Assignment 1 Report

Task 1

task 1a)

We start by rewriting $\alpha \frac{\partial C}{\partial w_{ji}}$ using the chain rule.

$\alpha \frac{\partial C}{\partial w_{ji}} = \alpha(\frac{\partial C}{\partial z_j} \cdot \frac{\partial z_j}{\partial w_{ji}})$.

Since we know that $\delta_j = \frac{\partial C}{\partial z_j}$ and $\frac{\partial z_j}{\partial w_{ji}} = x_i$,

from equation: $w_{ji} := w_{ji} - \alpha \frac{\partial C}{\partial w_{ji}}$

that $w_{ji} := w_{ji} - \alpha \delta_j x_i$ □

Now we want to show that:

$\delta_j = f'(z_j) \sum_k w_{kj} \delta_k$

We rewrite this to: $\frac{\partial C}{\partial z_j} = \frac{\partial C}{\partial a_j} \cdot \frac{\partial a_j}{\partial z_j} = \underline{(\sum_k \frac{\partial C}{\partial z_k} \cdot \frac{\partial z_k}{\partial a_j}) \frac{\partial a_j}{\partial z_j}}$

By definition, we have that $a_j = f(z_j) \rightarrow \frac{\partial a_j}{\partial z_j} = f'(z_j)$

We know that $z_k = \sum_j w_{kj} a_j \rightarrow \frac{\partial z_k}{\partial a_j} = \sum_j w_{kj}$

From the previous task we know that $\delta_k = \frac{\partial C}{\partial z_k}$

Thus,

$\delta_j = f'(z_j) \sum_k w_{kj} \delta_k$□

task 1b)

We have the following update rules for the weights:

- Hidden layer to output layer:

  - $w_{kj} := w_{kj} - \alpha \frac{\partial C}{\partial w_{kj}}$

- Input layer to hidden layer:
  - $w_{ji} := w_{ji} - \alpha \frac{\partial C}{\partial w_{ji}}$

In order to write these update rules in matrix/vector notation we first need to define the following notations:

- $\boldsymbol{W_1} = w_{ji}$
- $\boldsymbol{W_2} = w_{kj}$
- $\boldsymbol{z_1} = \boldsymbol{W_1} \cdot \boldsymbol{x}$
- $\boldsymbol{a_1} = f(\boldsymbol{z_1})$
- $\boldsymbol{\delta_1} = \delta_j = f'(\boldsymbol{z_1}) \sum_k \boldsymbol{W_2 \delta_2}$
- $\boldsymbol{\delta_2} = \delta_k = -(\boldsymbol{y_k} - \boldsymbol{\hat{y}_k})$

We can now write the update rules using these notations:

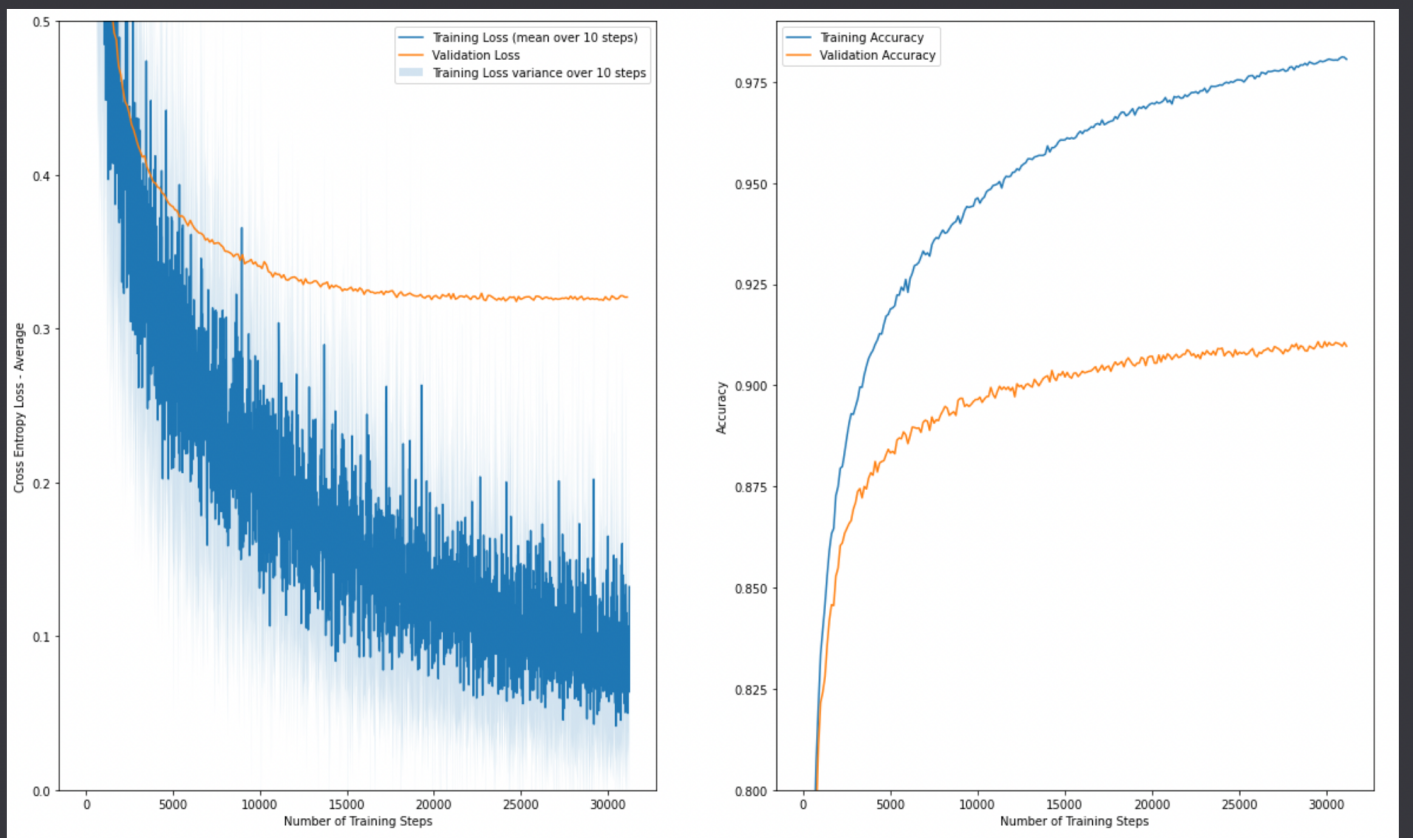
- Hidden layer to output layer:
  - $\boldsymbol{W_2} := \boldsymbol{W_2} - \alpha \boldsymbol{\delta_2}^T \cdot \boldsymbol{a_1}$

