Assignment 1 Report

Task 1

task 1a)

1a

y^n - label
$\hat{y}$ - result

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

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

$$\hat{y} = f(x) = \frac{1}{1+e^{-w^T x}} = \frac{1}{1+e^{-z}} = \delta(z) \quad , z = w^T x$$

$$\frac{\partial C}{\partial w} = \frac{\partial C}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \delta} \cdot \frac{\partial \delta}{\partial w}$$

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

$$\frac{\partial \hat{y}}{\partial \delta} = \frac{e^{-z}}{(e^{-z}+1)^2} = \frac{1}{e^{-z}+1} \cdot \frac{e^{-z}}{e^{-z}+1} = \frac{1}{1+e^{-z}}\left(\frac{e^{-z}+1}{e^{-z}+1} - \frac{1}{e^{-z}+1}\right) = \hat{y}(1-\hat{y})$$

$$\frac{\partial \delta}{\partial w} = x$$

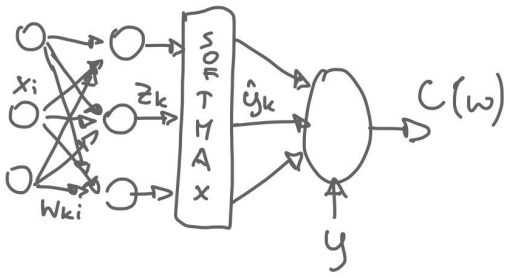
$$\Rightarrow \frac{\partial C}{\partial w} = -\frac{\hat{y}-y}{(1-\hat{y})\hat{y}} \cdot \hat{y}(1-\hat{y})\, x = (\hat{y}-y)\, x = -(y^n - \hat{y}^n)\, x_i$$

task 1b)

1b.

$$\hat{y}_k = \frac{e^{z_k}}{\sum_{k'}^{K} e^{z_{k'}}}, \quad z_k = w_k^T x = \sum_i^I w_{ki} \cdot x_i$$

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



$$\frac{\partial C^n}{\partial w_{ki}} = \frac{\partial C^n}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z_k} \frac{\partial z_k}{\partial w_{kj}}$$

Derivatives of $\hat{y}_k$ has to cases when $k=k'$ and $k \neq k'$:

case1. $$\frac{\partial \hat{y}_k}{\partial z_k} = \frac{e^{z_k} \sum_{k'}^{K} e^{z_{k'}} - e^{z_k} e^{z_k}}{\left(\sum_{k'}^{K} e^{z_{k'}}\right)^2} = \hat{y}_k (1 - \hat{y}_k)$$

case2. $$\frac{\partial \hat{y}_k}{\partial z_{k'}} = \frac{0 - e^{z_k} e^{z_{k'}}}{\left(\sum_{k'}^{K} e^{z_{k'}}\right)^2} = -\hat{y}_k \hat{y}_{k'}$$

$$\frac{\partial C}{\partial \hat{y}_k} = -\sum_{k=1}^{K} \frac{y_k}{\hat{y}_k}$$

$$\frac{\partial C}{\partial z_k} = \frac{\partial C}{\partial \hat{y}_k} \cdot \frac{\partial \hat{y}_k}{\partial z_k} = \overbrace{+\sum_{k \neq k'}^{K} \frac{y_k}{\hat{y}_k} \hat{y}_k \hat{y}_{k'}}^{k' \neq k} \overbrace{- \frac{y_{k'}}{\hat{y}_{k'}} \cdot \hat{y}_{k'}(1 - \hat{y}_{k'})}^{k' = k}$$

$$= \sum_{k \neq k'}^{K} y_k \hat{y}_{k'} - y_{k'} + y_{k'} \hat{y}_{k'}$$

$$= \sum_{k}^{K} y_k \hat{y}_{k'} - y_{k'} = \hat{y}_{k'} \underbrace{\sum_{k}^{K} y_k}_{=1} - y_{k'} = \hat{y}_{k'} - y_{k'} \overset{*}{=} \hat{y}_k - y_k$$

