Assignment 1 Report

Task 1

task 1a)

We have given:

$$C^n = -((y^n \ln \hat{y}^n) + (1 - y^n)\ln(1 - \hat{y}^n))$$

and:

$$\frac{\partial f(x^n)}{\partial w_i} = x_i^n f(x^n)(1 - f(x^n))$$

$$\hat{y}^n = f(x^n)$$

we start by calculating $\frac{\partial C^n(w)}{\partial f(x^n)}$:

$$\frac{\partial C^n(w)}{\partial f(x^n)} = \frac{\partial C^n(w)}{\partial \hat{y}^n}$$

$$\frac{\partial C^n(w)}{\partial \hat{y}^n} = \frac{\hat{y}^n - y^n}{(1 - \hat{y}^n)\hat{y}^n}$$

From the chain rule we know that:

$$\frac{\partial C^n(w)}{\partial w_i} = \frac{\partial C^n(w)}{\partial f(x^n)} * \frac{\partial f(x^n)}{\partial w_i}$$

Therefore we have that

$$\frac{\partial C^n(w)}{\partial w_i} = \frac{\hat{y}^n - y^n}{(1 - \hat{y}^n)\hat{y}^n} * x_i^n f(x^n)(1 - f(x^n))$$

since $\hat{y} = f(x)$ we get:

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

task 1b)

$$\hat{y}_k = \frac{e^{z_k}}{\sum_{k'}^{K} e^{z_k}{'}}$$

we start with the derivation of $\hat{y}_k$, where $\hat{y}_k$ represent the probability that x is a member of class k.

We will break the derivation of the softmax into two stages, namely $k = k'$ and $k \neq k'$:

$$k = k' :$$

$$\frac{\partial \hat{y}_k}{\partial z_k} = \frac{e^{z_k} * \sum_{k'}^{K} e^{z_k}{'} - e^{z_i} * e^{z_k}}{(\sum_{k'}^{K} e^{z_k}{'})^2}$$

$$= \frac{e^{z_k}}{\sum_{k'}^{K} e^{z_k}{'}} * \frac{\sum_{k'}^{K} e^{z_k}{'} - e^{z_k}}{\sum_{k'}^{K} e^{z_k}{'}}$$

$$= \hat{y}_k(1 - \hat{y}_k)$$

$$k \neq k' :$$

$$\frac{\partial \hat{y}_k}{\partial z_{k'}} = \frac{0 - e^{z_k} e^{z_k}{'}}{(\sum_{k'}^{K} e^{z_k}{'})^2}$$

$$= -\hat{y}_k \hat{y}_{k'}$$

We can now use these to simplify our equation when derivating the cross entropy.

The cross-entropy cost function for multiple classes is defined as:

$$C(w) = \frac{1}{N} \sum_{n=1}^{N} C^n(w)$$

where

$$C^n(w) = -\sum_{k=1}^{K} y_k^n \ln \hat{y}_k^n$$

$$\frac{\partial C^n(w)}{\partial z_k} = -\frac{\partial y_k}{\partial z_k} * \frac{y_k}{\hat{y}_k} - \sum_{k \neq k'}^{K} \frac{y_{k'}}{\hat{y}_k} * \frac{\partial y_k}{\partial z_{k'}}$$

We simplify the equation with our result from the derivative of softmax:

$$\frac{\partial C^n(w)}{\partial z_k} = -y_k + \hat{y}_k \sum_{k=1}^{K} y_k$$

$$= -y_k + \hat{y}_k$$

We are given:

$$z_k = \sum_i^I w_{k,i} * x_i$$

We then get:

