TDT4265 - Assignment 1

Group 200 - Jostein Nordal Lysberg & Ludvig Vik Løite

Task 1

task 1a)

$$\underline{1.a} \qquad C^n(w) = -y^n \ln(\hat{y}^n) + (1-y^n)\ln(1-\hat{y}^n),$$

where $\hat{y} = f(x)$

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

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

$$= x_i^n f(x^n)(1-f(x^n))\left(-y^n \frac{1}{f(x^n)} + (1-y^n)\frac{1}{1-f(x^n)}\right)$$

$$= x_i^n f(x^n)(1-f(x^n))\left(\frac{-y^n(1-f(x^n)) + f(x^n) - y^n f(x^n)}{f(x^n)(1-f(x^n))}\right)$$

$$= x_i^n \left(f(x^n) - y^n\right)$$

$$= -x_i^n \left(y^n - \hat{y}^n\right)$$

task 1b)

1.b

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

$$= -\sum_{k=1}^{K} y_k^n \cdot \underbrace{\ln\left(e^{z_k}\right)}_{z_k} + \ln\left(\sum_{k'=1}^{K} e^{z_{k'}}\right) \underbrace{\sum_{k=1}^{K} \hat{y}_k^n}_{1}$$

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

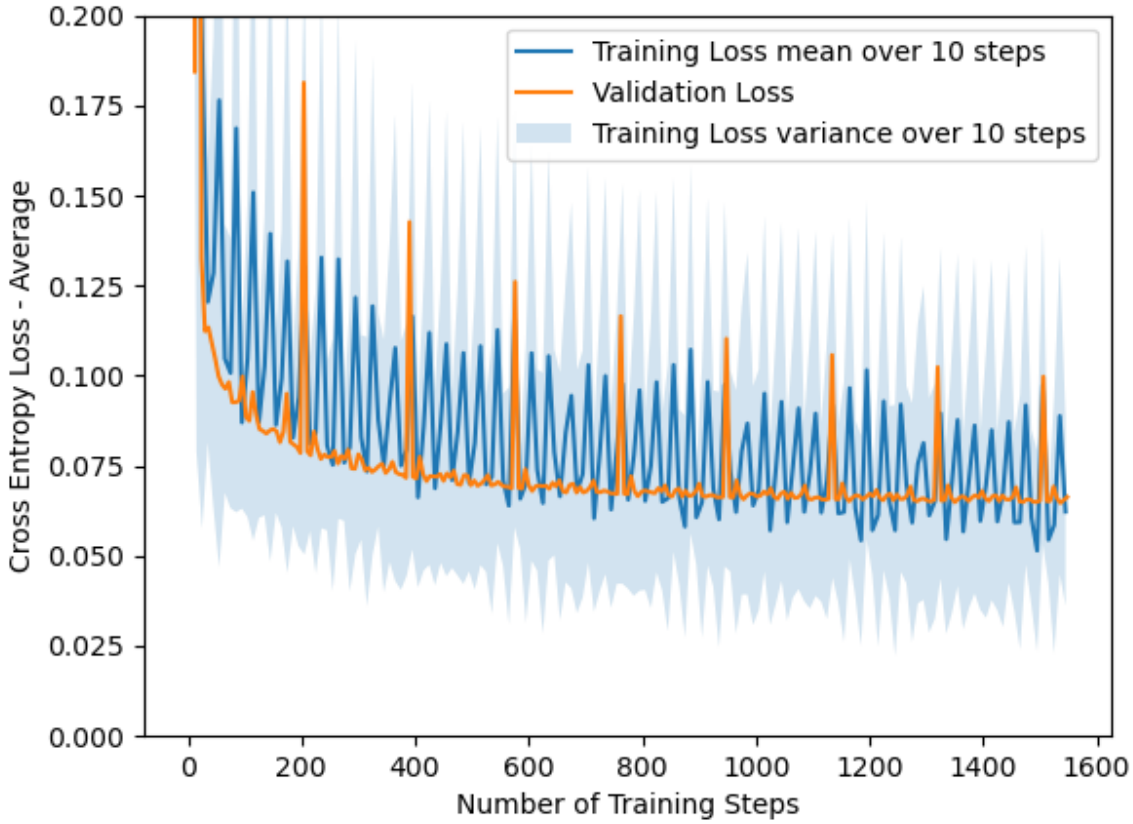
$$\frac{\partial C^n(w)}{\partial w_{kj}} = \frac{\partial C^n(w)}{\partial z_k} \cdot \frac{\partial z_k}{\partial w_{kj}}$$

