TDT4265 - Computer Vision and Deep Learning

Vegard Iversen, Sebastian Skogen Raa

Feb 2022

1a

$$w_{ji} := w_{ji} - \alpha \frac{\partial C}{\partial w_{ji}} \tag{1}$$

We find the expression for $\alpha \frac{\partial C}{\partial w_{ji}}$

$$\begin{aligned} \alpha \frac{\partial C}{\partial w_{ji}} &= \alpha \cdot -\frac{1}{N} \sum_{n=1}^{N} \frac{\partial C^n(w)}{\partial w_{ji}} \\ \alpha \frac{\partial C}{\partial w_{ji}} &= -\alpha \frac{1}{N} \cdot N(y_j - \hat{y}_j) x_i \\ \alpha \frac{\partial C}{\partial w_{ji}} &= -\alpha (y_j - \hat{y}_j) x_i \end{aligned} \tag{2}$$

From previous calculations we know that $\sigma_j = -(y_k - \hat{y}_k)$ and can then write:

$$\alpha \frac{\partial C}{\partial w_{ji}} = \sigma_j x_i \tag{3}$$

Substituting this into Equation 1 we end up with:

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

(4)

1b

Update rule for hidden layer to output layer can be expressed on matrix form as:

$$\mathbf{W}_L := \mathbf{W}_L - \alpha \delta_L \odot \mathbf{A} \tag{5}$$

Similarly the update from in between the hidden layers will be:

$$\mathbf{W}_l := \mathbf{W}_l - \alpha \delta_l \odot \mathbf{Z} \tag{6}$$

In a network with I input layers in our case 785, j=64 number nodes in one hidden layer and j=10 output nodes will the dimensions of $\mathbf{W_l} = [785 \times 64]$ and $\mathbf{W_L} = [64 \times 10]$.

The δ_L will be dependent on the number of output nodes k while δ_l is dependent on the number of in the hidden layer nodes j. In our case with 64 nodes in the hidden layer, will the dimensions of δ_l be $[64 \times 1]$ and δ_L with $[10 \times 1]$.
$\mathbf{X}$ corresponds to the values (here pixel values) plus the bias. Therefor the dimension of $\mathbf{Z} = [785 \times 1]$.

2a

We get a mean of 33.5 and std of 78.9. We used zmap from scipy.stats which does the entire normalization.

2c

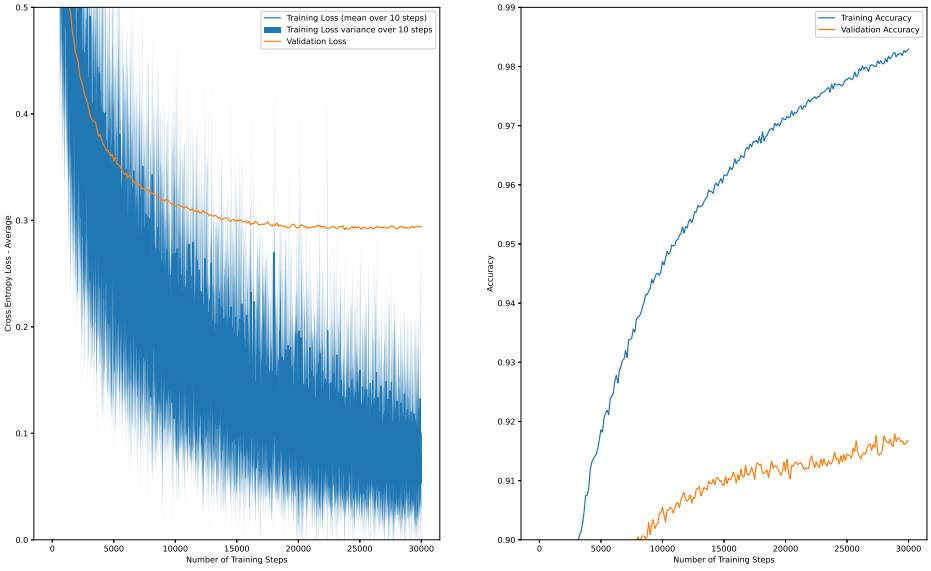

Figure 1: Accuracy plot

2d

$$\begin{aligned} \text{params} &= \text{num of weights} + \text{num of biases} \\ \text{params input and hidden} &= (784 + 1) \cdot 64 = 50240 \\ \text{params hidden and output} &= 64 \cdot 10 = 640 \\ \text{Total params} &= 50880 \end{aligned} \tag{7}$$