$$\frac{\partial z_k}{\partial w_{k,i}} = x_i$$

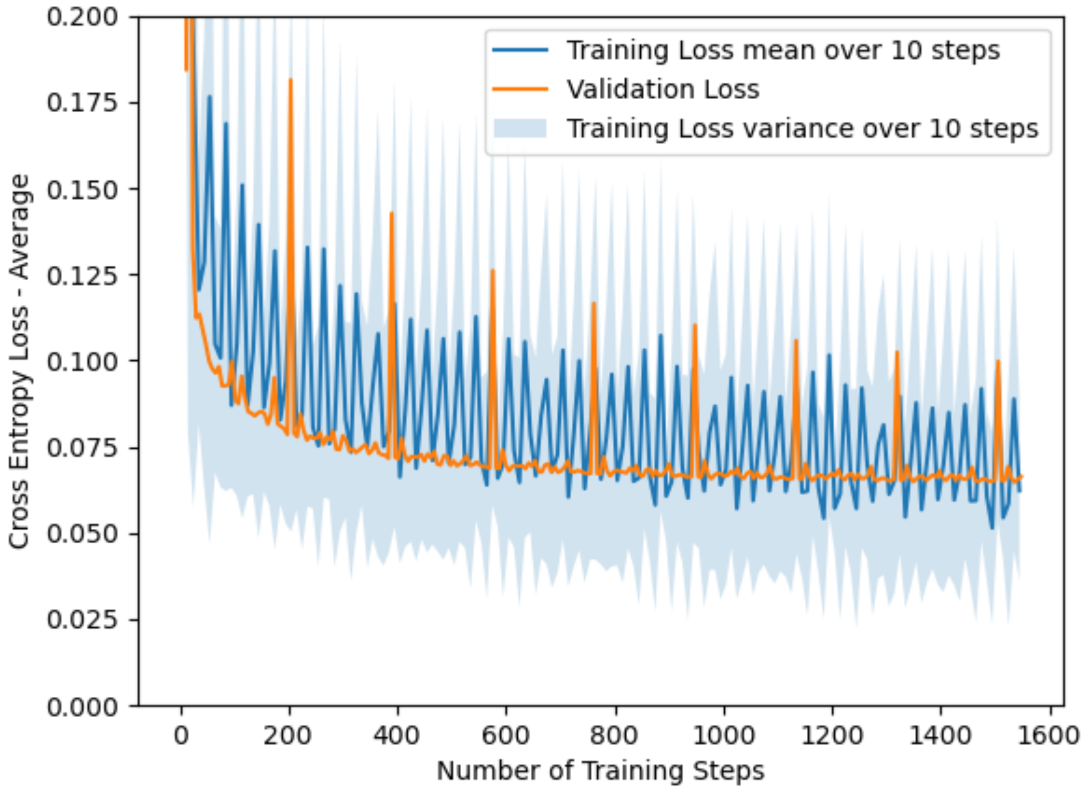
$$\frac{\partial C^n(w)}{\partial w_{k,i}} = \frac{\partial z_k}{\partial w_{k,i}} * \frac{\partial C^n(w)}{\partial z_k}$$