$$= \left(-y_k^n + \frac{e^{z_k}}{\sum_{k'=1}^{K} e^{z_{k'}}}\right) x_j^n$$

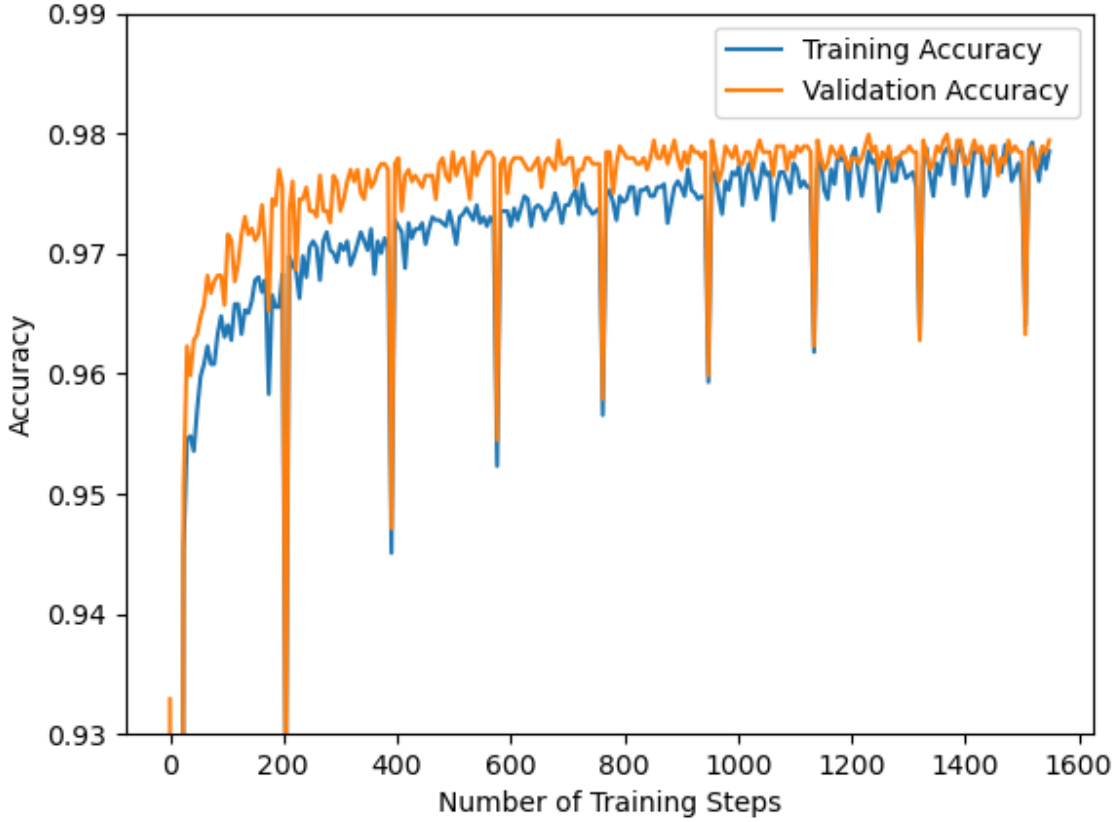
$$= -x_j^n \left(y_k^n - \hat{y}_k^n\right)$$

Task 2

Task 2b)


