

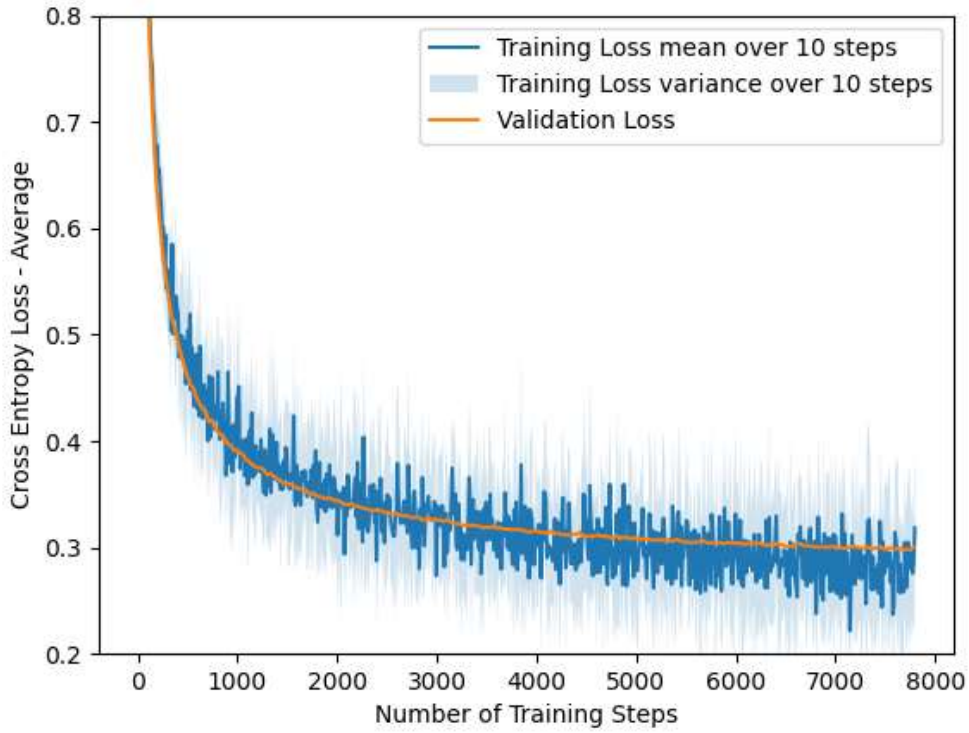
# Task 2e)

Not sure about this one.

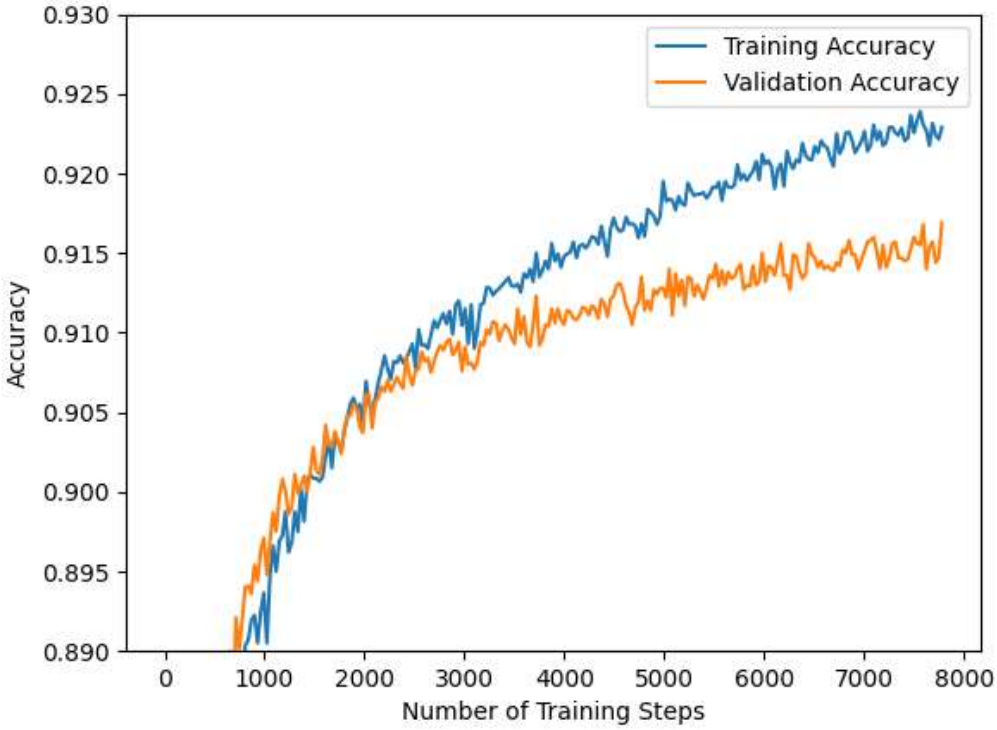



# Task 3

## Task 3b)




## Task 3c)




## Task 3d)

Yes, the training accuracy continues to rise, while validation accuracy does not improve. This indicates overfitting, as the model is only getting better at recognizing the training data instead of improving at identifying numbers in general.

