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)

1a)

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

As $\hat{y}^n = f(x^n)$, $\frac{\partial \hat{y}^n}{\partial f(x^n)} = 1$

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

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

$$= \frac{1-y^n}{1-\hat{y}^n} - \frac{y^n}{\hat{y}^n}$$

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

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

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

We know $\frac{\partial f(x^n)}{\partial w}$ from the question, Hence,

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

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

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

$$= -(y^n - \hat{y}^n)\, x_i^n \quad \text{(Shown)}$$

task 1b)

(b) $C^n(w) = -\sum_{k=1}^{K} y_k^n \ln(\hat{y}_k^n)$ $\quad \hat{y}_k^n = \frac{e^{z_k}}{\sum_{k'}^{K} e^{z_{k'}}}$ $\quad z_k = \sum_{i}^{I} w_{k,i} \cdot x_i$

$$\frac{\partial C^n(w)}{\partial w_{kj}} = \frac{\partial C^n(w)}{\partial \hat{y}_k^n} \times \frac{\partial \hat{y}_k^n}{\partial z_k} \times \frac{\partial z_k}{\partial w_{kj}}$$

$\frac{\partial z_{k'}}{\partial w_{k'j}} = x_j$ as all other w in matrix treated as const.

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

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

When $k = k'$,

$$\frac{\partial \hat{y}_k^n}{\partial z_{k'}} = \frac{\left(\sum e^{z_{k'}}\right) e^{z_{k'}} - e^{z_{k'}} \cdot e^{z_{k'}}}{\left(\sum e^{z_{k'}}\right)^2}$$

$$= \frac{e^{z_{k'}}}{\sum_j e^{z_{k'}}} - \left(\frac{e^{z_{k'}}}{\sum_i c^{z_{k'}}}\right)^2$$

$$= \hat{y}^n_{k'} - \left(\hat{y}^n_{k'}\right)^2 = \hat{y}^n_{k'}\left(1 - \hat{y}^n_{k'}\right)$$

when $k \neq k'$

$$\frac{\partial \hat{y}^n_k}{\partial z_{k'}} = \frac{\sum e^{z_{k'}} \times 0 - e^{z_k} \cdot e^{z_{k'}}}{\left[\sum_{k}^{K} e^{z_{k'}}\right]^2}$$

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

$$= -\hat{y}^n_k \times \hat{y}^n_{k'}$$

$$\frac{\partial C^n(w)}{\partial \hat{y}^n_k} = -\sum_{k'=1}^{K} \frac{\partial}{\partial \hat{y}^n_{k'}} \left(y^n_{k'}\right) \ln\left(\hat{y}^n_{k'}\right)$$

$$= -\sum_{k'=1}^{K} \frac{y^n_{k'}}{\hat{y}^n_{k'}}$$

$$\frac{\partial C^n(w)}{\partial w_{kj}} = -\sum_{k'=1}^{K} \frac{y^n_{k'}}{\hat{y}^n_{k'}} \times \frac{\partial \hat{y}^n_k}{\partial z_{k'}} \times x_j$$

$$= -x_j \sum_{k'=1}^{K} \frac{y^n_{k'}}{\hat{y}^n_{k'}} \times \frac{\partial \hat{y}^n_k}{\partial z_{k'}} \quad \Big\}$$

Only 1 instance when $k = k'$, rest $k \neq k'$

$$= -x_j \left[\frac{y^n_{k'}}{\hat{y}^n_{k'}} \times \hat{y}^n_{k'}\left(1 - \hat{y}^n_{k'}\right) + \sum_{k'=1,\, k' \neq k}^{K} \frac{y^n_{k'}}{\hat{y}^n_{k'}} \left(-\hat{y}^n_k \times \hat{y}^n_{k'}\right) \right]$$

$$= -x_j\left[\, y_{k'}^n\left(1-\hat{y}_{k'}^n\right) + \sum_{k'=1,\, k'\neq k}^{K}\left(-\, y_{k'}^n\, \hat{y}_k^n\right)\right]$$

$$= x_j\left[\, y_{k'}^n\, \hat{y}_{k'}^n - y_{k'}^n + \sum_{k'=1,\, k'\neq k}^{K} \hat{y}_k^n \times y_{k'}^n \right]$$

$$= x_j\left[-y_{k'}^n + y_{k'}^n\, \hat{y}_k^n + \sum_{k'=1,\, k'\neq k}^{K} \hat{y}_k^n \times y_{k'}^n \right],$$ as in $y_{k'}^n\, \hat{y}_{k'}^n$, $k=k'$, $\therefore \hat{y}_{k'}^n = \hat{y}_k^n$,

$$= x_j\left[-y_{k'}^n + \hat{y}_k^n\left[\, y_{k'}^n + \sum_{k'=1,\, k'\neq k}^{K} y_{k'}^n \right]\right]$$

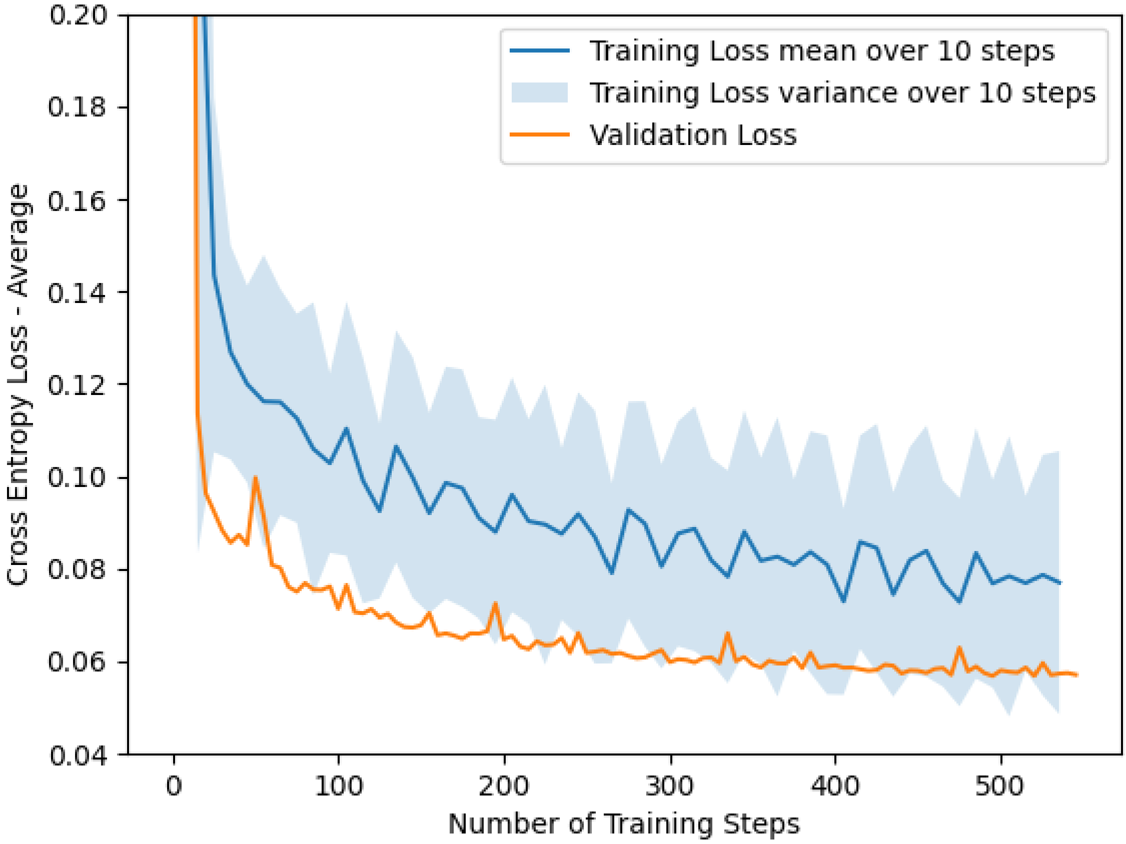
$$= x_j\left[-y_{k'}^n + \hat{y}_k^n\left(\sum_{k'=1}^{K} y_{k'}^n\right)\right]$$

$$= -x_j\left[\, y_k^n - \hat{y}_k^n\,(1)\right]$$ as $k=k'$ in that instance and $\sum_{k'=1}^{K} y_{k'}^n = 1$