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
1200
1400
1600

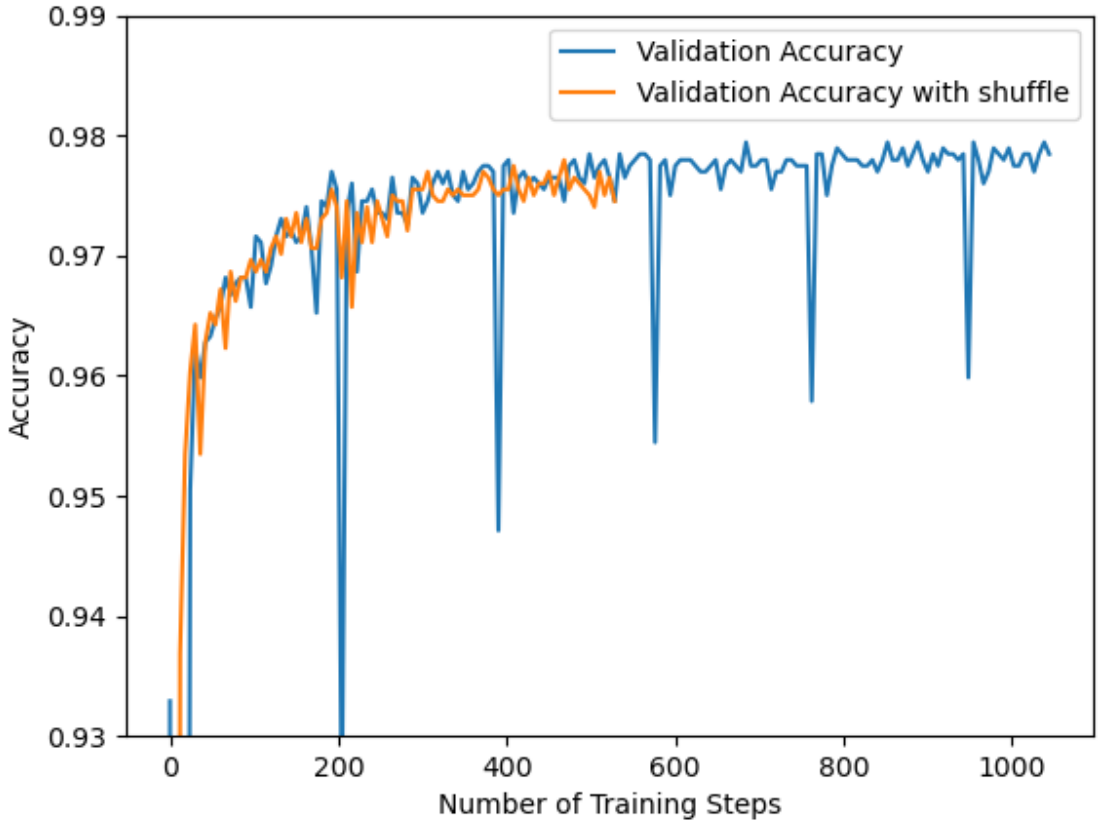
Task 2c)



Task 2d)

Early stopping kicks in at epoch number 33. At this point, the validation loss had not improved over the last 10 validation checks. The validation cross entropy stabilized around 0.067.

Task 2e)



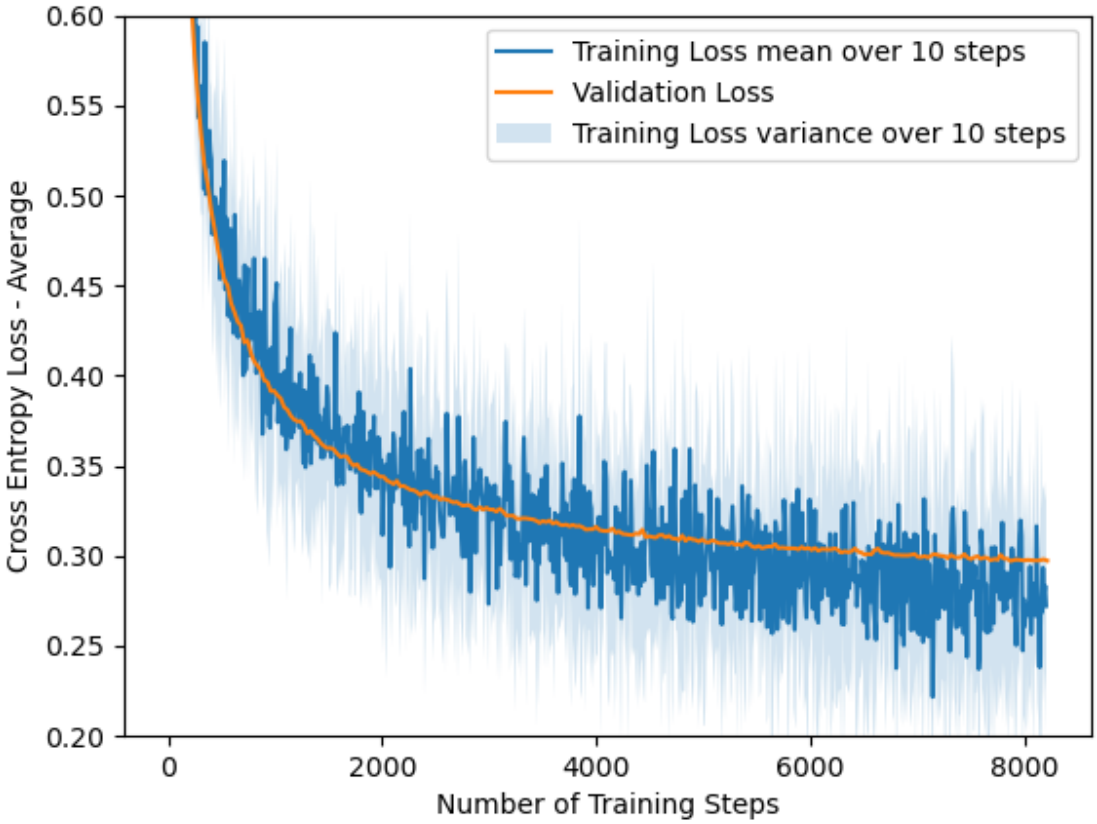
As is apparent in both the validation loss and the train accuracy, our model suffers greatly from performance spikes once every sixth epoch.