$$= x_i * (-y_k + \hat{y}_k)$$

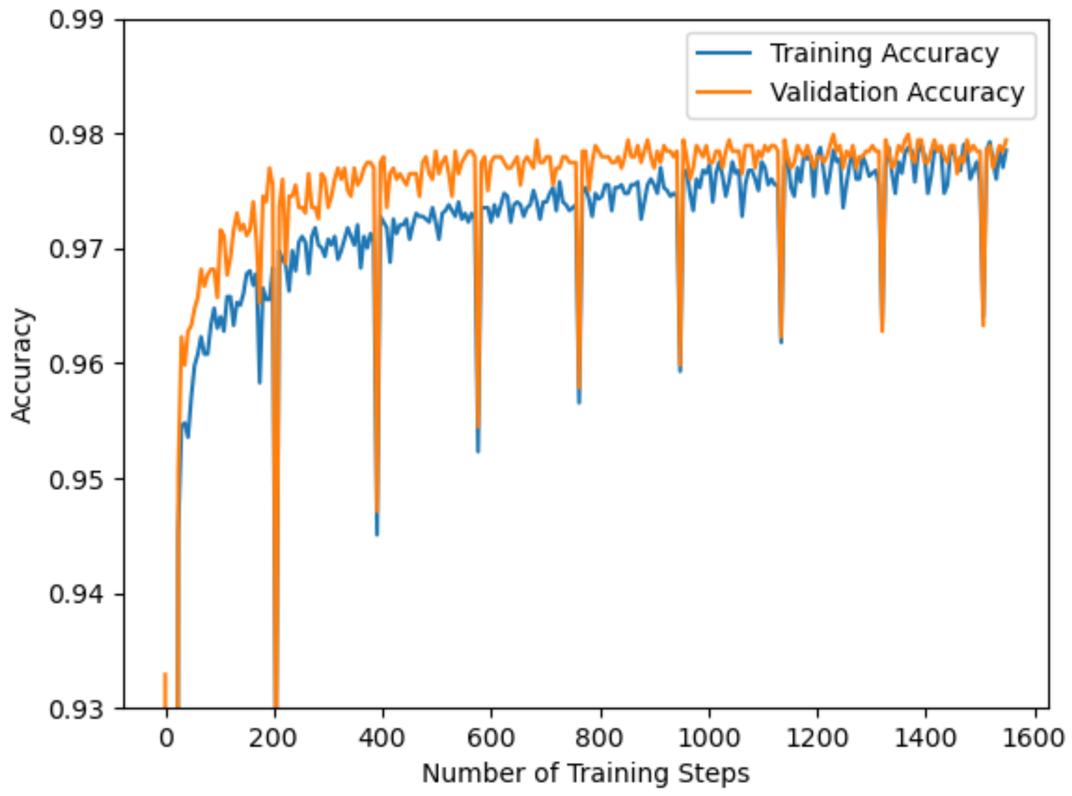
$$= \underline{\underline{x_i^n(y_k^n - \hat{y}_k^n)}}$$

Task 2

Task 2b)