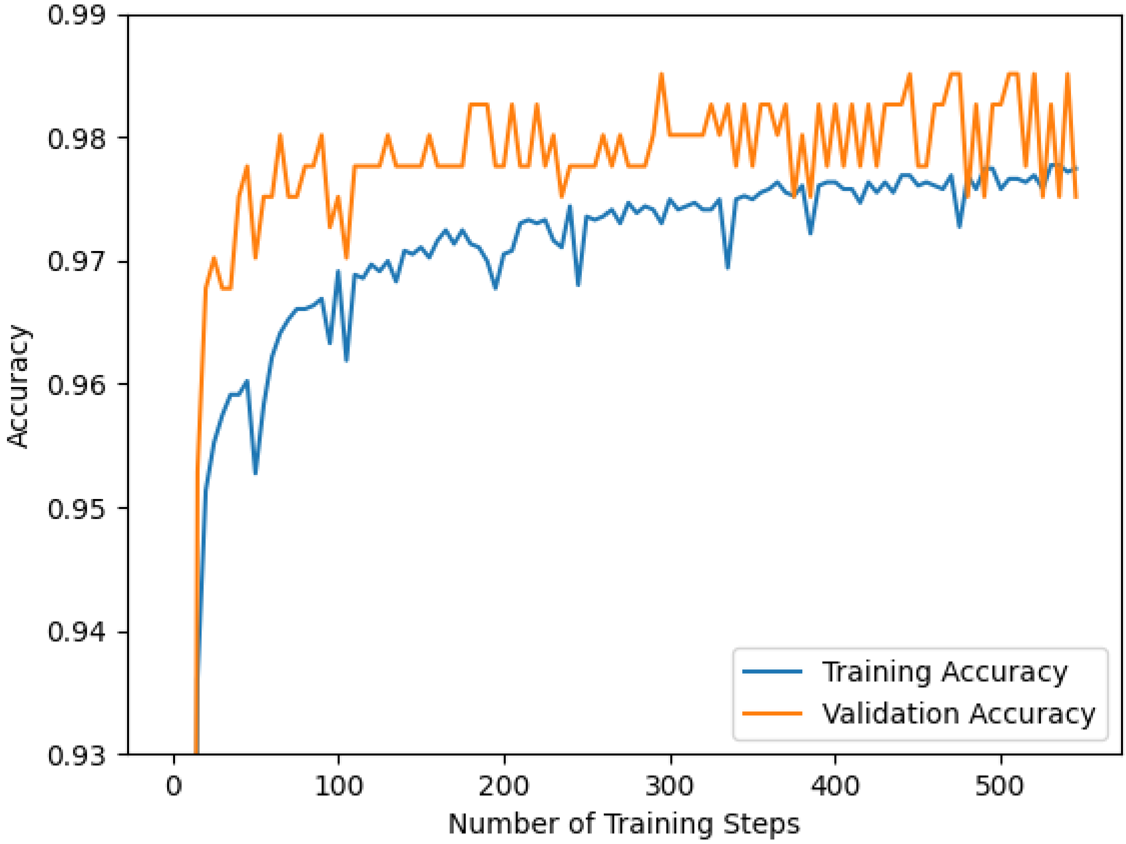
$$= -x_j\left[\, y_k^n - \hat{y}_k^n\right]$$ (Shown)

Task 2

Task 2b)


Training Loss mean over 10 steps
Training Loss variance over 10 steps
Validation Loss
Cross Entropy Loss - Average
0.20
0.18
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0
100
200
300
400
500
Number of Training Steps

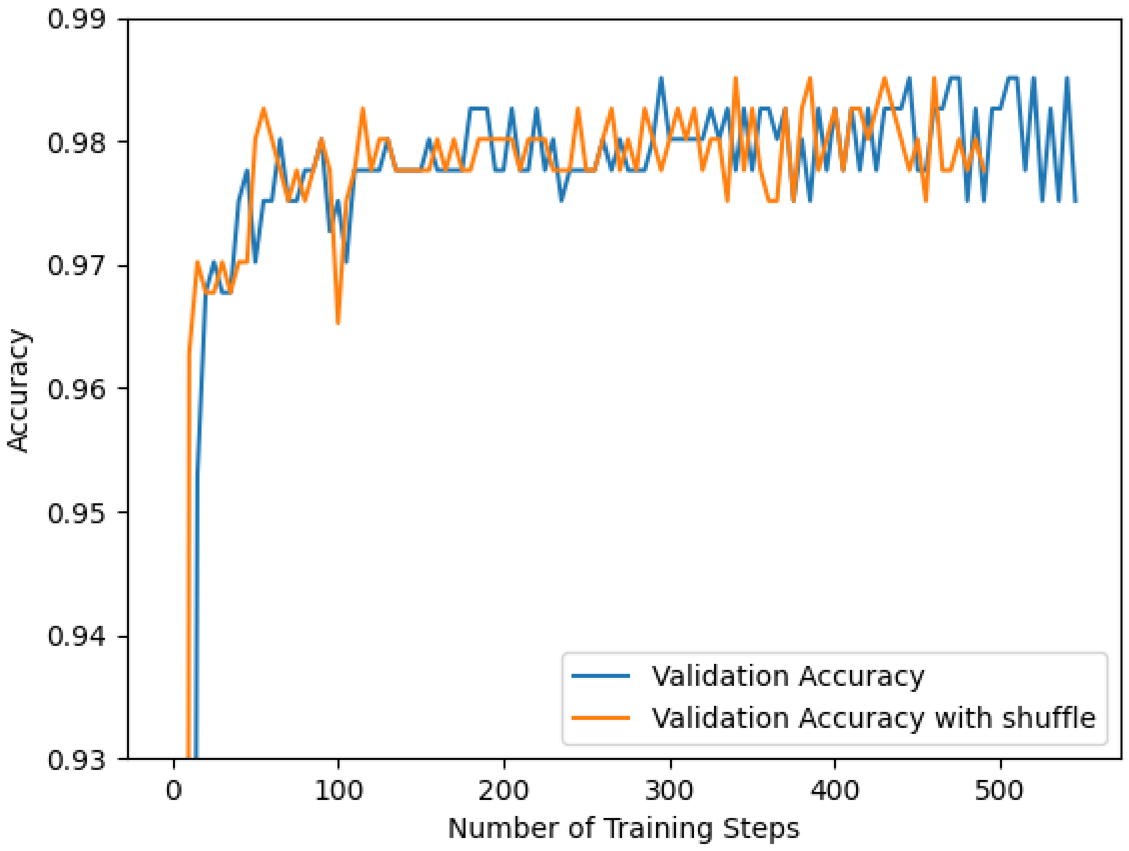
Task 2c)