We believe these spikes are caused by bad images in one of the 31 batches of our training data set, which makes sense considering the spike in training loss once every epoch. Since validation is performed on an entirely different data set only once every fifth training step, this problem does not become apparent until we run a validation step right after training our model on this bad batch. With 31 batches in our training data, and running validation once every fifth step, this accounts for the performance spike in validation loss and accuracy once every 186th step.

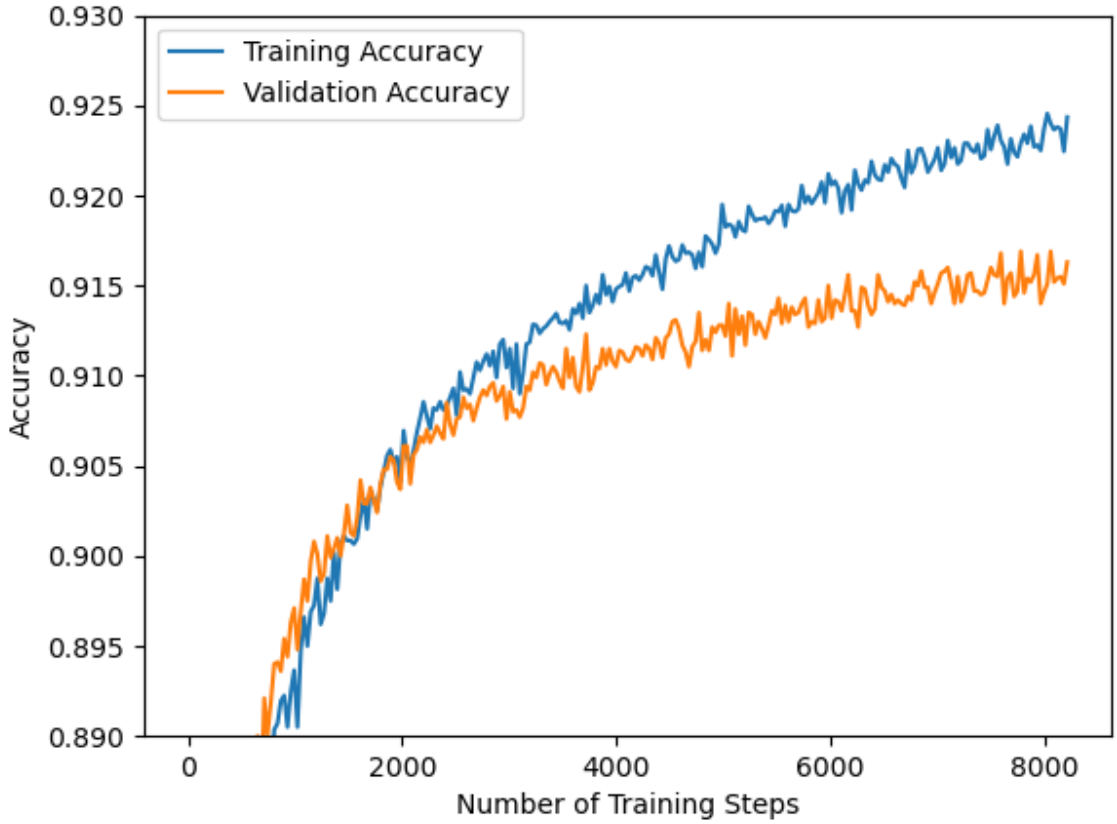
By introducing data set shuffling, we can avoid this problem to some extent. We will of course still suffer from the same bad data, but spread over different batches every epoch, the model is not consistently updated with bad weights at regular intervals. This lead to the validation accuracy having much less spikes when including dataset shuffling.