# Task 4

## Task 4a)

Fill in image of hand-written notes which are easy to read, or latex equations here

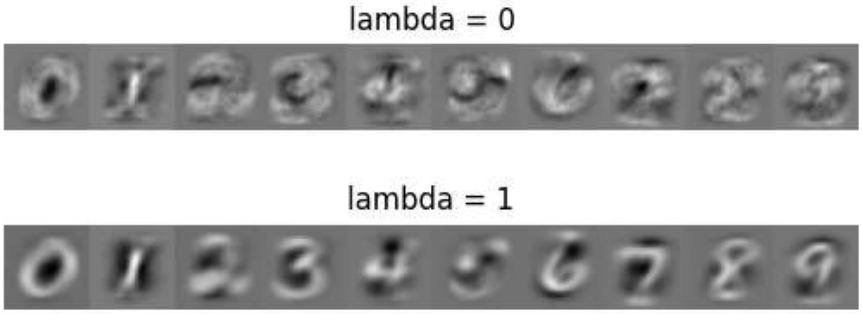
$$\frac{\partial J(w)}{\partial w} = \frac{\partial C(w)}{\partial w} + \lambda \frac{\partial R(w)}{\partial w}$$

$$\frac{\partial R(w)}{\partial w} = \frac{\partial}{\partial w} \sum_{i,j} w_{i,j}^2 = 2 \sum_{i,j} w_{i,j} = 2w$$

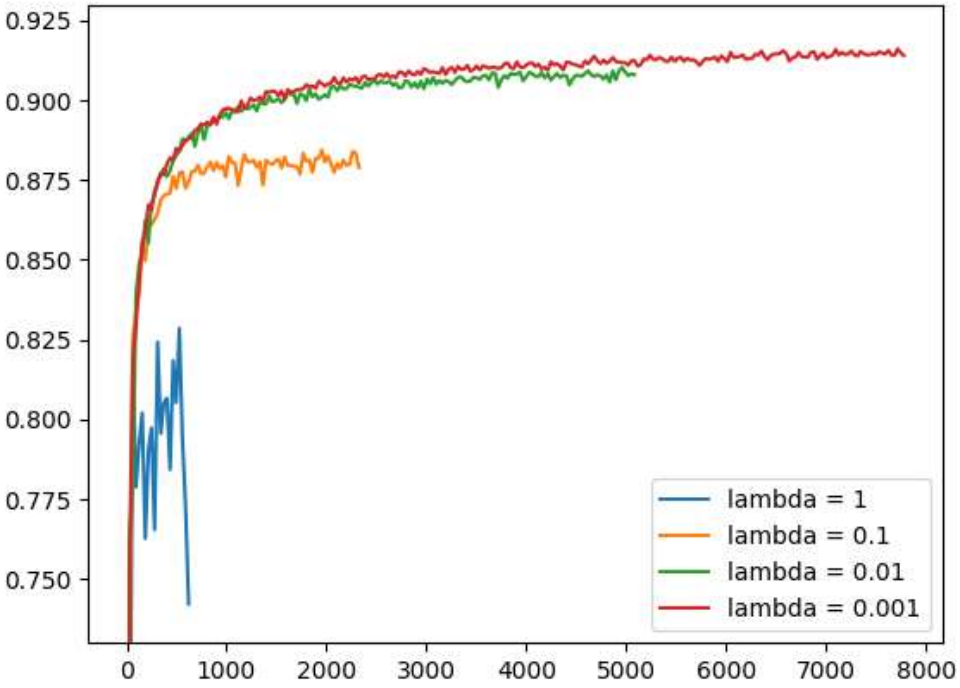
$$\frac{\partial J(w)}{\partial w} = \frac{\partial C(w)}{\partial w} + 2\lambda w$$

## Task 4b)

The weights get less noisy when introducing the term that penalizes a complex model. It makes sense that a complex, overtrained model will have very specific weights for each node leading to a more "noisy" visualization, whereas a less complex model will have a more "smooth" distribution between the weights. The less complex model will then pick up on more broad shapes and might be better outside of training.




## Task 4c)

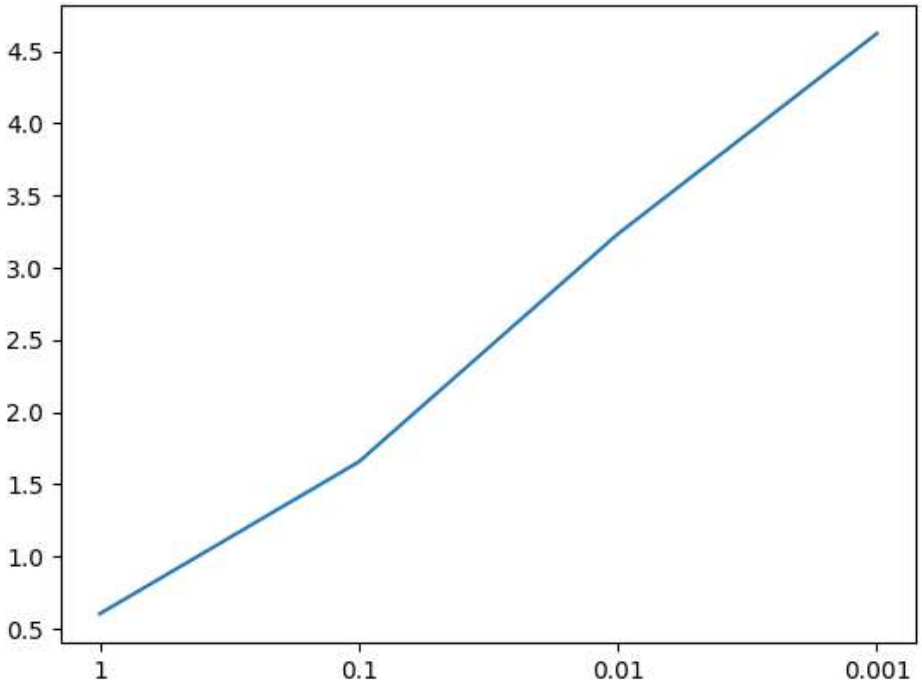



## Task 4d)

I think this is because we don't have a very complex model to begin with, and applying too much regularization might "dumb down" the model too much to work in a general example.

## Task 4e)

The norm increases with lower $\lambda$-values.


4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
1
0.1
0.01
0.001