TDT4265: Computer Vision and Deep Learning

Sander M. Neraas
Olav R. Meberg

January 2021

NTNU | Norwegian University of Science and Technology

1 Task 1

1.1 a. Backpropagation

By using the definition of δ_j , show that the equation $w_{ji} := w_{ji} - \alpha \frac{\partial C}{\partial w_{ji}}$ can be written as; $w_{ji} := w_{ji} - \alpha \delta_j x_i$ and show that $\delta_j = f'(z_j) \sum_k w_{kj} \delta_k$.

Let's examine $\frac{\partial C}{\partial w_{ji}}$

$$\frac{\partial C}{\partial w_{ji}} = \sum_k \frac{\partial C}{\partial z_k} \frac{\partial z_k}{\partial a_j} \frac{\partial a_j}{\partial z_j} \frac{\partial z_j}{\partial w_{ji}}$$

where

$$\delta_k = \frac{\partial C}{\partial z_k}$$

$$\frac{\partial z_k}{\partial a_j} = \frac{\partial \sum_j w_{kj} a_j}{\partial a_j} = w_{kj}$$

$$\frac{\partial a_j}{\partial z_j} = \frac{\partial f}{\partial z_j} = f'(z_j)$$

$$\frac{\partial z_j}{\partial w_{ji}} = \frac{\partial \sum_i w_{ji} x_i}{\partial w_{ji}} = x_i$$

Hence:

$$\frac{\partial C}{\partial w_{ji}} = \delta_j x_i$$

and

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

OK.

1.2 b. Vectorize computation

(Please note that a matrix $\boldsymbol{M}$ is denoted with ***bold***)
Update rule form the output layer is given by:

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

with matrix notation we get

$$\boldsymbol{W}_{kj} = \boldsymbol{W}_{kj} - \alpha \boldsymbol{\delta}_k \boldsymbol{\alpha}_j^T$$

and now lets look at the hidden layer:

$$w_{ji} = w_{ji} - \alpha f'(z_j) \sum_k w_{kj} \delta_k x_i$$

and see that

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

is

$$\boldsymbol{\delta}_j = \boldsymbol{f}'(\boldsymbol{z}_j) \circ \sum_j = \boldsymbol{f}'(\boldsymbol{z}_j) \sum_j$$

leading to

$$\boldsymbol{W_{ji}} = \boldsymbol{W_{ji}} - \alpha \boldsymbol{f}'(\boldsymbol{z}_j) \sum_j \boldsymbol{x}_i^T$$

where

$$\boldsymbol{z_j} = \boldsymbol{x_j} \boldsymbol{W}_{ji}^T$$

2 Task 2

2.1 c. Plot of loss and accuracy

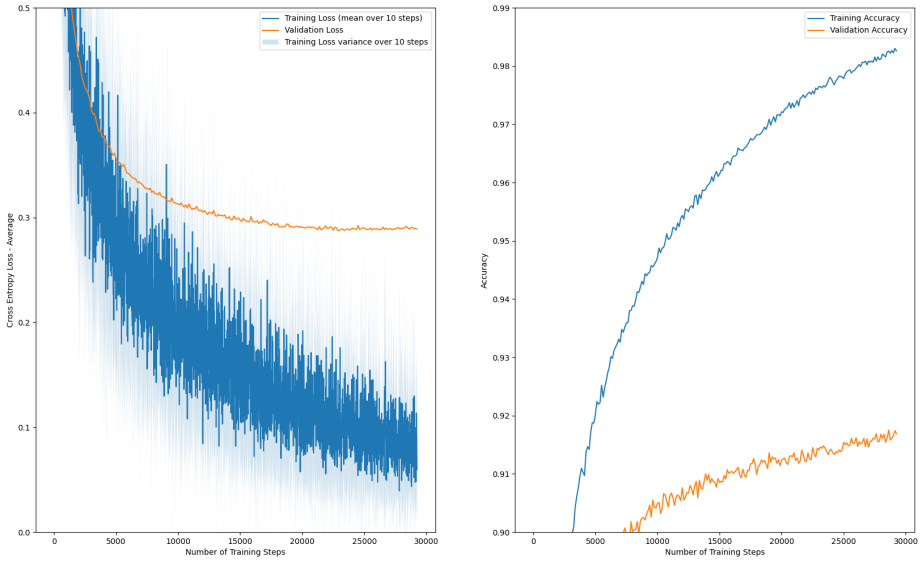

Figure 1: Loss and accuracy over training and validation.

2.2 d. Parameters

The number of parameters for the model is given by $Numberofparameters = numberofweights + numberofbiases$. Since we have used the bias trick, the number of parameters are only dependent on the number of weights. This gives us $785 * 64 + 64 * 10 = 50.880$ parameters.