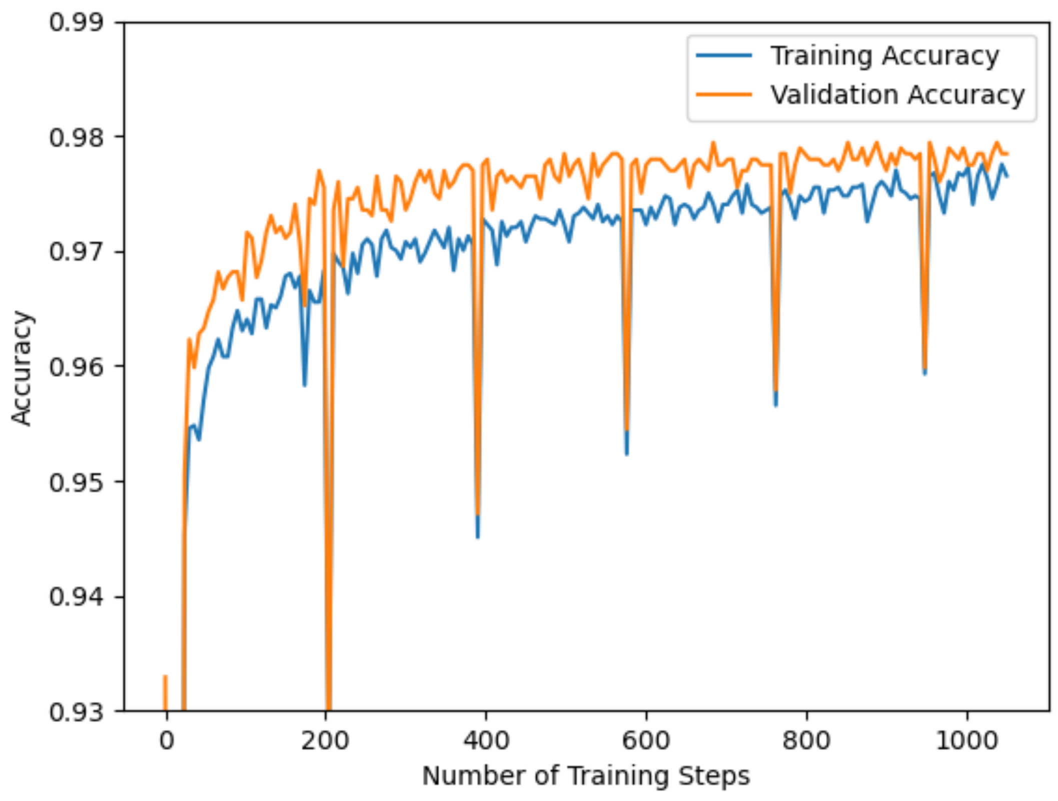


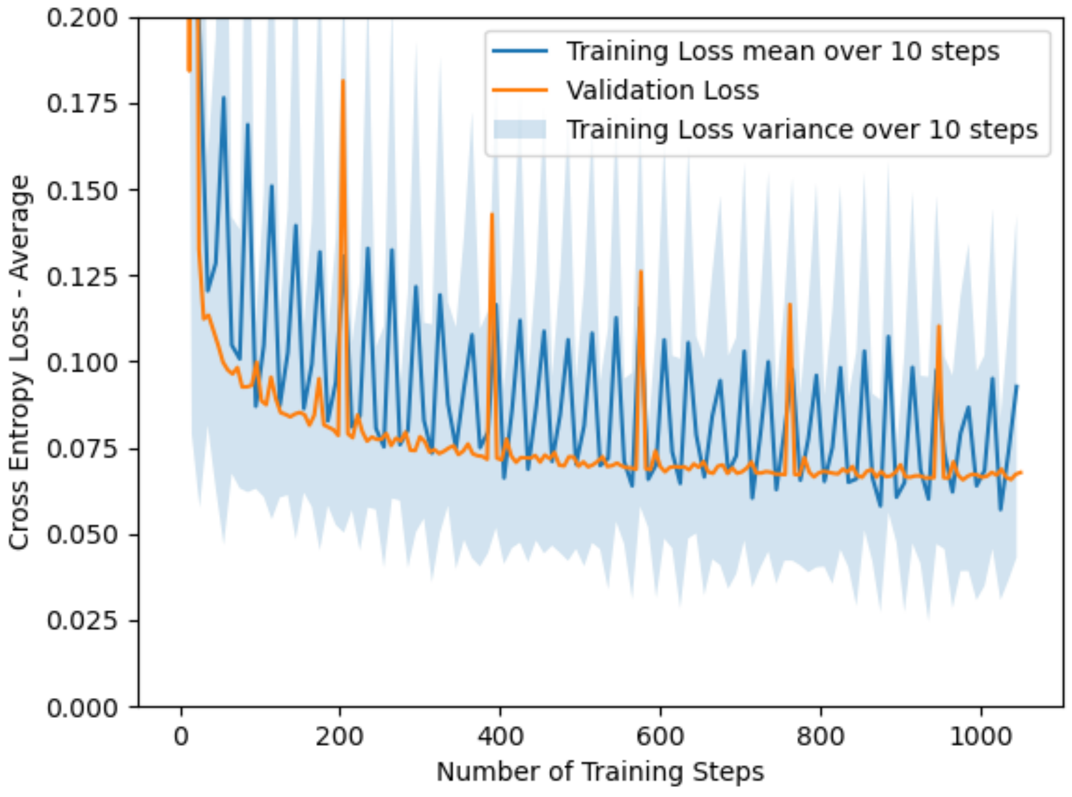
Task 2c)