- Input layer to hidden layer:
  - $\boldsymbol{W_1} := \boldsymbol{W_1} - \alpha \boldsymbol{\delta_1}^T \cdot \boldsymbol{x}$

Task 2

Task 2c)

```
Final Train Cross Entropy Loss: 0.08429470894721514
Final Validation Cross Entropy Loss: 0.3210173711252464
Train accuracy: 0.98105
Validation accuracy: 0.9104
```



Task 2d)

Since we have a weight matrix of size (758, 64) from the first layer and a weight matrix of size (64, 10) from the hidden layer, we know that the number of parameters will be: $(785 \cdot 64) + (64 \cdot 10) = \underline{50880}$

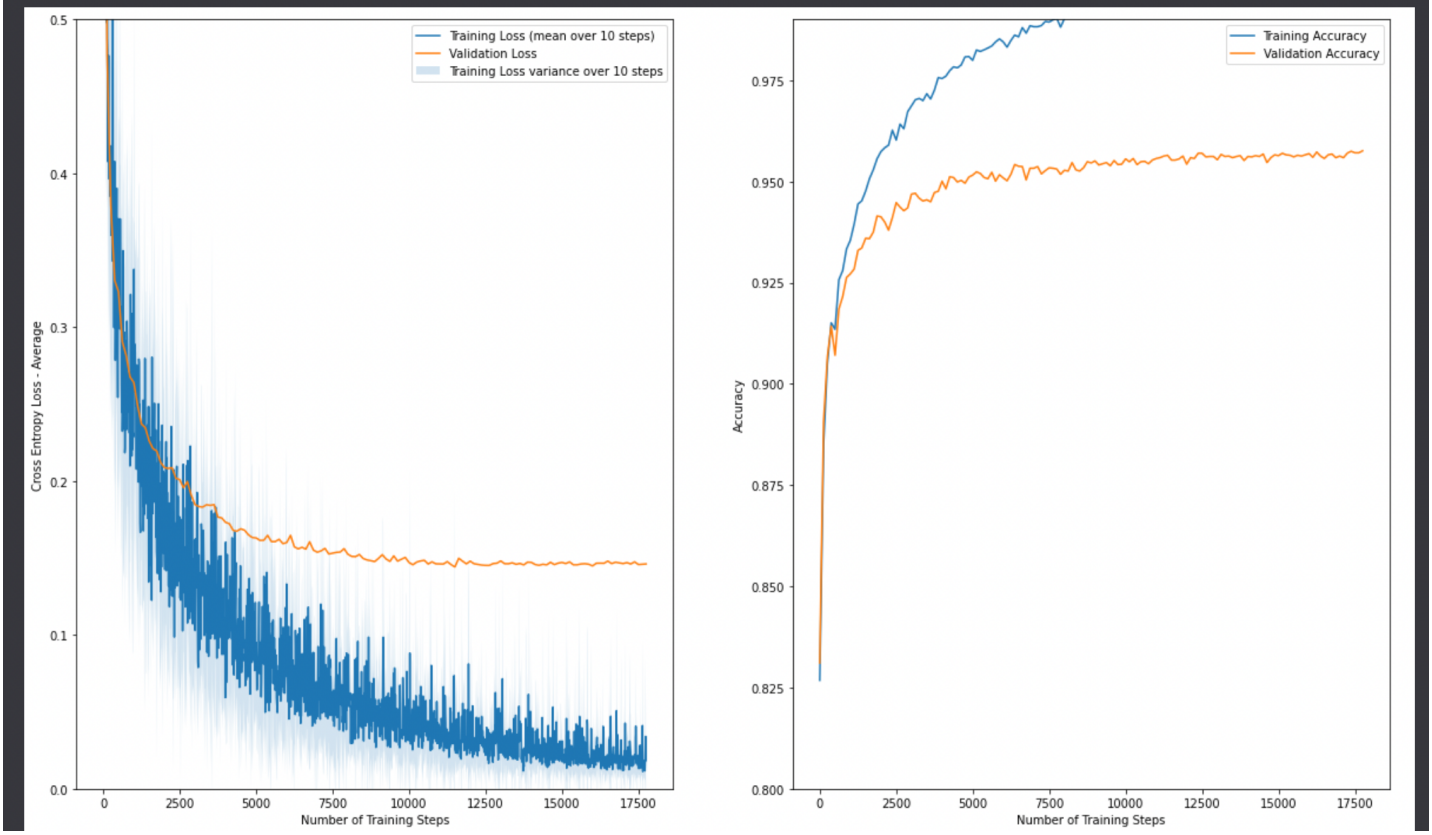
Task 3

For these tasks we have implemented the early stopping method from Assigmnent 1 to prevent overfitting.

a) Improved weight initialization

With improved weight initialization we see that the train accuracy, but especially the validation accuracy improve a lot, as we see in in the plot. The model stopped training after 28 epochs to prevent overfitting. If we disabled early stopping we would see from the plot that our model would start overfitting around 17500 training steps.