Accuracy
0.99
0.98
0.97
0.96
0.95
0.94
0.93
0
100
200
300
400
500
Number of Training Steps
Training Accuracy
Validation Accuracy

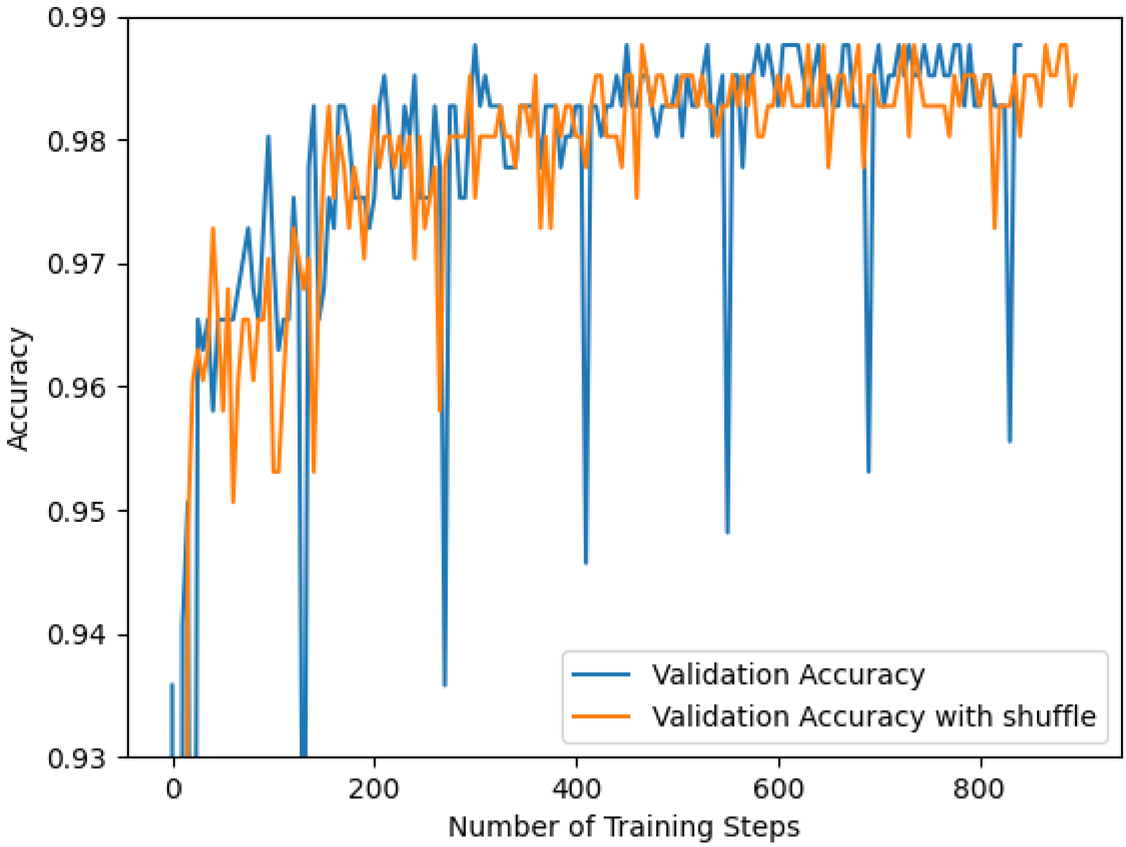
Task 2d)

It kicks in at 19 epochs.

Task 2e)



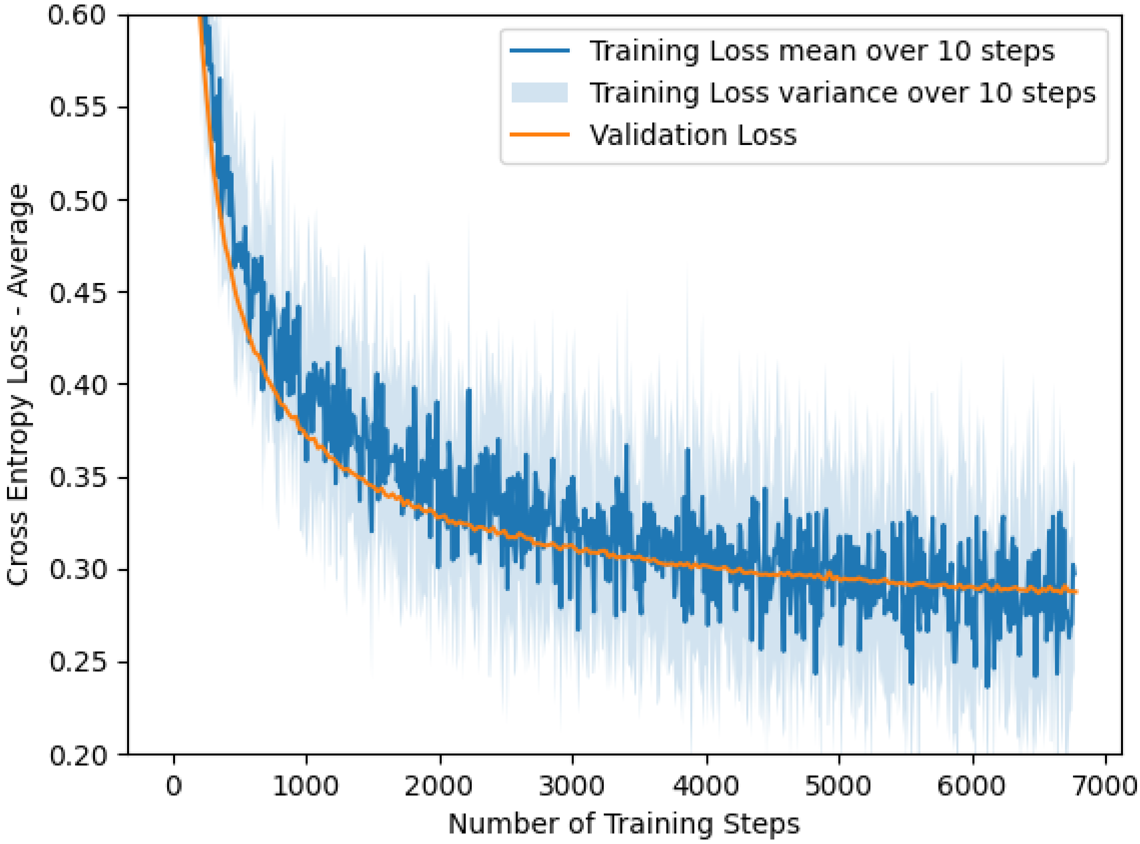
Stochastic sampling is turned on above without much spikes. Stochastic sampling is turned off to show the spikes below.



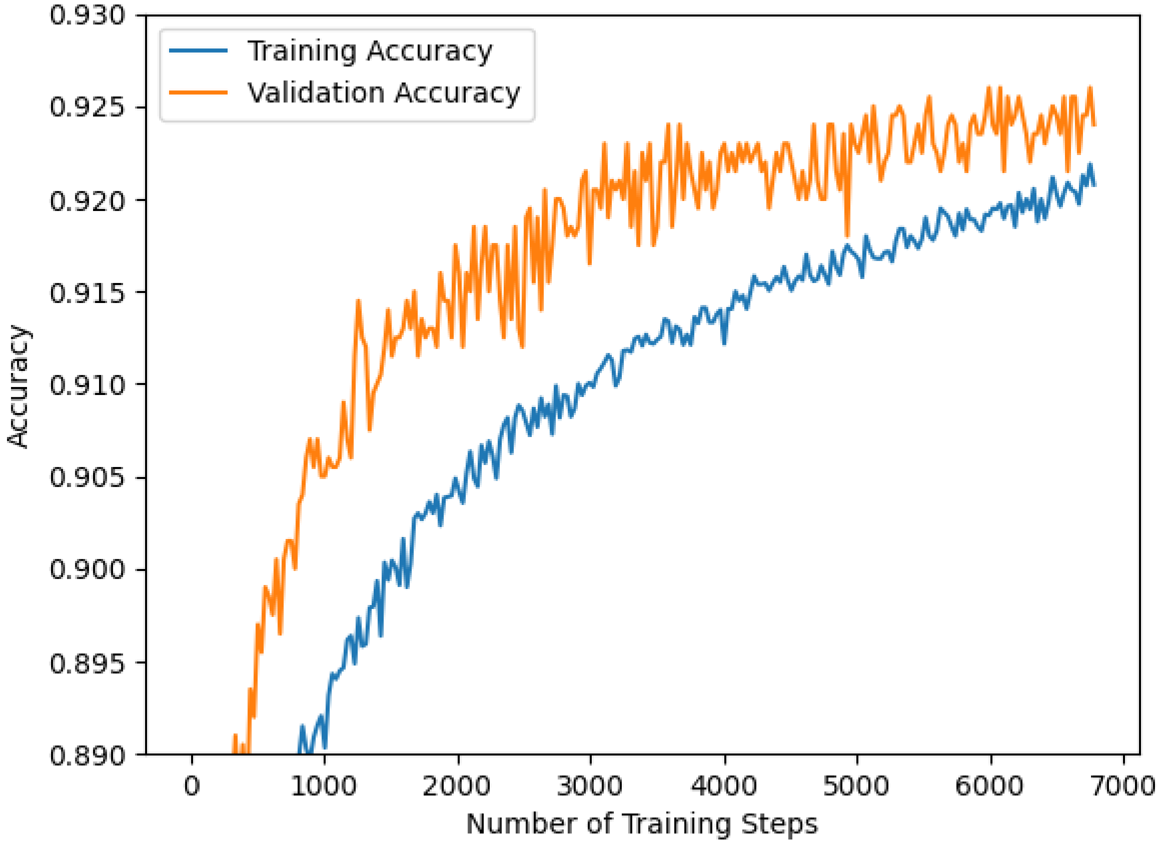
The shuffled data shows to have less spikes than the one without. Without randomizing the samples, the train set will be trained on the same data for the series of batches in each epoch. This may lead to some batches being harder than others. Hence causing spikes in the model without shuffled dataset. Shuffling the dataset reduces the chance of this occurence and hence less significant spikes.

Task 3

Task 3b)


Training Loss mean over 10 steps
Training Loss variance over 10 steps
Validation Loss
Cross Entropy Loss - Average
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0
1000
2000
3000
4000
5000
6000
7000
Number of Training Steps

Task 3c)


Training Accuracy
Validation Accuracy
Accuracy
0.930
0.925
0.920
0.915
0.910
0.905
0.900
0.895
0.890
0
1000
2000
3000
4000
5000
6000
7000
Number of Training Steps

Task 3d)

I do not observe overfitting in the model through this graph. Overfitting occurs when the training accuracy is much higher than that of the validation accuracy. Instead, the validation accuracy is higher than that of the training accuracy. This could be due the validation set having relatively simpler to identify samples than the training set. Furthermore, looking at the loss graph, we still do not see significant deviation between the two.

Task 4

Task 4a)

4a)

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

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

$$\frac{\partial R(w)}{\partial w} = \frac{\partial}{\partial w} \frac{1}{2} \sum_{i,j} w_{i,j}^2$$

Derivative will only be non-zero when $i' = i$ and $j' = j$

Hence at each matrix cell, the derivative is:

$$\frac{1}{2} \times \frac{\partial}{\partial w} w_{i,j}^2 = w_{i,j}$$

Hence $\frac{\partial R(w)}{\partial w} = w$, where w is the weight matrix.

$$\therefore \lambda \cdot \frac{\partial R(w)}{\partial w} = \lambda w \longleftarrow \text{Update due to L2}$$

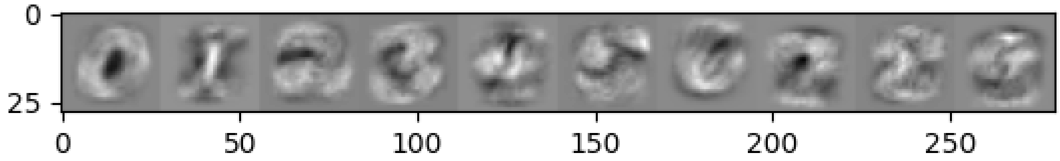
We know $C(w) = \frac{1}{N} \sum_{n=1}^{N} C^n(w)$

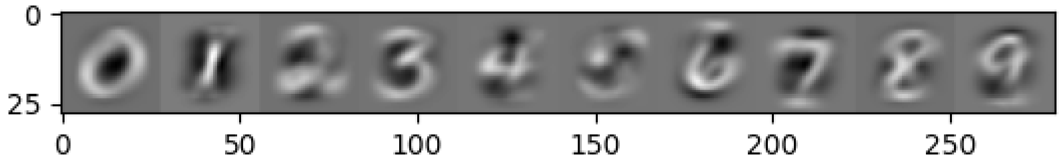
$$\therefore \frac{\partial C(w)}{\partial w} = \frac{1}{N} \sum_{n=1}^{N} \frac{\partial C^n(w)}{\partial w}$$

As earlier computed, we know $\frac{\partial C^n(w)}{\partial w}$ for softmax regression. hence we have our update term for softmax regression with L2 regularization, $\frac{\partial J(w)}{\partial w}$.

$$\frac{\partial J(w)}{\partial w} = \frac{1}{N}\sum_{n=1}^{N}\left[-x_j^n\left(y_k^n - \hat{y}_k^n\right)\right] + \lambda w$$

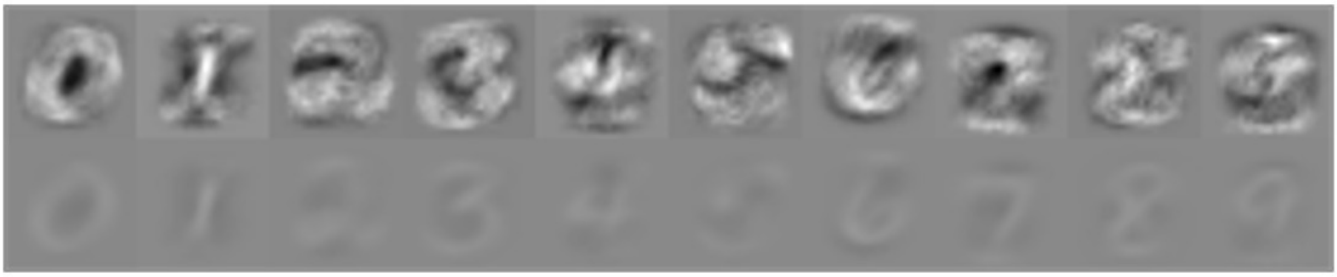
Task 4b)





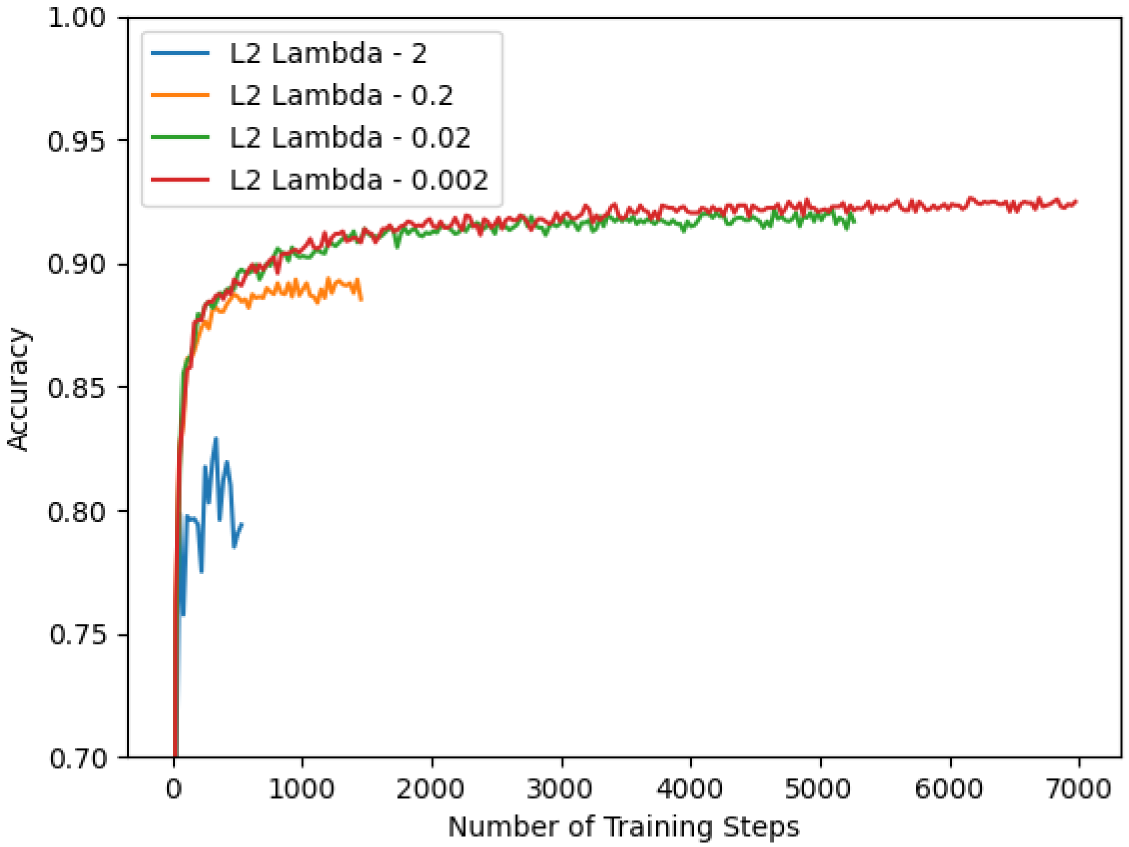
Weights for Lambda = 0.0 visualized above and the weights for Lambda = 2.0 visualized below.