* I think I may have indexed it other way around

$$z_k = \sum_i^I w_{ki} \cdot x_i \xrightarrow{i=j} \frac{\partial z_n}{\partial w_{kj}} = x_j$$

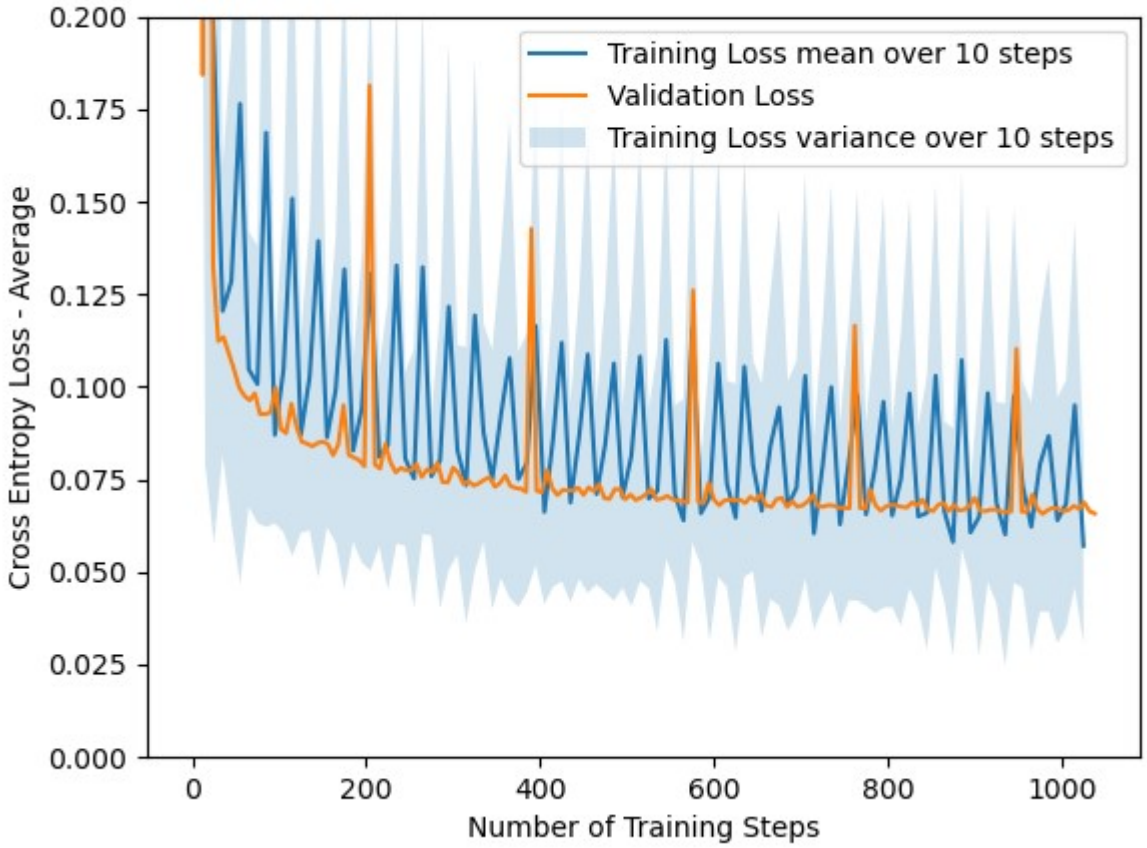
$$\Rightarrow \frac{\partial C^n(w)}{\partial w_{ki}} = \frac{\partial C^n}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z_k} \frac{\partial z_k}{\partial w_{ki}} = x_i (\hat{y}_k - y_k) = -x_i (y_k - \hat{y}_k)$$

Task 2

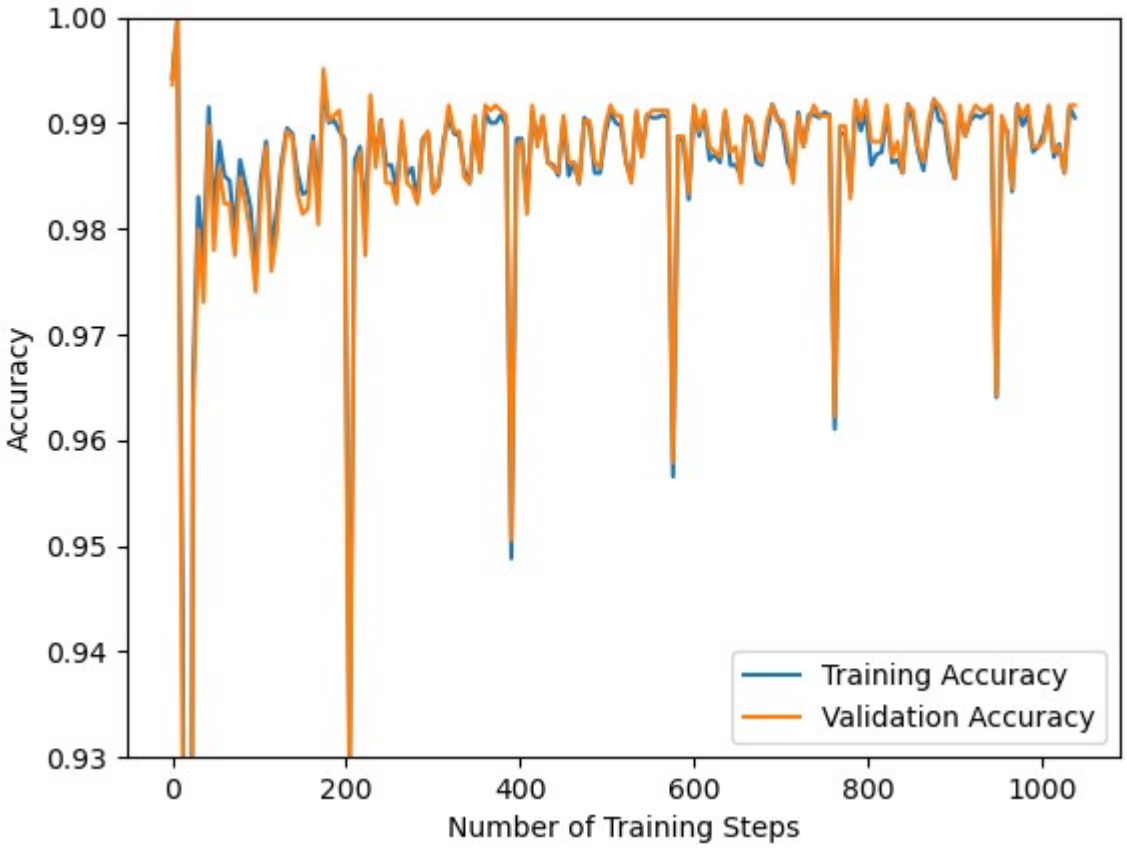
Task 2a)

For naormalizing the images i chose a linear function, that converts numbers from [0,255] -> [-1,1]. Now it is also quite easy to apply a sigmaoid function if it can give better result.

Task 2b)



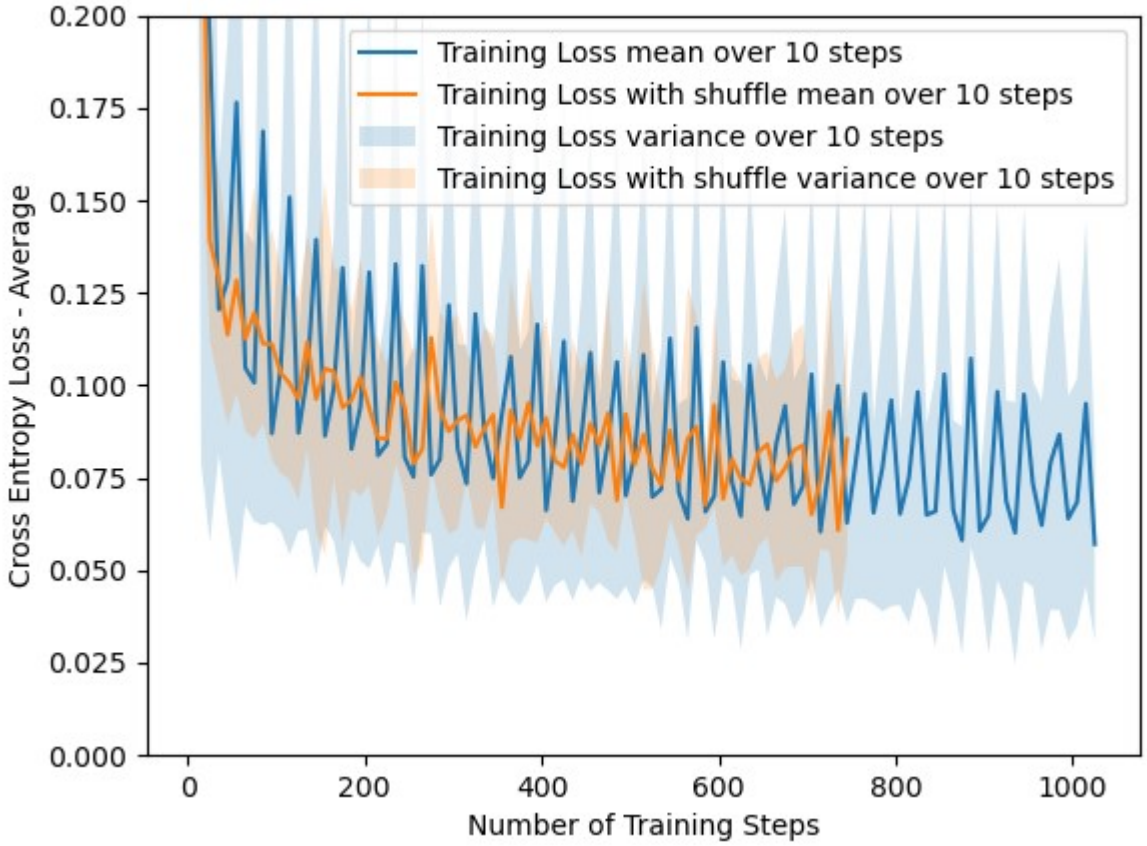
Task 2c)

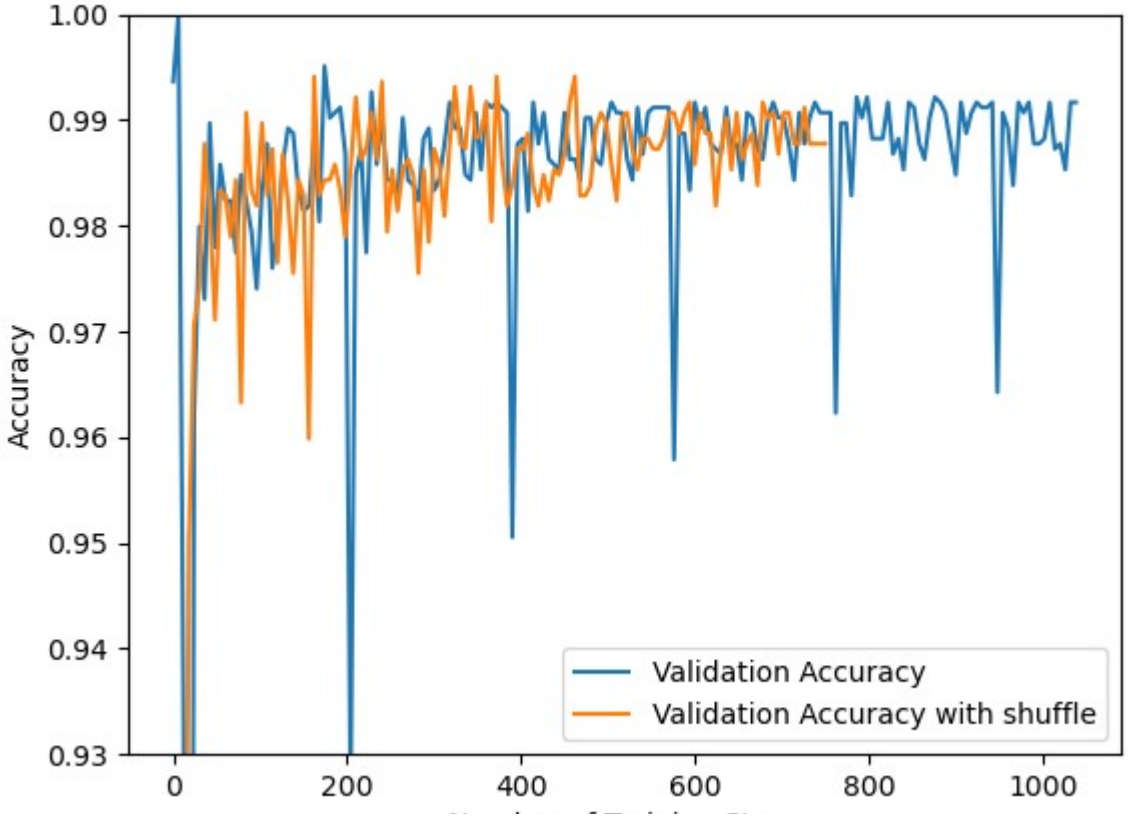


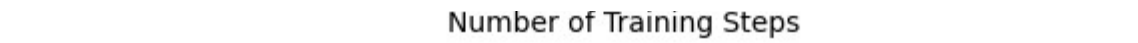
Task 2d)

The trainings stops after 33 epochs. The end accuracy results are also better then previously.

Task 2e)


Training Loss mean over 10 steps
Training Loss with shuffle mean over 10 steps
Training Loss variance over 10 steps
Training Loss with shuffle 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

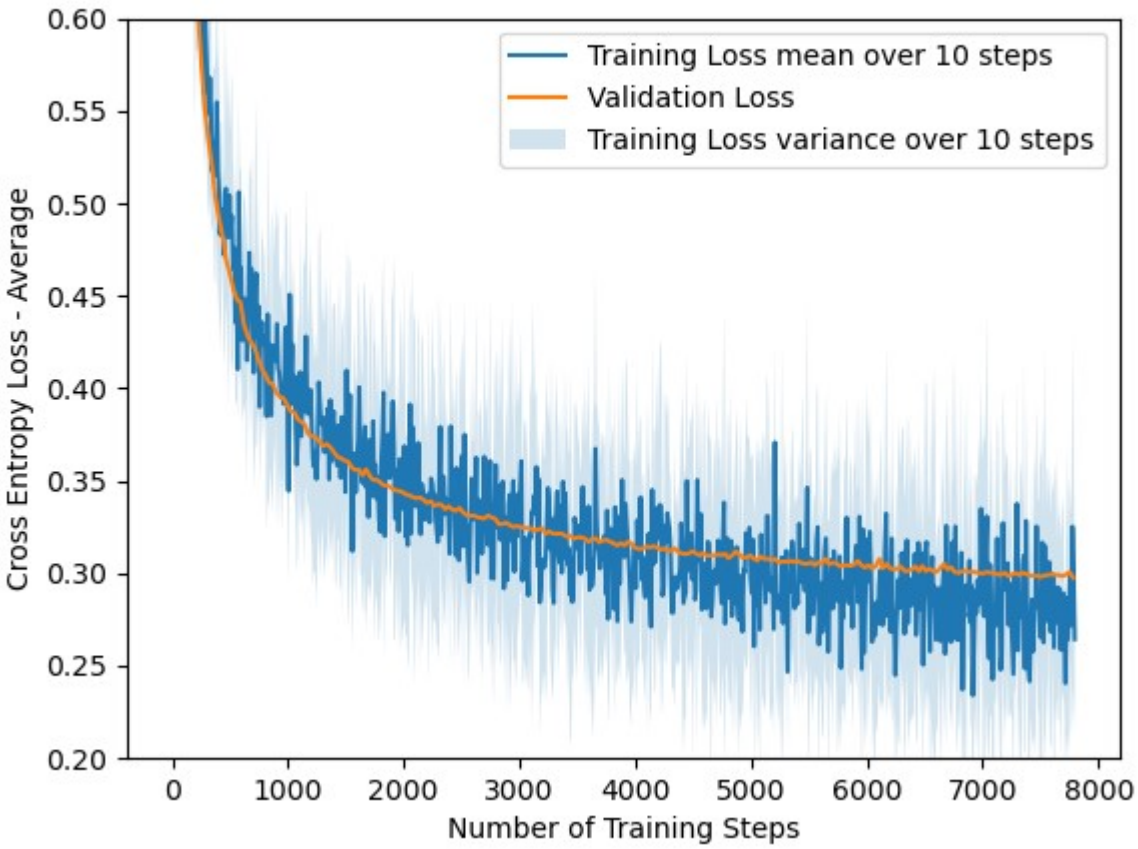

Validation Accuracy
Validation Accuracy with shuffle
Accuracy
1.00
0.99
0.98
0.97
0.96
0.95
0.94
0.93
0
200
400
600
800
1000



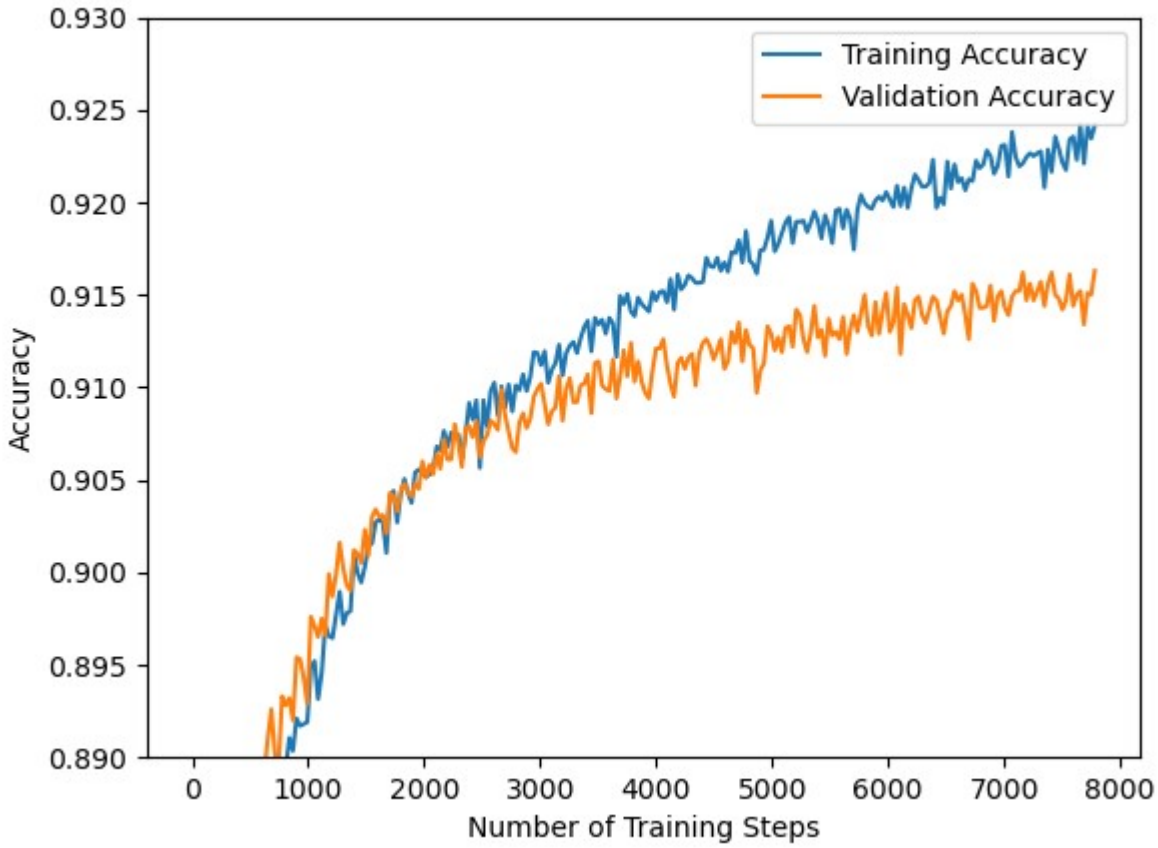
We can see that data shuffling removes many of the spikes that we could see prevoiusly. They were result of some difficult batch. When wwe shuffel the data the "difficult" images are spread around and the spikes disapear. We can also see that early stopping kicks in a bit earlier (after 24 batches). It will however vary from time to time since the data is shuffled randomly.

Task 3

Task 3b)



Task 3c)



Task 3d)

The first sign of overfitting is that the training loss is not improving very much afterabot 3000 samples. We can also see that the validation accuracy is getting quite lower than the training accuracy (also after about 3000 samples). It means that the network has learned the traingin set good, but is worsa at recognising new data. Which is overfitting.

Task 4

Task 4a)

$$J(w) = C(w) + \lambda R(w) \qquad R(w) = \|w\|^2 = \sum_{ij} w_{ij}^2$$

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

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

$$= -x_j^n (y_k^n - \hat{y}_k^n) + 2\lambda w$$

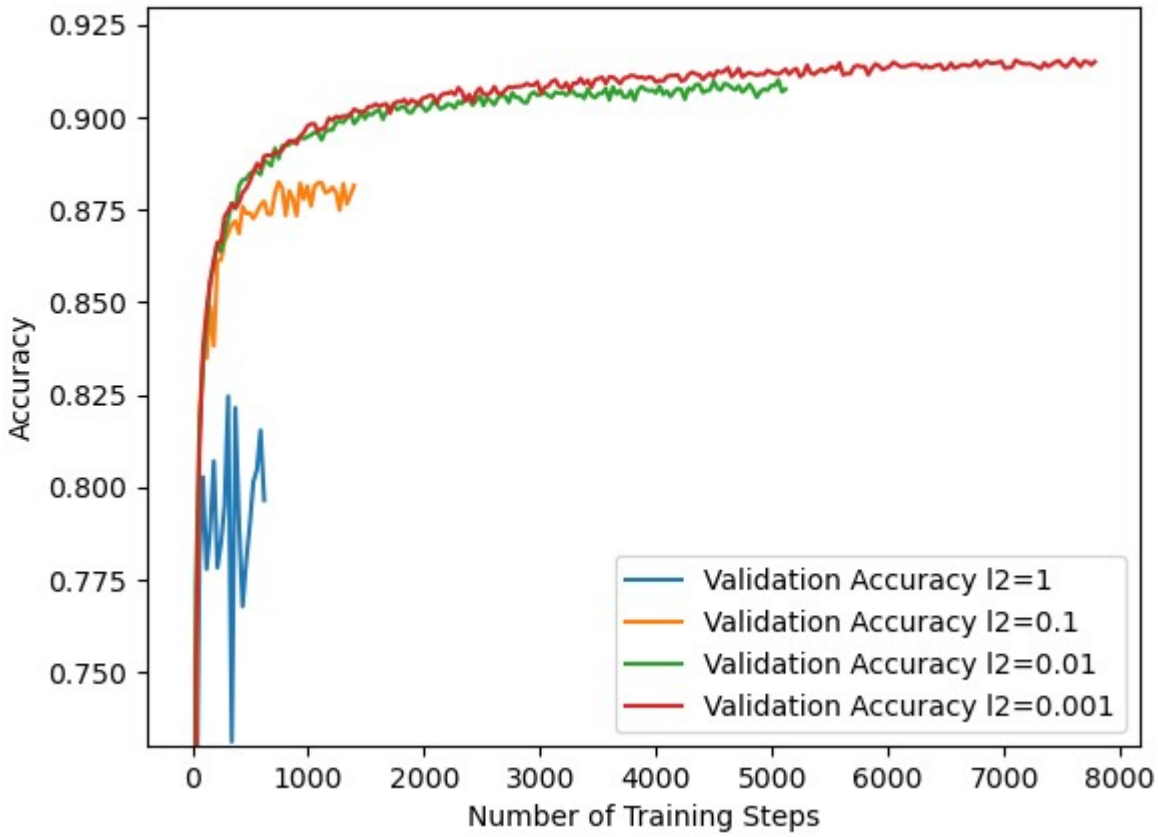
Task 4b)

lambda = 0.0

lambda = 1.0

With lambda=1 the gradiant will be smaller and weights wil also change slower and be smaller. Smaller weights leads to less complex models that doesn't overfit that much. We can often see that overfitting model will take the noise into account and becomes less general. The L2 regression makes makes model more general and we can see that clearly from the images of the weights. The weight without regularization are much morenoisy then those with.

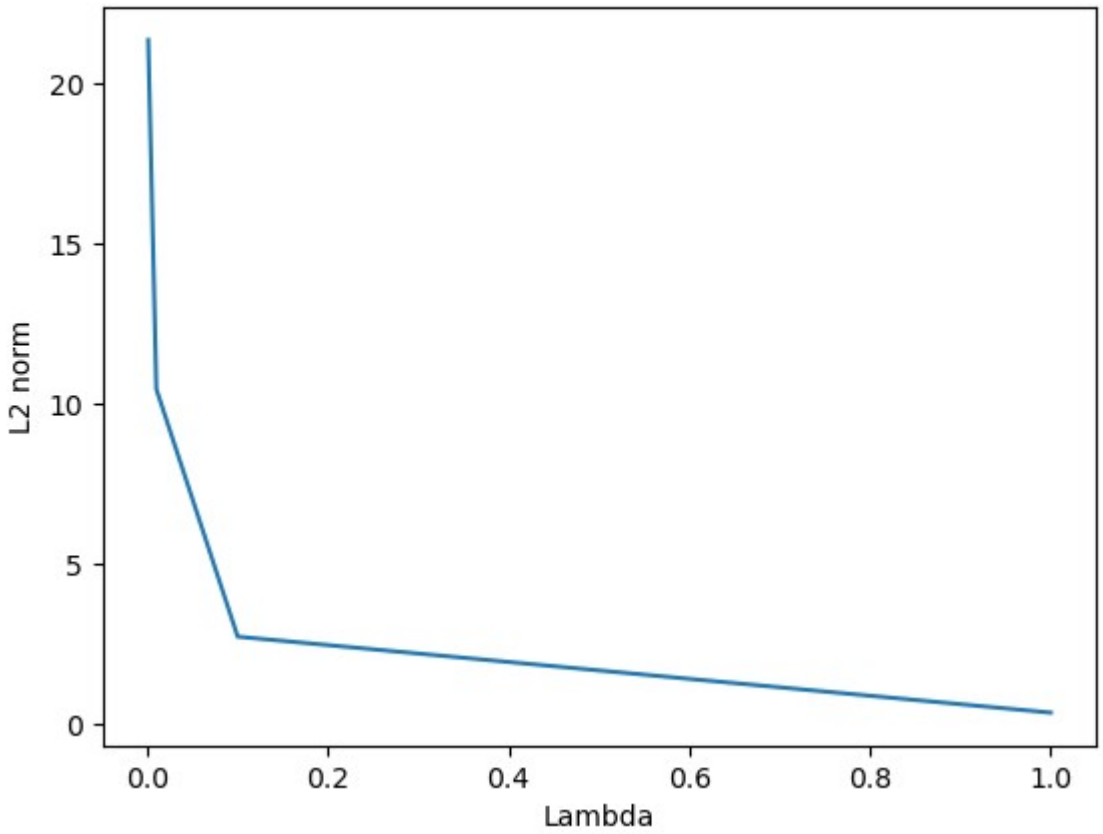
Task 4c)



Task 4d)

With regularization the model might become too "simple" for the task and wont be able respond well for more varying inputs. This may not be a problem when working with more complex problems.

Task 4e)



We can see that the higher the lambda is, the smaller the weights get. The smaller weights will lead to simpler model and less overfitting. This confirms what we see both in task b and c.