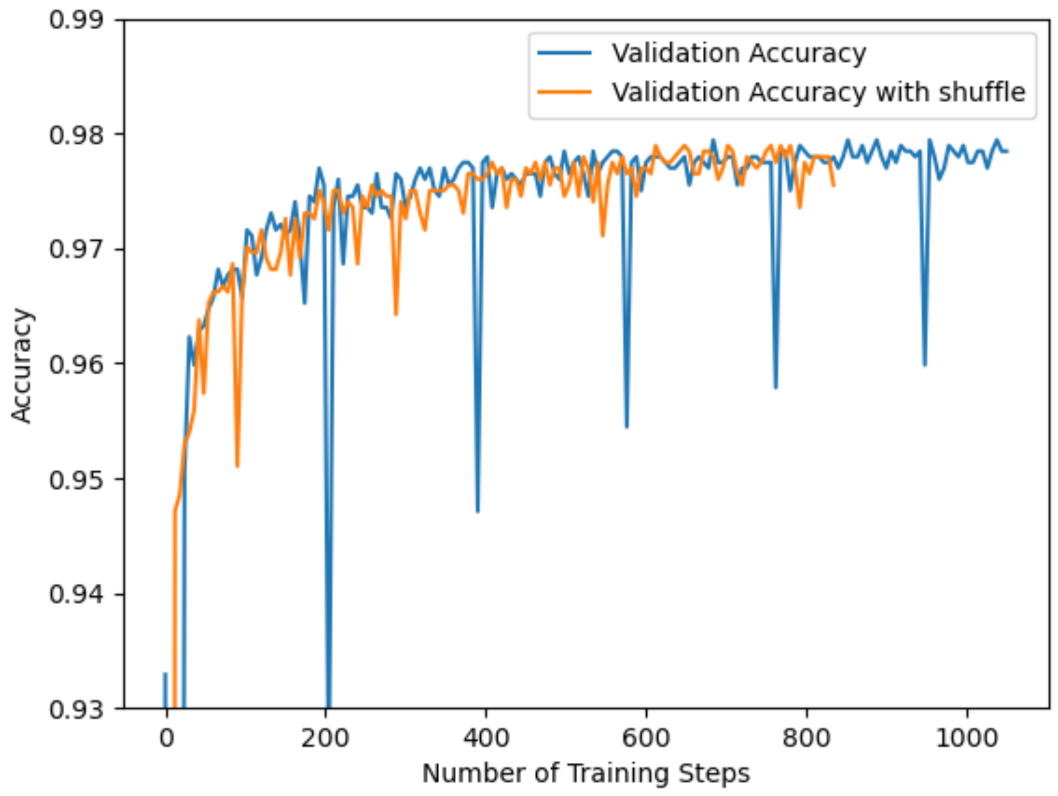
Task 2d)

In the un-shuffled case, the training stops after 34 epochs using early stopping. The plots are displayed below


Training Accuracy
Validation Accuracy
Accuracy
Number of Training Steps
0.99
0.98
0.97
0.96
0.95
0.94
0.93
0
200
400
600
800
1000


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

Task 2e)