L2 Regularization heavily penalizes weights > 1 and "encourages" weights towards 0. As such, it reduces reliance on one/few specific node(s) to determine the class of the object.



When plotting both weights on the same plot, we observe that the weights with Lambda=2.0 are lighter in color, indicating that they all have less extreme values as compared to the ones with Lambda=0.0. The visualized weights for lambda = 2.0 looks less noisy as the model has to rely on more general shape, therefore a smoother gradient of color as well as a darker dark region to indicate areas where they completely do not expect to see the shape such as for 0 and 1.

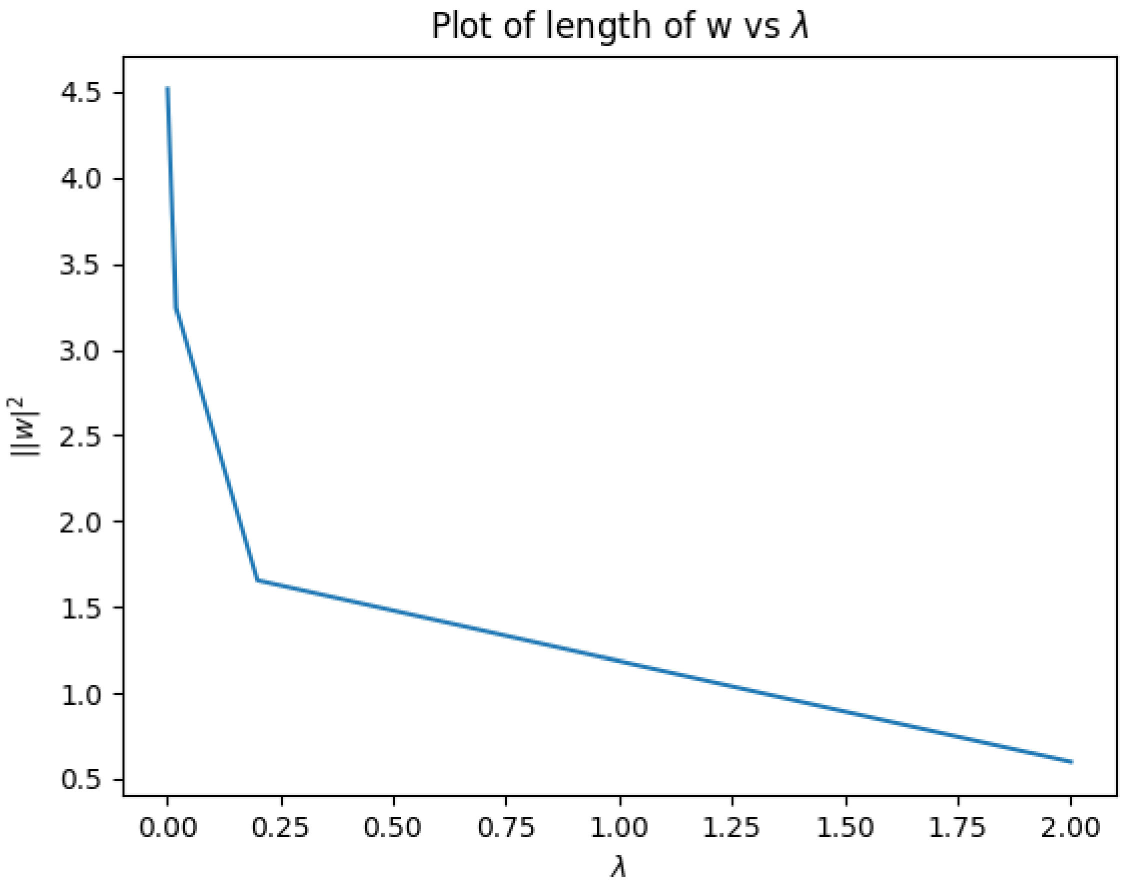
Task 4c)



Task 4d)

As regularization penalizes the model for being too complex, we will lose predictive power as we reduce the complexity of the model. The model is forced to explore more general relationships and be less reliant on specific nodes to determine classes. The model cannot as closely "memorize" the answers and hence validation accuracy will fall slightly.

Task 4e)



I observe that with a higher lambda value, the L2 norm of the weight vector decreases.