3c

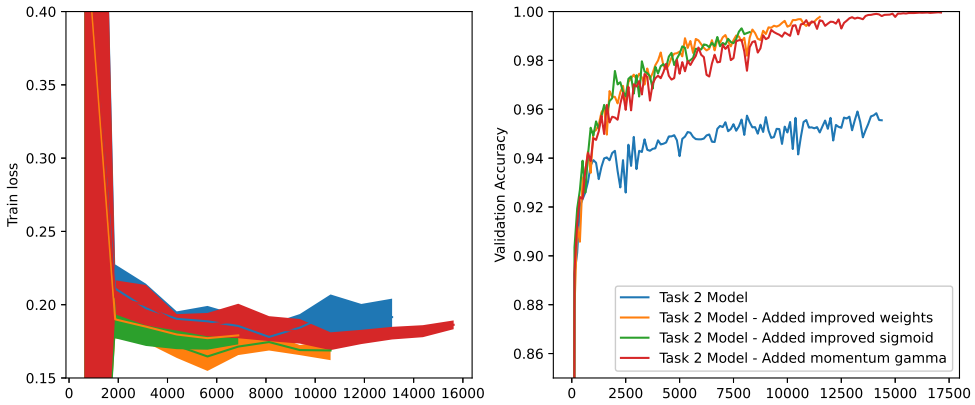

Figure 2: Accuracy plot 3c

As we can see there is an improvement of both learning speed and convergence when using improved weights versus the not improved weights. The difference between the train loss and validation loss seems to be decreased with the improved weights which can indicate that the model has generalized more.

When adding improved sigmoid also makes the model learn faster, but after a while the difference between loss and validation increases which can indicate overfitting. The improved sigmoid will be better as long as early stopping is used.

Adding moment increases the training time again, but again the validation loss increases after a while so the model will overfit. But this model needs the fewest amount of epochs/iterations to learn.

4a

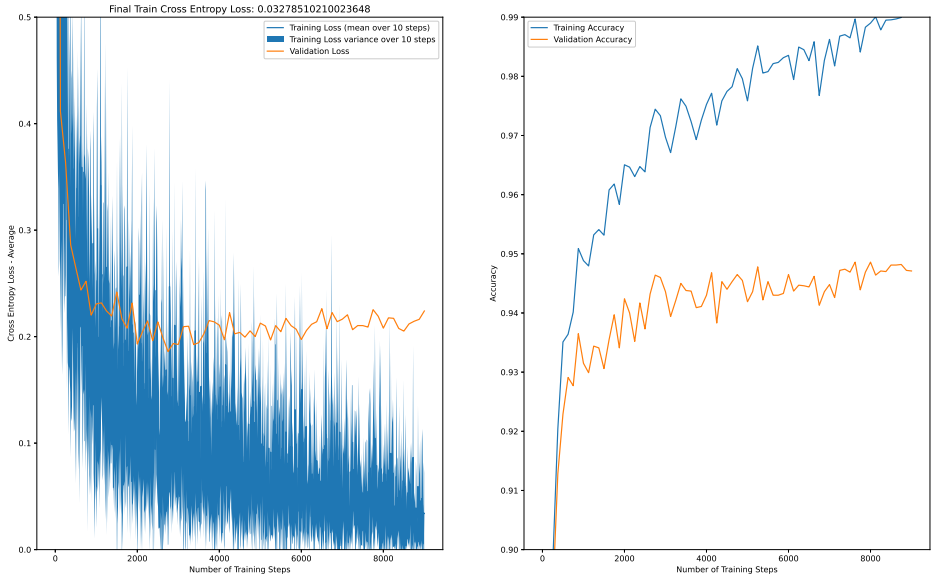

Figure 3: 4a: Plot of result from training with 32 hidden layer units.

Here we observe that the convergence speed is slower than the model from task 3. It also does not reach the same accuracy as the model in task 3.

4b

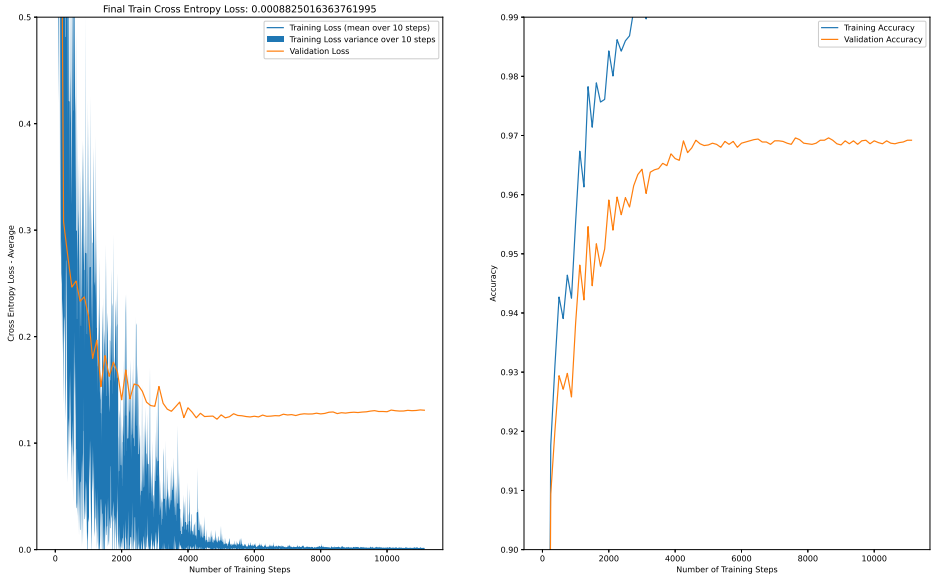

Figure 4: 4b: Plot of result from training with 128 hidden layer units.