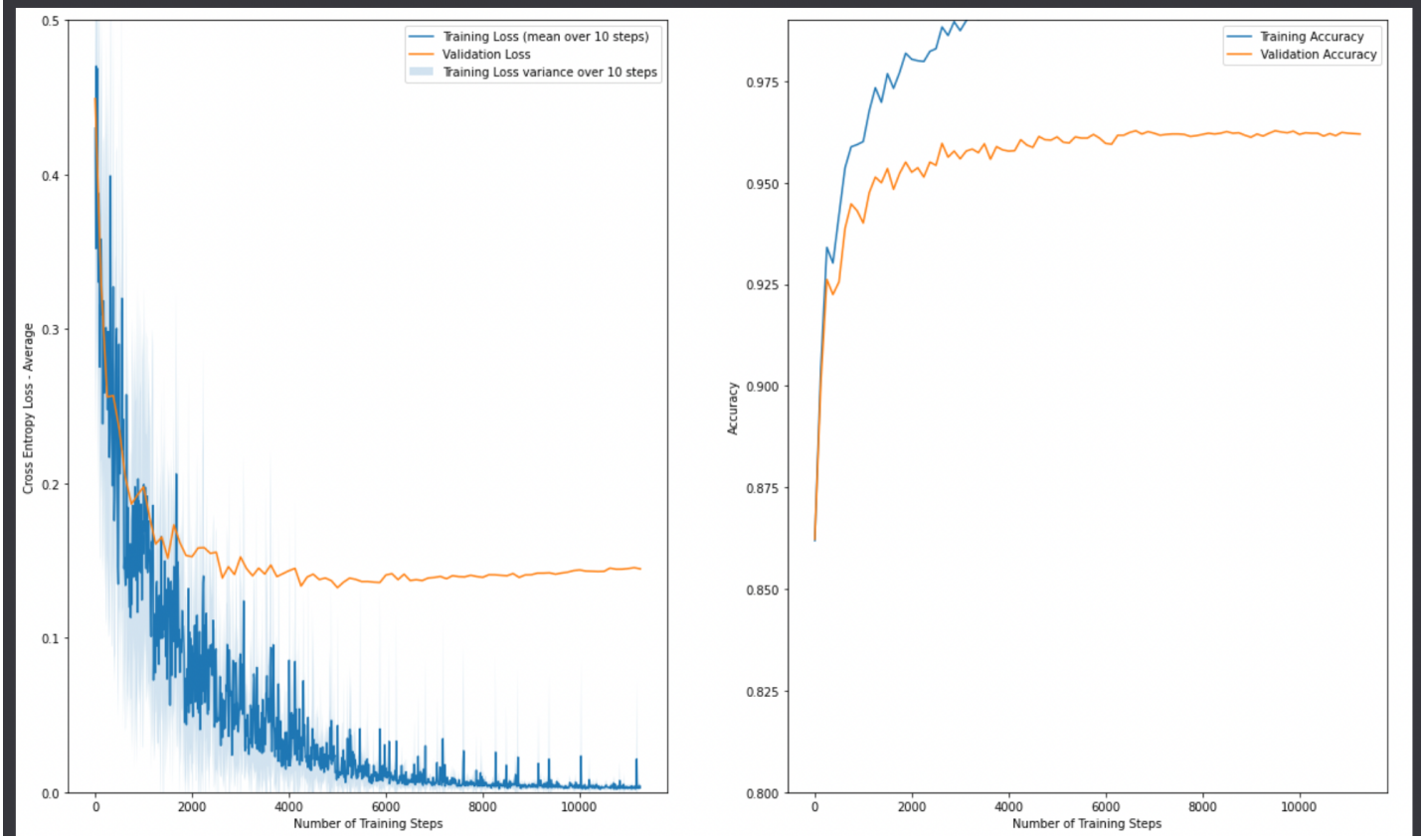
```
Early stop at  28 epochs
Final Train Cross Entropy Loss: 0.01795475840444326
Final Validation Cross Entropy Loss: 0.1461397824935992
Train accuracy: 0.9988
Validation accuracy: 0.9576
```



b) Improved sigmoid

With improved weight initialization and improved sigmoid we see that the train accuracy and validation accuracy improve a bit. The model stopped training after 18 epochs. If we disabled early stopping we would see from the plot that our model would start overfitting around 10000 training steps. This is an improvement in convergence speed from our model in task 3a.

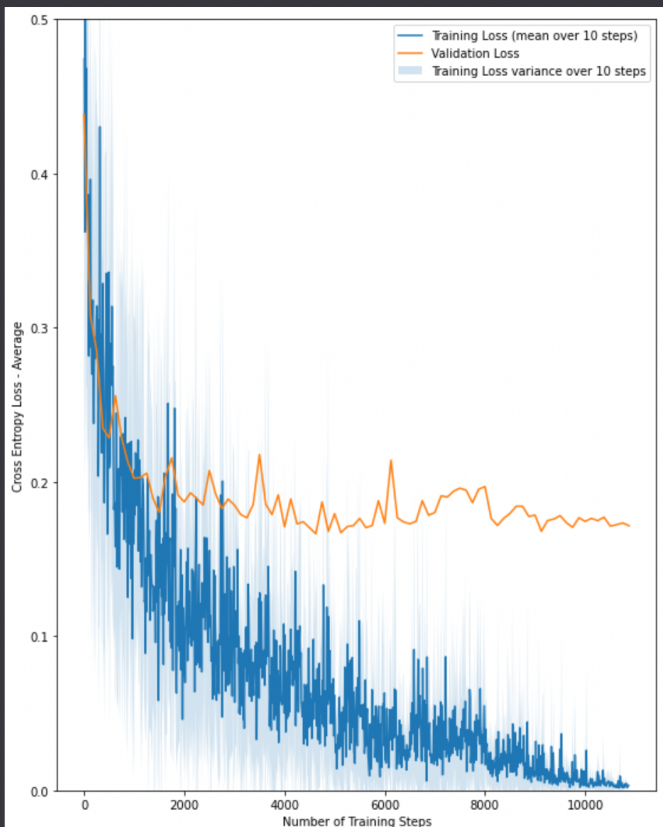
```
Early stop at  18 epochs
Final Train Cross Entropy Loss: 0.0029959357034577343
Final Validation Cross Entropy Loss: 0.14467747194142916
Train accuracy: 0.9999
Validation accuracy: 0.962
```

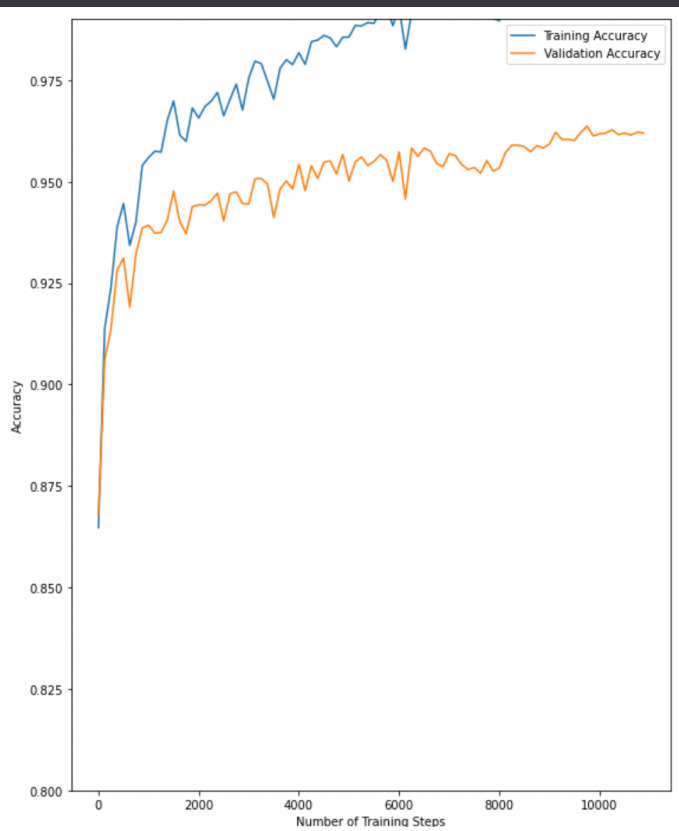


c) Momentum

With improved weight initialization, improved sigmoid and momentum we see that the train accuracy gets a very small improvement while validation remains the same. The model stopped training after 17 epochs, which is a small improvement in convergence speed from our model in task 3b. If we disabled early stopping we would see from the plot that our model would start overfitting around 10000 training steps. Even though the improvement is minimal in our model, we know that implementation of momentum is an efficient tool to improve the convergence speed, especially for networks with complex loss surfaces, although determining the right momentum gamma and learning rate is a difficult task. This is because the momentum algorithm prevents the gradient from being stuck at saddle points.

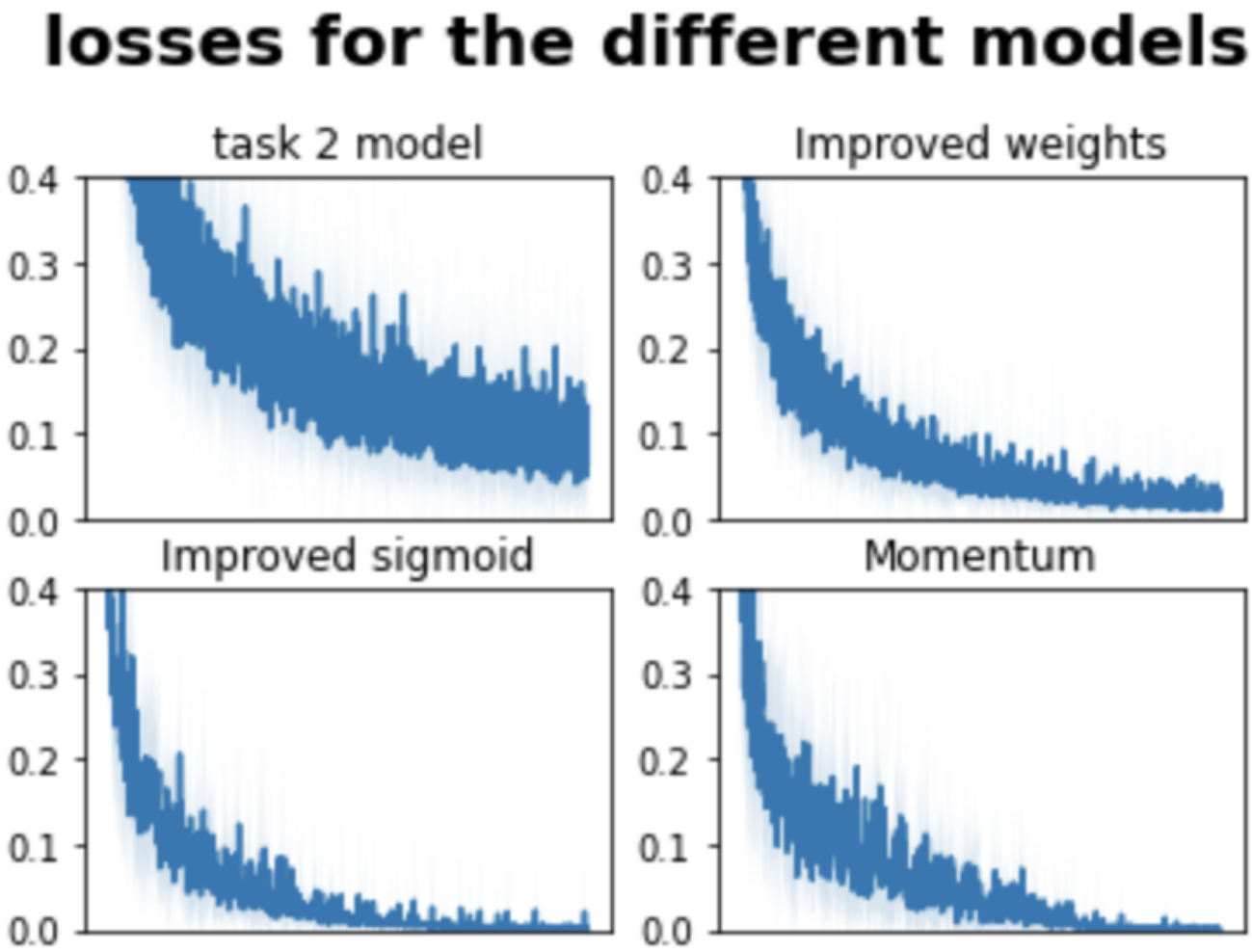
```
Early stop at  17 epochs
Final Train Cross Entropy Loss: 0.002437337889733899
Final Validation Cross Entropy Loss: 0.1716032975776262
Train accuracy: 0.99995
Validation accuracy: 0.962
```

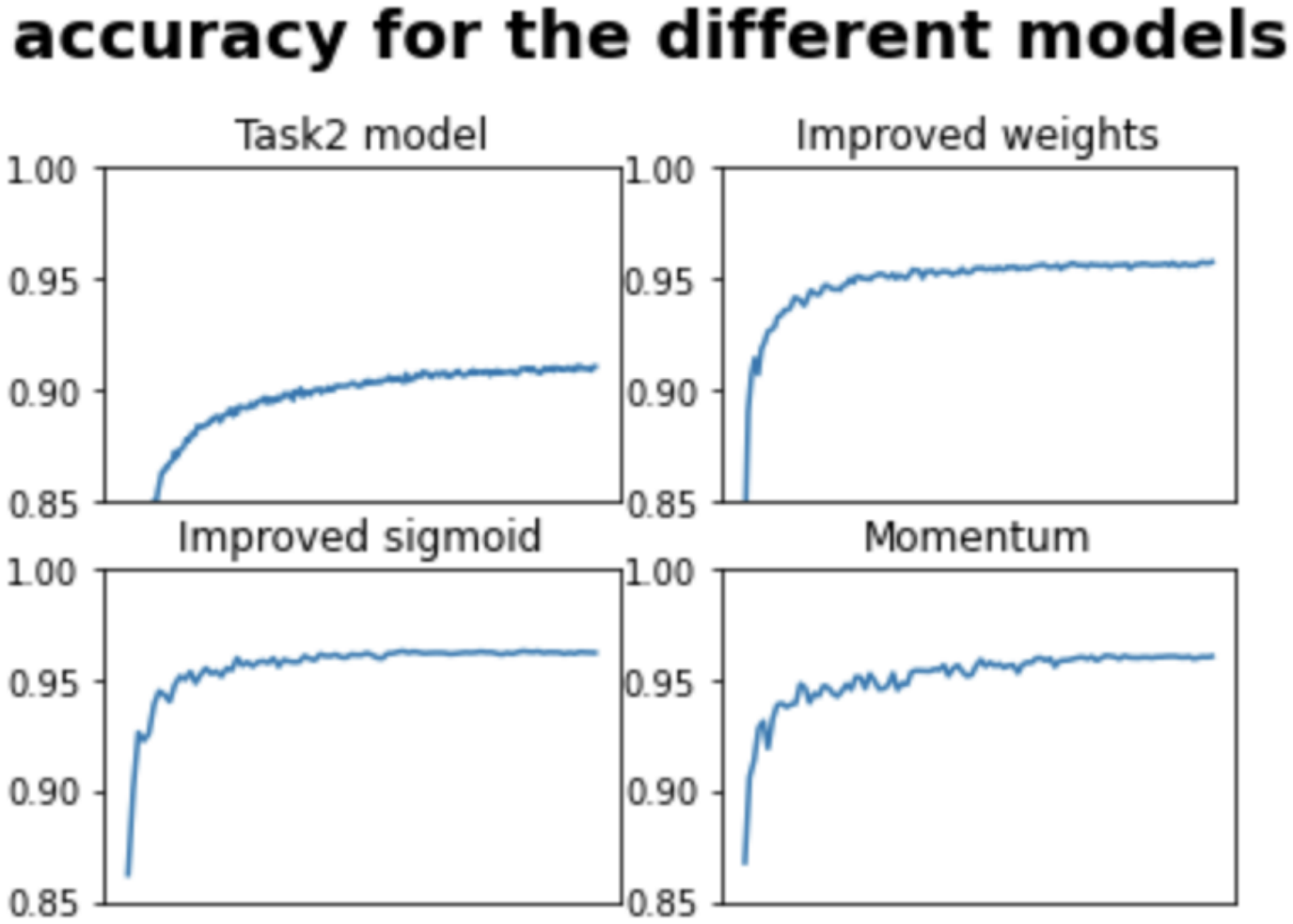




Conclution:

As we see from the plots below we see that we get an significant improvement in convergence speed, train/validation accuracy and loss when we implement these tricks to our model.


losses for the different models
task 2 model
Improved weights
Improved sigmoid
Momentum
0.4
0.3
0.2
0.1
0.0



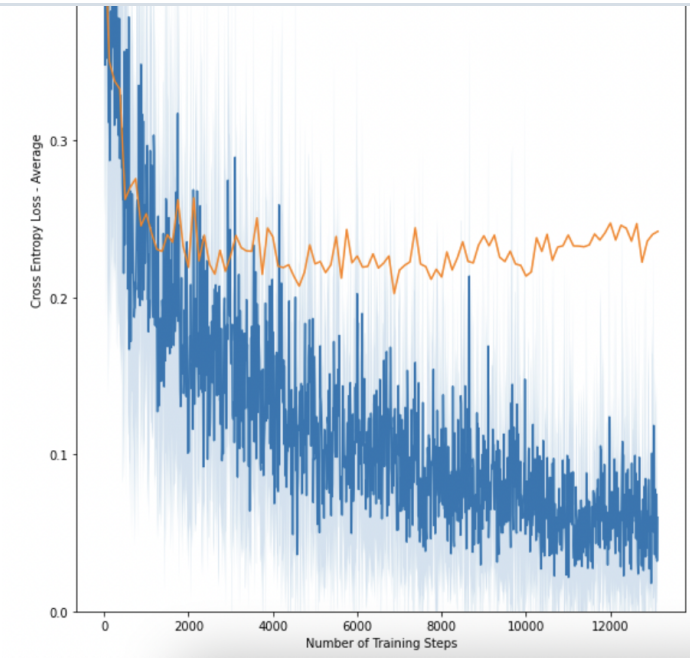
Task 4

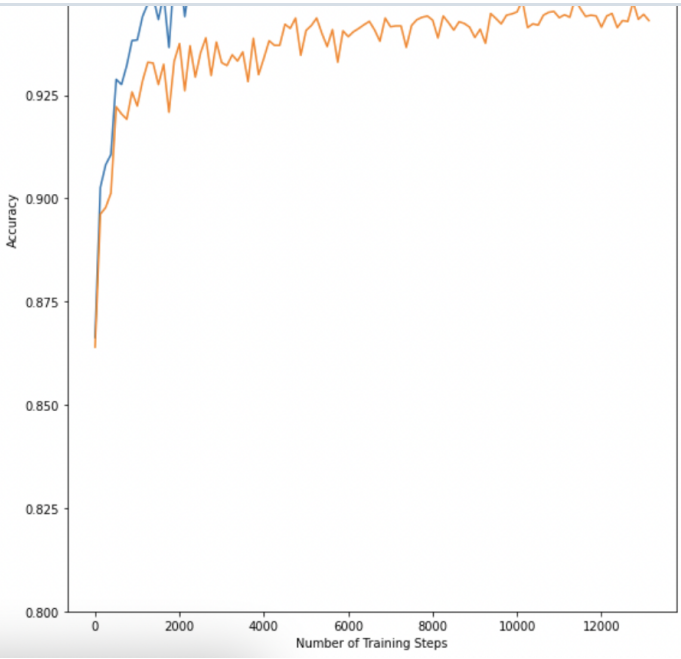
Task 4a)

When implementing 32 units in the hidden layer we see that the model achieves worse results then it does with 64 units. It has less accuracy, both for the training set and the validation set, and the cross entropy loss increases as well. With a smaller amount of hidden units we have less weights in our model which will have a negative impact on the final result after training. This is because we have a non-linear problem. With a linear problem, we can solve the it with fewer units in the hidden layer, even without a hidden layer, but when the problem increases in complexity, the general rule is to increase the number of units in the hidden layer or add more hidden layers.

```
Early stop at  21 epochs
Final Train Cross Entropy Loss: 0.04577145897721053
Final Validation Cross Entropy Loss: 0.24218276994141288
Train accuracy: 0.9861
Validation accuracy: 0.9431
```

Even Lauvsnes / **Assignments Deep learning** Published at Feb 18, 2022 Private

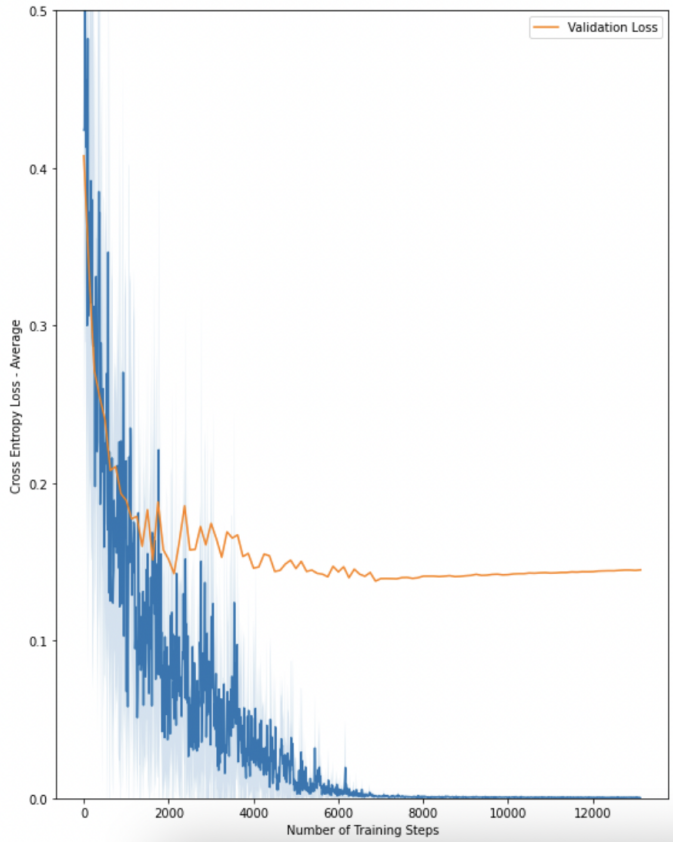


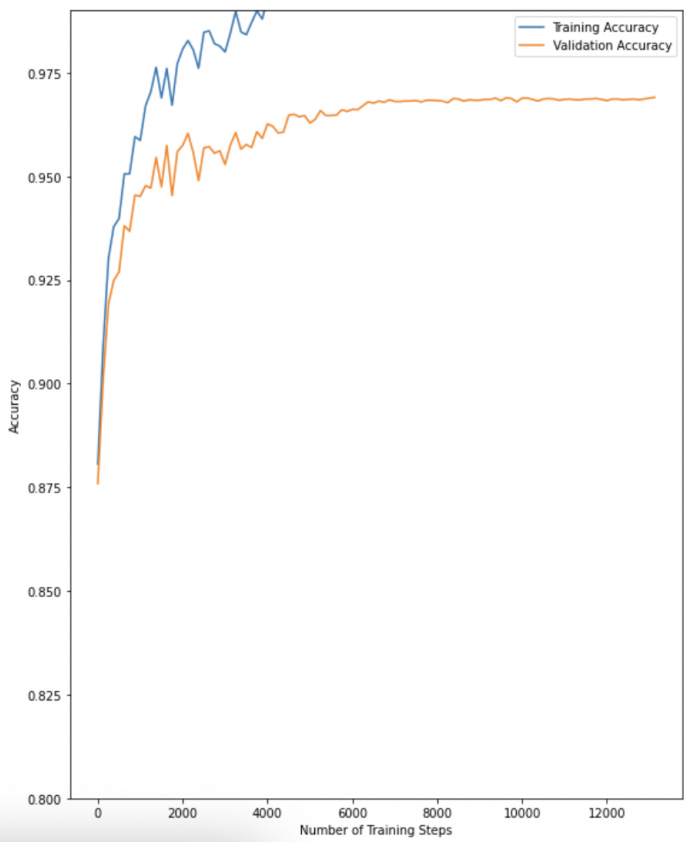


Task 4b)

When implementing 128 units in the hidden layer we can see that we achieve better accuracy and loss, but the convergence speed is the same as the model with 32 units (Task 4a). We also see that the model achieves a slightly better validation accuracy than the model with 64 units, but this is at the cost of slower convergence speed and slower training, due to more connections in the network. As we can see, the train accuracy is 1.0, which indicates that implementing 128 units in the hidden layer results in a less general model. This is a sign of overfitting which early stopping did not detect. If we were to use another dataset, the accuracy would probably not be as good.

```
Early stop at  21 epochs
Final Train Cross Entropy Loss: 0.0004199232931435683
Final Validation Cross Entropy Loss: 0.14495509860830158
Train accuracy: 1.0
Validation accuracy: 0.9691
```





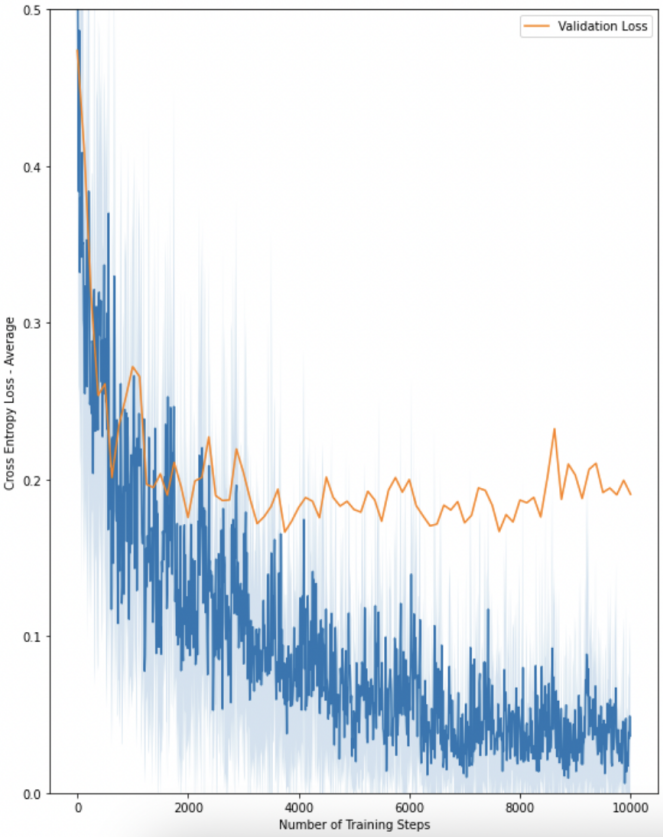
Task 4d)

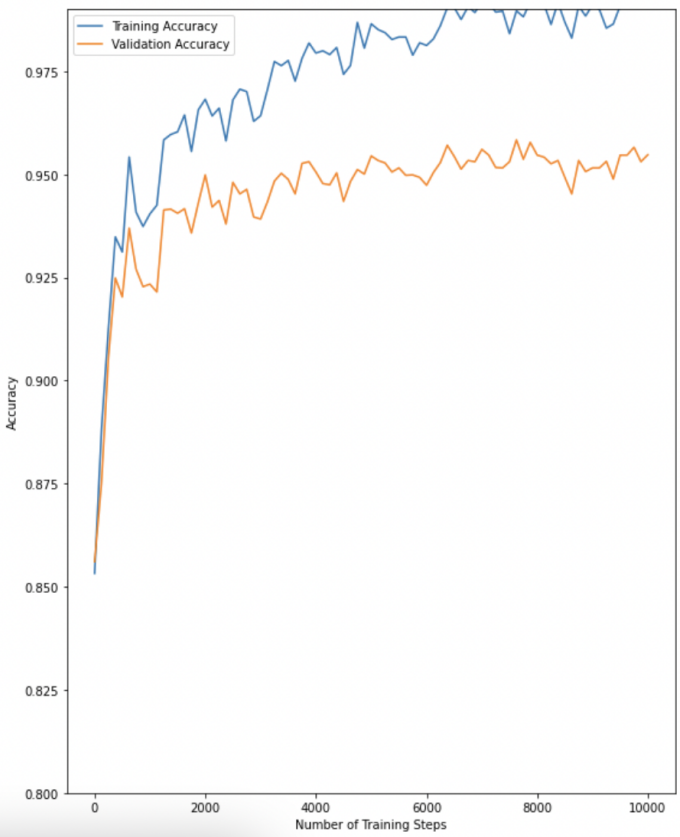
Number of parameters in our network from task 3: $(785 \cdot 64) + (64 \cdot 10) = \underline{\underline{50880}}$

Number of parameters in our network with 2 hidden layers with 60 units:

$(785 \cdot 60) + (60 \cdot 60) + (60 \cdot 10) = \underline{\underline{51300}}$

```
Early stop at  16 epochs
Final Train Cross Entropy Loss: 0.02514107151708828
Final Validation Cross Entropy Loss: 0.19061847807277307
Train accuracy: 0.99235
Validation accuracy: 0.9548
```

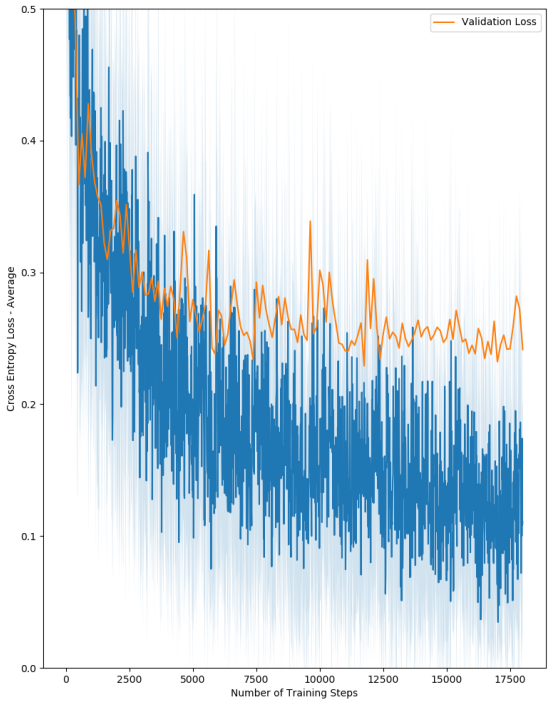


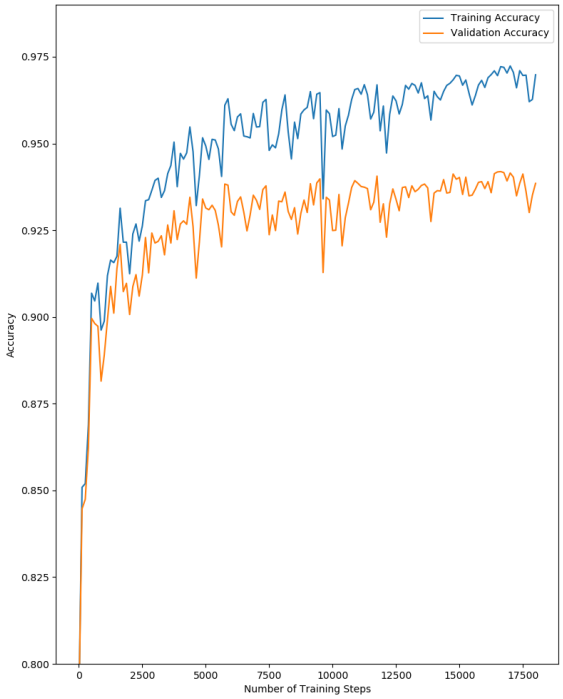


Task 4e)

If we increase the number of hidden layers in our model to ten, the backpropagated error in the early layers can be really small, as the weights are updated with the learning rate for each layer(here 0.02). This will cause the network to learn very slow, as we see from the figures. If we compare this to the optimized model from task 3, we can clearly see the loss does not decrease much over time, which is also an indication of slow learning. Another con with this deep architecture is that it takes a lot of time to train, and as we see it is neither more accurate nor more generalized. Thus, we conclude that ten hidden layers for this model is not an optimal solution.

```
Early stop at  28 epochs
Final Train Cross Entropy Loss: 0.10200317094314237
Final Validation Cross Entropy Loss: 0.2414377977811705
Train accuracy: 0.9698
Validation accuracy: 0.9385
```


Validation Loss
Cross Entropy Loss - Average
Number of Training Steps
0.0
0.1
0.2
0.3
0.4
0.5
0
2500
5000
7500
10000
12500
15000
17500


Training Accuracy
Validation Accuracy
Accuracy
Number of Training Steps
0.800
0.825
0.850
0.875
0.900
0.925
0.950
0.975
0
2500
5000
7500
10000
12500
15000
17500