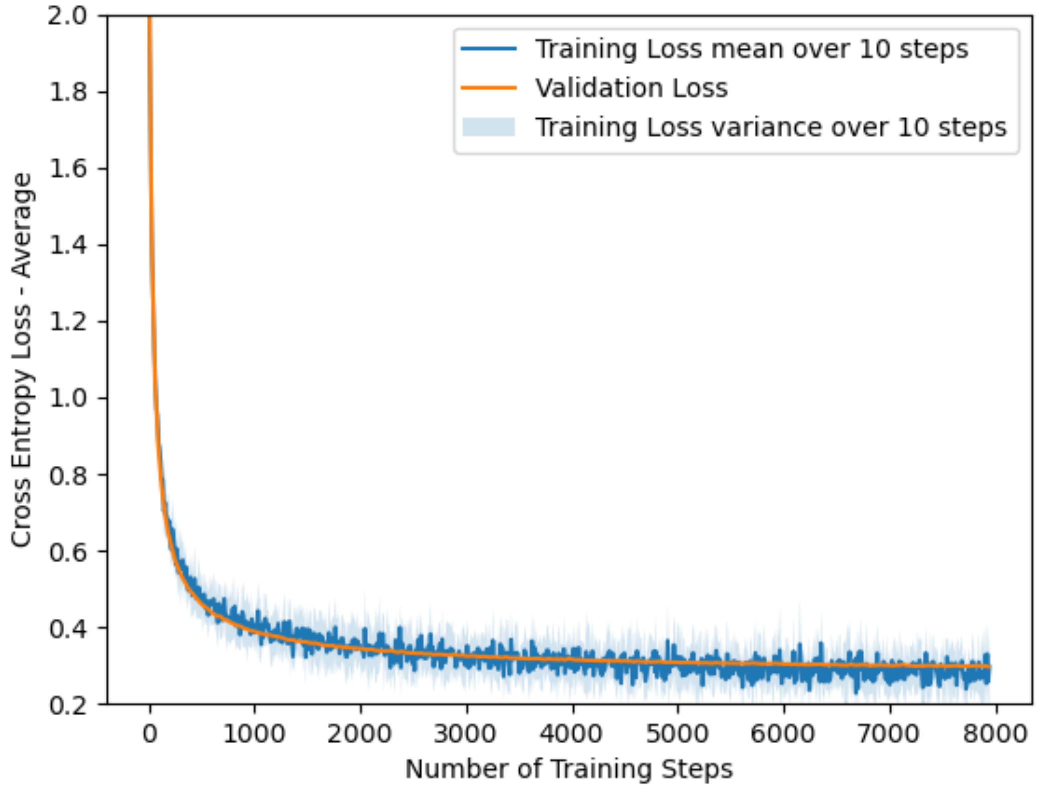


Shuffling the gives less "spikes". In the binary case, if 2/10 images have label 1 and the remaining 8/10 have the label 0 which is the case with this dataset then classifying 0 will generally hava a high successrate. This may be the reason that the unshuffled case gives periodic drops in accuracy and corresponding periodic spikes in the cost function. The spikes do become smaller and smaller as training proceeds, so this wrongful naive guessing occurs less frequently as the model is trained.

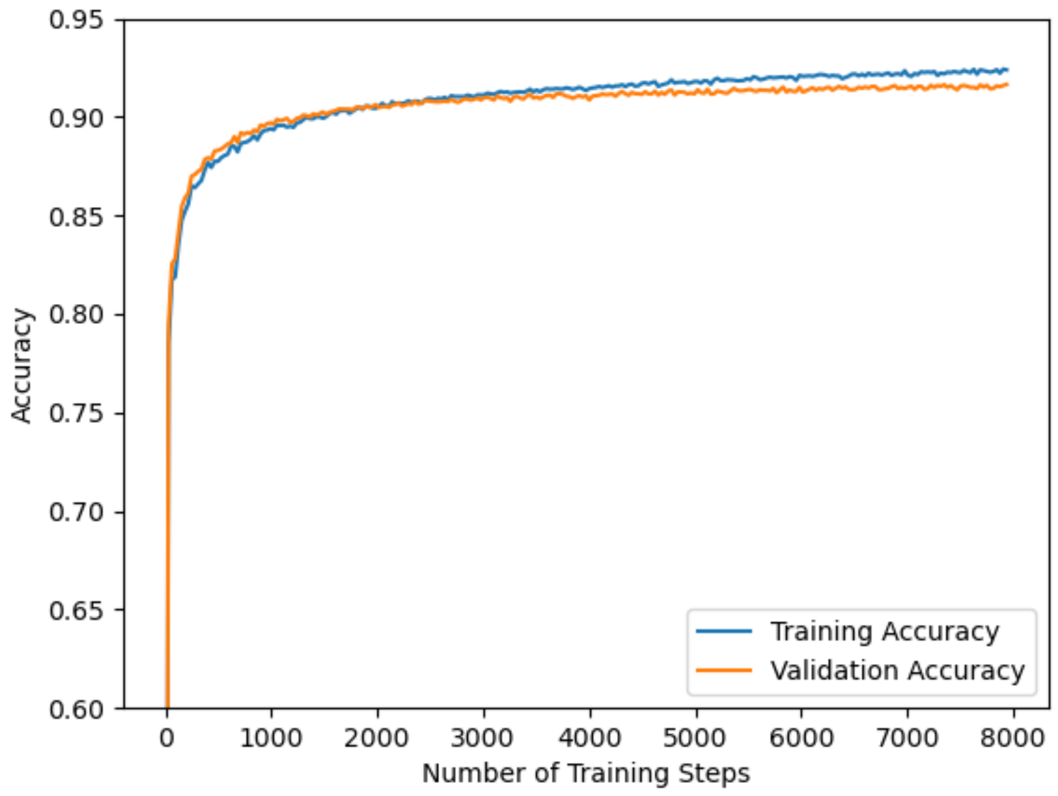
When shuffling the dataset, these periodic spikes disappear, as the pattern is no longer predictable. Using this technique, the different labels occur randomly, making the curve more "Gaussian".