Task 3

Task 3b)


Training Loss mean over 10 steps
Validation Loss
Training Loss variance over 10 steps
Cross Entropy Loss - Average
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0
2000
4000
6000
8000
Number of Training Steps

Task 3c)



Task 3d)

As can be seen in the above two figures, there are some signs of overfitting in our model.

It is most apparent in the figure showing accuracy, where the training accuracy diverges from the validation accuracy around 2000 training steps. After this point, the training accuracy is increasing much faster than the validation accuracy. In the figure cross entropy loss, one can see that the validation loss flats out at the same time as training loss keep descreasing. Both are signs of overfitting.

However, since both the validation accuracy and the validation cross entropy loss keeps improving until around 8000 training steps, there is no reason to stop training earlier than we have.

Task 4

Task 4a)

4.a

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

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

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

where $\frac{\partial C}{\partial w}$ was shown in task 1a to be $\frac{\partial C}{\partial w} = -\frac{x^T(y-\hat{y})}{N}$

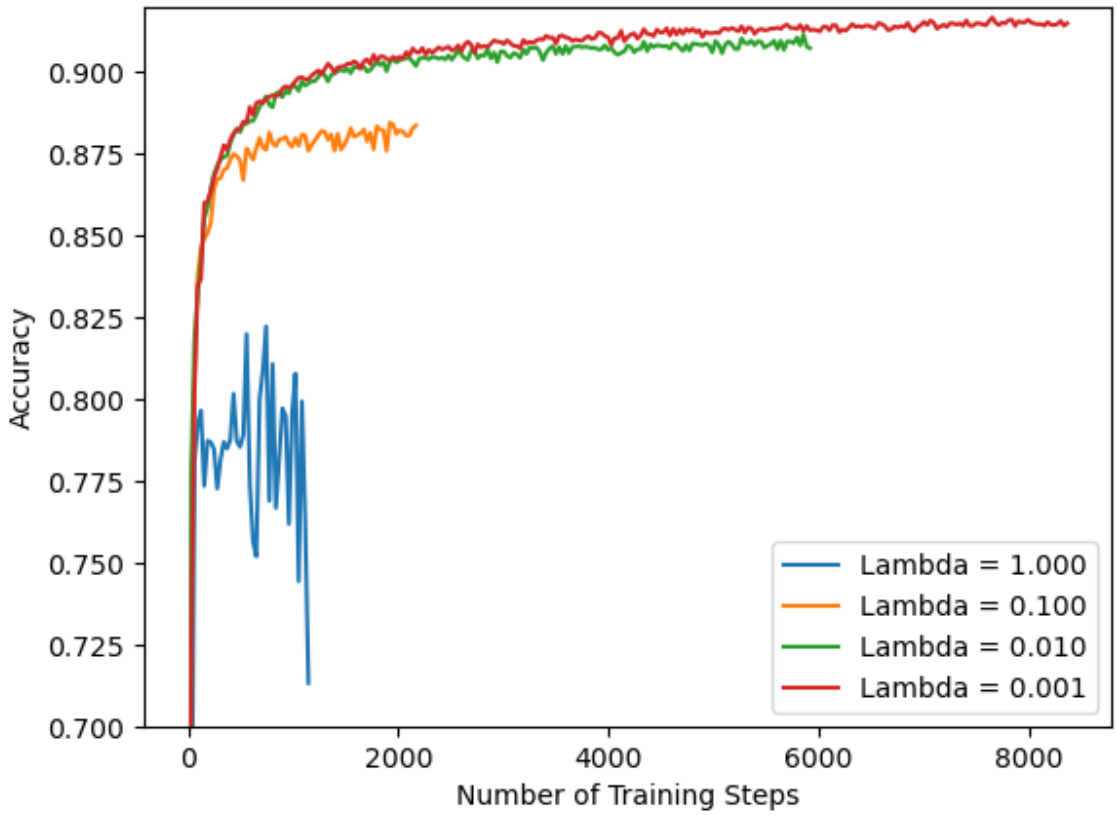
$$\frac{\partial R}{\partial w} = 2w$$

$$\frac{\partial J}{\partial w} = -\frac{x^T(y-\hat{y})}{N} + 2\lambda w$$

Task 4b)