3 Task 3

When gradually applying the "tricks of the trade" the model drastically changed in terms of performance. However, the implementation of an improved sigmoid function did not improve on the models performance as seen in 3. With the improved sigmoid function, the model stopped after 21 epochs with a validation accuracy of 0.8997, while the original model stopped after 46 epochs with a validation accuracy of 0.9158. This is a result of that the model with improved sigmoid is converging faster, and thus stops early to prevent overfitting. When we applied the improved weight initialization, the models performance drastically changed with respect to loss and accuracy, and further improved with the implementation of momentum. All three model that used the "tricks of the trade" converged faster than the original.

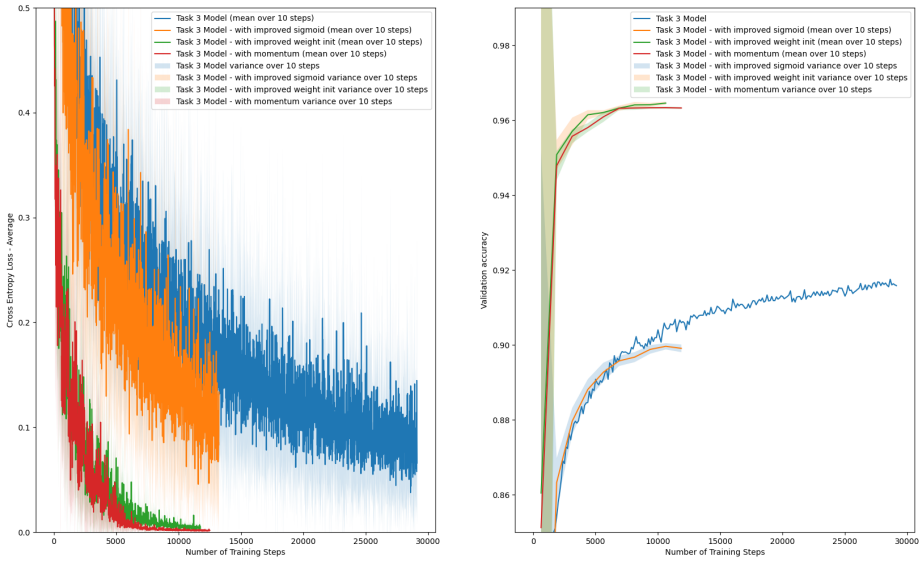

Figure 2: Loss and accuracy over training and validation.

4 Task 4: Experiment with network topology

4.1 a. Setting number of hidden units to 32

If the number of hidden units are too small, the model will produce a higher loss and a lower validation accuracy. This can be seen in 4.2.

4.2 b. Setting number of hidden units to 128

If the number of hidden units are too large, the model will perform better in terms of loss and accuracy. This can be seen in 4.2. However, the model is slower due to more computations.

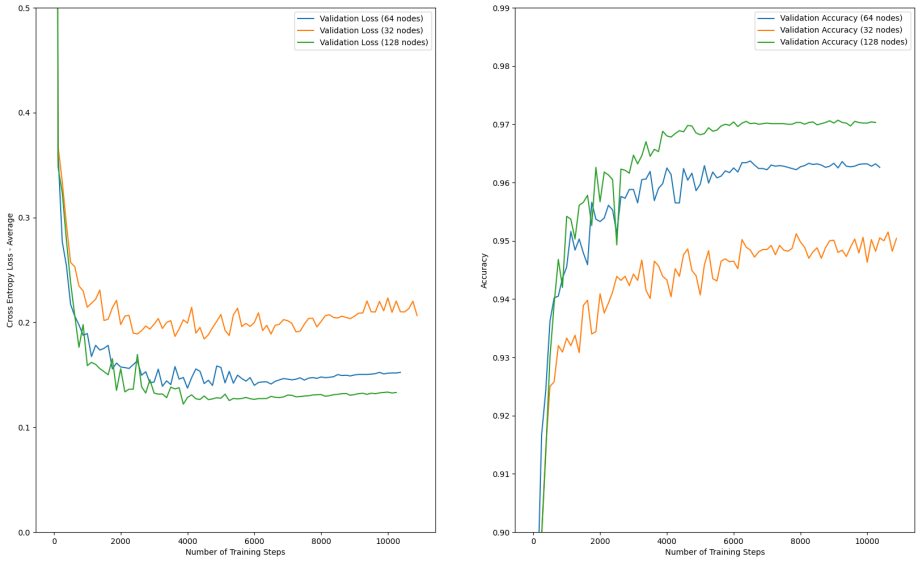

Figure 3: Loss and accuracy over validation.

4.3 d. Model with two hidden layers

The model from task 3 has a total of 50880 parameters. In order to get the same amount of parameters we solved the equation $50880 = 785x + x^2 + x10$, which gave $x = 59.54$. Thus, we chose to use 60 nodes in each hidden layer, resulting in 51300 parameters.

When adding more hidden layers, the model is often more prone to overfitting. This can be clearly seen in 4.3 where the training loss is dropping towards 0.0 while the validation loss stabilizes at 0.2. It is also seen in the accuracy plot where the training and validation accuracy is clearly splitting after about 500-1000 training steps. The performance of this model is comparable to the model with a single hidden layer with 64 hidden nodes, but this model is out-performed by the model with a single hidden layer with 128 hidden nodes.

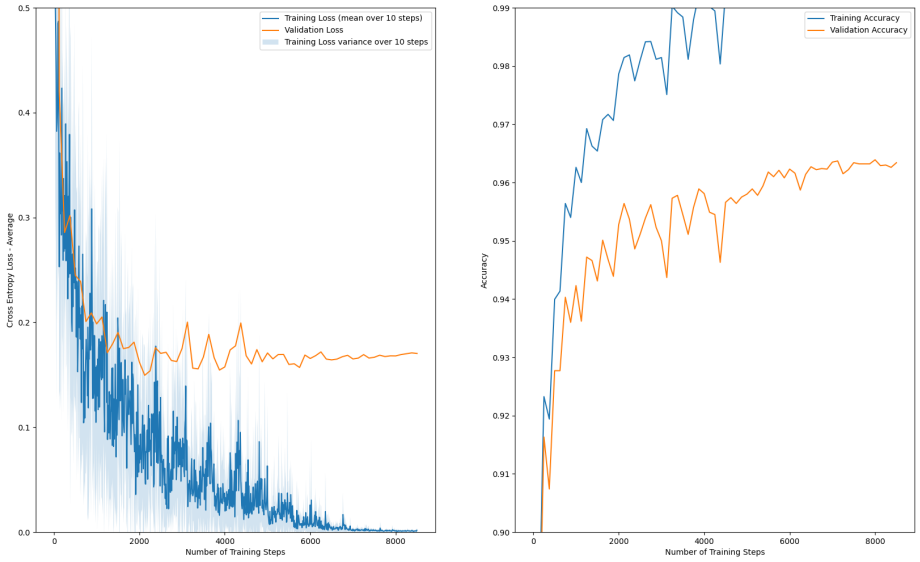

Figure 4: Loss and accuracy over training and validation.

4.4 e. Model with ten hidden layers

With ten hidden layers, we expected the model to overfit. The training took a lot longer time with `num_epochs` set to 50. The results from the training can be seen in 4.4. The model is clearly overfitting, and the results are comparable to the ones in the last task. The final validation accuracy is about 0.945. Compared to the model with a single hidden layer with 128 nodes, this is a drop of almost 0.03.

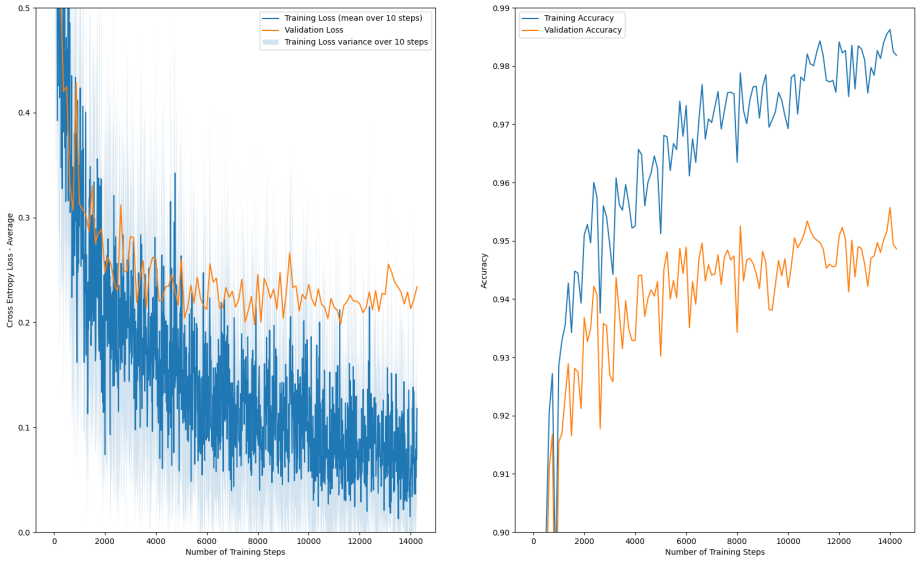

Figure 5: Loss and accuracy over training and validation.

* * *