Task 3

Task 3b)


Training Loss mean over 10 steps
Validation Loss
Training Loss variance over 10 steps
Cross Entropy Loss - Average
Number of Training Steps
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0
1000
2000
3000
4000
5000
6000
7000
8000

Task 3c)



Task 3d)

From the accuracy plot, it seems that the training accuracy outperforms the validation accuracy after roughly 3000 training steps. Notice how the training accuracy keeps improving while the validation accuracy remains more or less constant (it is still increasing but only marginally). This can be a sign of overfitting, but seeing that the validation accuracy - and loss keeps improving, it is not overfitting yet.

Seeing this tendency of overfitting is less clear from the training - and validation loss, but it can arguably be said that the training loss outperforms the validation loss in this case as well.

There are tendencies of overfitting, but in this case overfitting is not an issue. Furthermore, since we are using Early Stopping from task 2 during training in this task as well, the training stops before overfitting becomes too visible. If training continued, however, the validation loss could actually converge or even start increasing while the training loss keeps decreasing. This is not the case here.

Task 4

Task 4a)

$$J(w) = C(w) + \lambda * R(w)$$

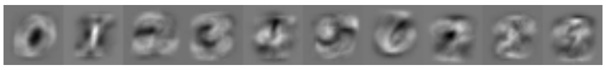
$$\frac{\partial J(w)}{\partial w} = \frac{\partial C(w)}{\partial w} + \lambda * \frac{\partial R(w)}{\partial w}$$

$$= -1/N * \sum_{n=1}^{N} -x_j^n(y_k^n - \hat{y}_k^n) + \lambda 2w$$

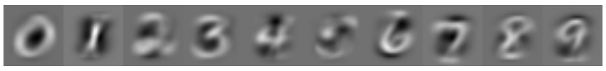
Task 4b)

$\lambda = 1$ appears clearer than $\lambda = 0$.

$\lambda = 0$:

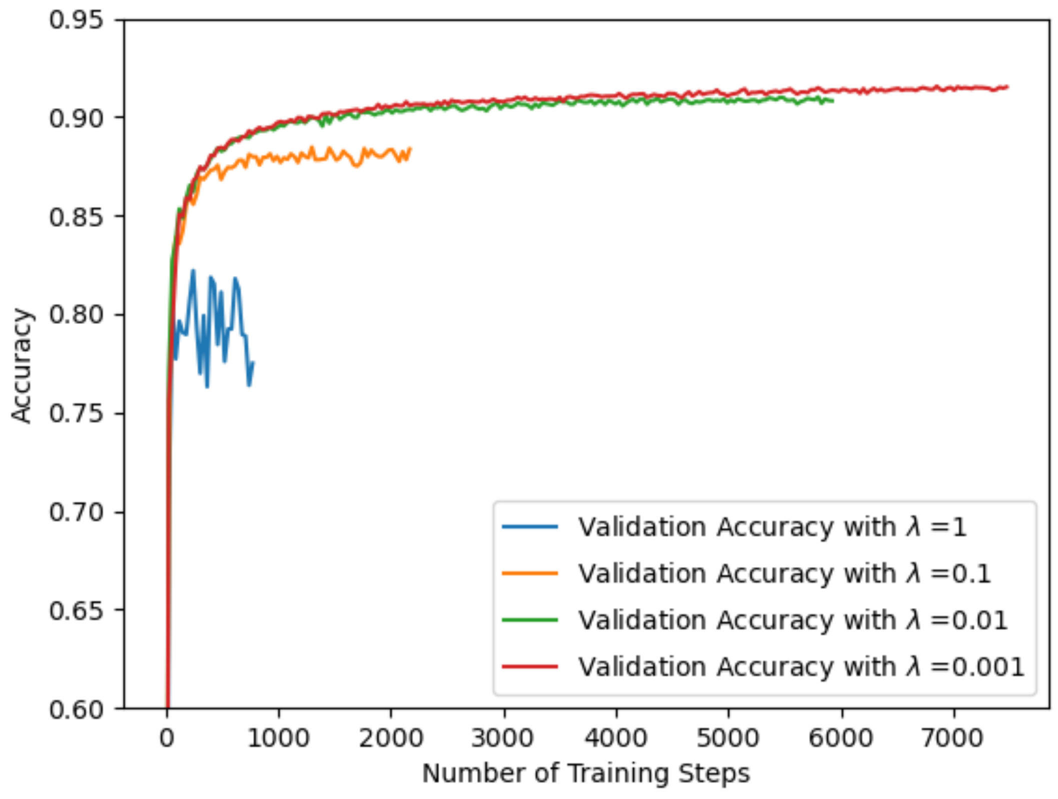


$\lambda = 1$:



This is most likely due to the fact that for higher $\lambda's$, the weights will tend to be as low as possible. This means that for a weight to increase, the effect of this has to "overcome" the negative effect of increasing the weight. With λ = 0 on the other hand, the weights may increase if this yields a slightly lower $C(w)$, and a higher weight will not give a higher cost.

Task 4c)

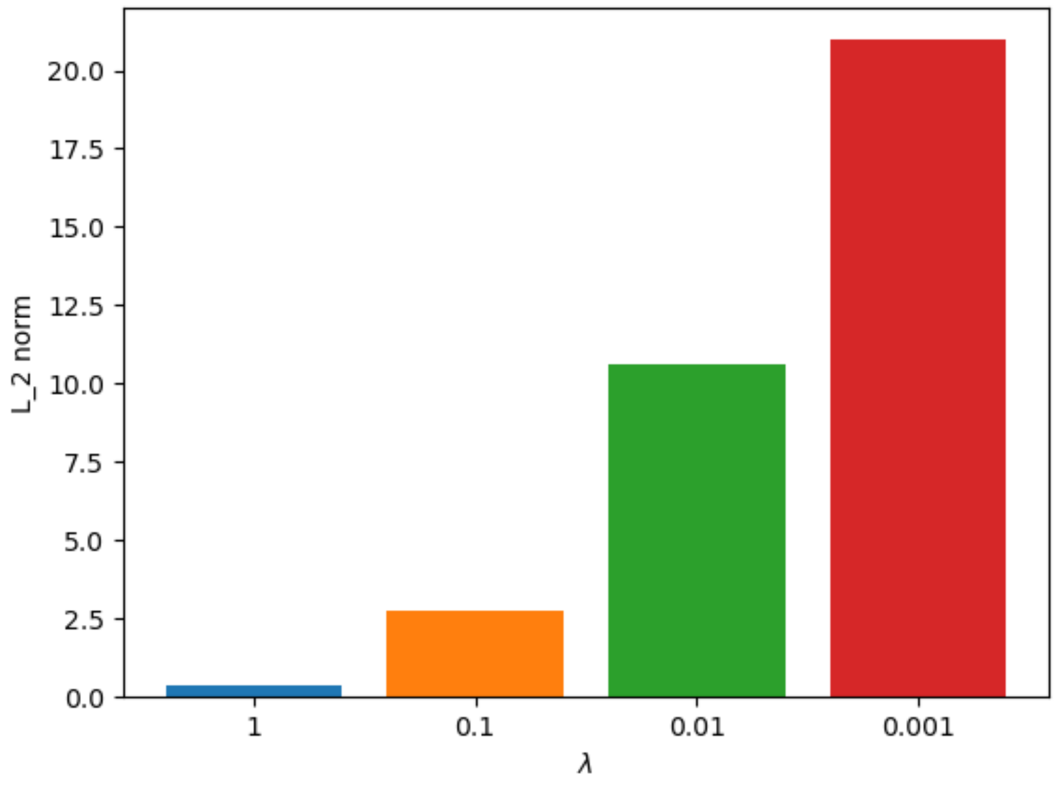


Task 4d)

Regularization is a way of bounding the weights by adding a penalty in the cost function that increases with a higher $L_2 - Norm$ of the weights. This forces the optimal point of the cost function away from the original minimum. The higher the λ, the "harder" the weights are pulled away from a minimum.

Note that Early Stopping is implemented, which is why the different curves stop after the training converges.

Task 4e)



Higher lambdas do - not surprisingly - give a lower L_2 norm of w. This has to do with that the cost function penalizes the magnitude of w, and it becomes clear from the plots that this impacts w significantly.