Here we observe that the convergence speed is increased, but the accuracy is not improved compared to task 3.

4d

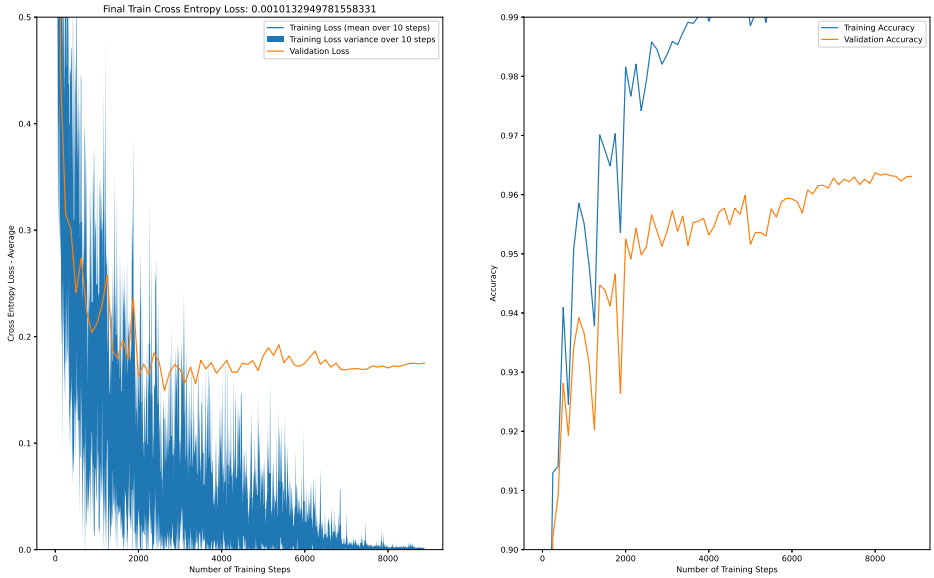

Figure 5: 4d: Plot of result from training a model using 2 layers with 64 hidden layer units each.

Number of parameters in task 3 is the same as computed in 2d (using same formula) with 50880. Number of parameters with 2 hidden layers is :

$$\text{total params} = 785 \cdot 64 + 64 \cdot 64 + 64 \cdot 10 = 54976 \tag{8}$$

There is not much difference in accuracy with this model and the task 3 model. But the loss is increased with the two hidden layers. The total number of hidden layer units is $2 \cdot 64 = 128$.

4e

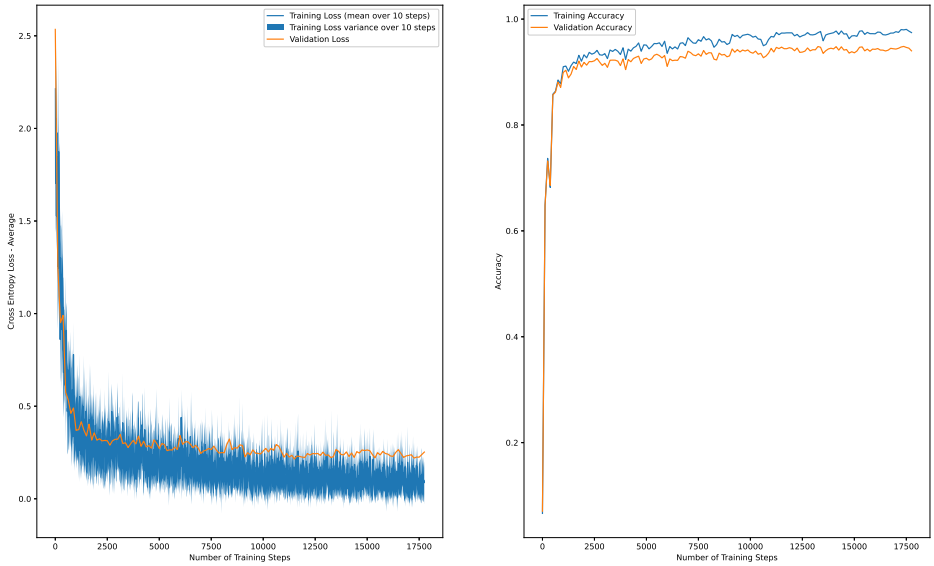

Figure 6: 4e: Plot of result from training a model using 10 layers with 64 hidden layer units each.

Both the accuracy and loss is worse in this model versus the task 3 model. The model is most likely to complex and it ”finds” patterns where its no useful pattern (for example). It is also slower as the size of the network is much greater. This shows that simply adding more layers does not mean a better model.