Lambda is proportional to the degree of complexity penalization of our model. The upper row of the figure shows the weights of the model when lambda = 0, which means that we have not included any regularization. The weights are very specialised to the data set. The bottom row shows the weights when lambda = 1. The weights are more generalized. When visualized, less complex weights from higher values of lambda consititute smoother and less noisy images. The generalization seems to "average" out the weights.

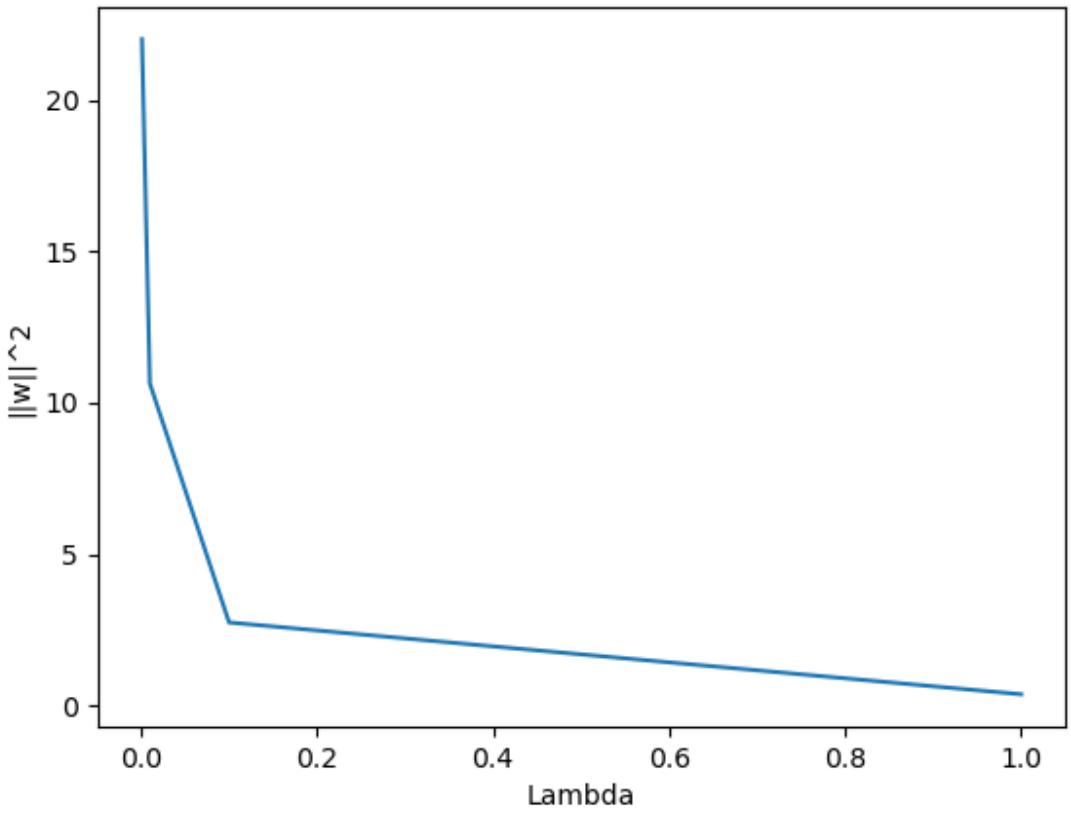
Task 4c)



Task 4d)

When applying regularization to our model, we essentially constrict the gradients of every training step from updating the weights to the optimal value in regard to the cross entropy loss. Thus, the performance is degraded. However, since regularization improves generalization, our regularized model would likely have an improved validation accuracy on an entirely new data set. Also, when having more complex models, with more layers, generalization would be even more useful.

Task 4e)



As is apparent in the figure above, higher values of lambda affect the length of our model weight vector. Again, as explained in 4b), this is likely because high values of lambda penalize complex models with large weights. The model is made significantly simpler and more generalized when the L2-norm of the weight vector is small.

In [ ]: