Assignment 2 Report

Group 121 – Etienne Gaucher, Benedikt Blumenstiel

Task 1

Task 1a)

We first use the chain rule.

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

We have $\frac{\partial C}{\partial z_k} = \delta_k$ by definition, $\frac{\partial z_k}{\partial a_j} = w_{kj}$, $\frac{\partial a_j}{\partial z_j} = f'(z_j)$ and $\frac{\partial z_j}{\partial w_{ji}} = x_i$.

We get

$$\frac{\partial C}{\partial w_{ji}} = \sum_k \delta_k w_{kj} f'(z_j) x_i = f'(z_j) x_i \sum_k \delta_k w_{kj}$$

Thus

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

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

Task 1b)

We refer to W_2 as the weight matrix from the hidden layer to the output layer, having a shape of $j \times k$ with k is the number of output neurons and j the number of hidden neurons. Similar, W_1 is the weight matrix from the input layer to the hidden layer with a shape of $j \times i$, where i is the number of input neurons. α refers to the learning rate.

δ_2 (shape $j \times k$) is the gradient between hidden layer and output layer, δ_1 (shape $j \times i$) is the gradient between input layer and hidden layer. x is the input vector (length i), y the target vector (length k) and o the predicted output vector (length k). z is the output of the hidden layer before activation and a the output of the hidden layer after activation (both length j). f' is the derivative sigmoid function.

$$W_2 = W_2 - \alpha \times \delta_2$$

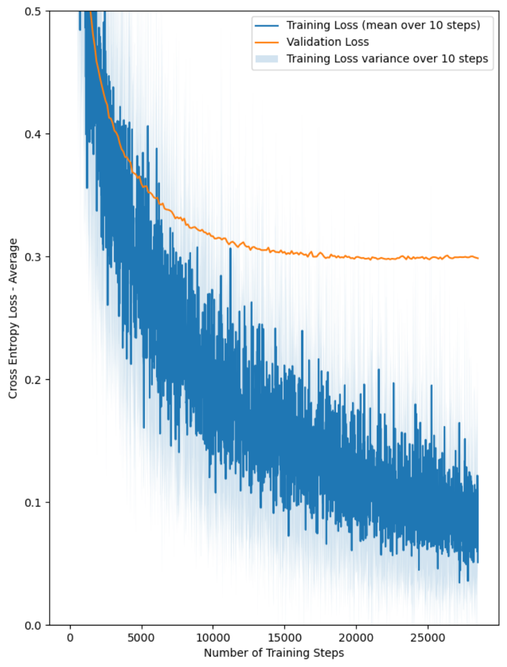
$$W_1 = W_1 - \alpha \times \delta_1$$

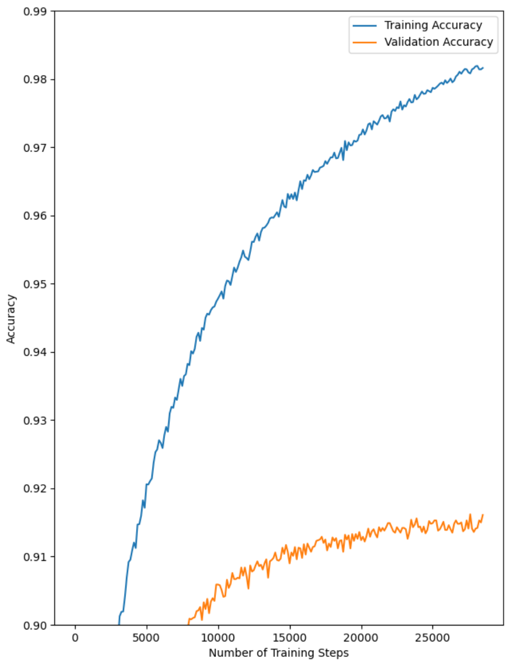
$$\delta_2 = (a)^T \cdot -(y - o)$$

$$\delta_1 = (X)^T \cdot -(y - o) \cdot (W_2)^T \circ f'(z_1)$$

Task 2

Task 2c)

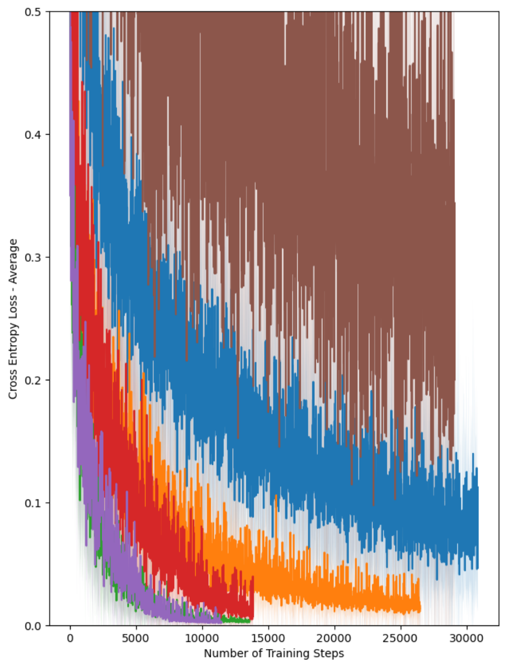


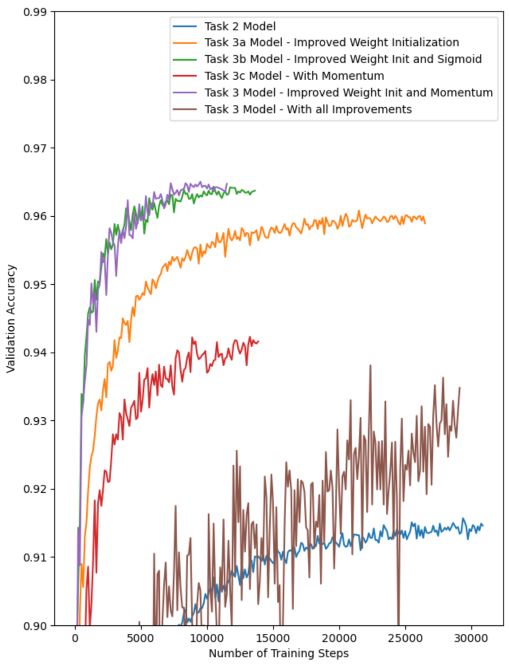


Task 2d)

$$\begin{aligned}\text{Number of parameter} &= \text{number of weights} + \text{number of biases}\\ &= 785 \times 64 + 64 \times 10 + 0\\ &= 50880\end{aligned}$$

Task 3





We can sort all the methods according to learning speed/convergence speed: model 2 < model with all improvements < model with momentum < model with improved weight initialization < model with improved weight initialization and momentum < model with improved weight initialization and sigmoid.

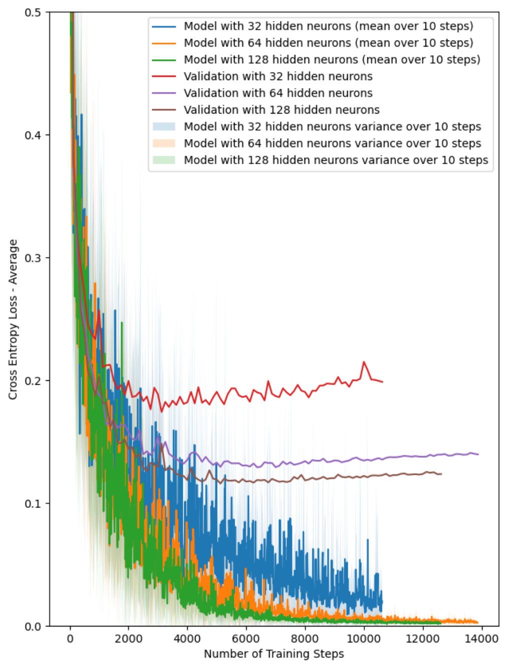
The different methods improve a lot the learning speed, but not so much if we use all the tricks at the same time. We do not know the exact explanation of the bad performance, when combining improved sigmoid with momentum. Perhaps there is a overlapping between the methods resulting in bad performance. For example, the improved sigmoid could lead to a very high momentum which leads the model to pass the optimum.

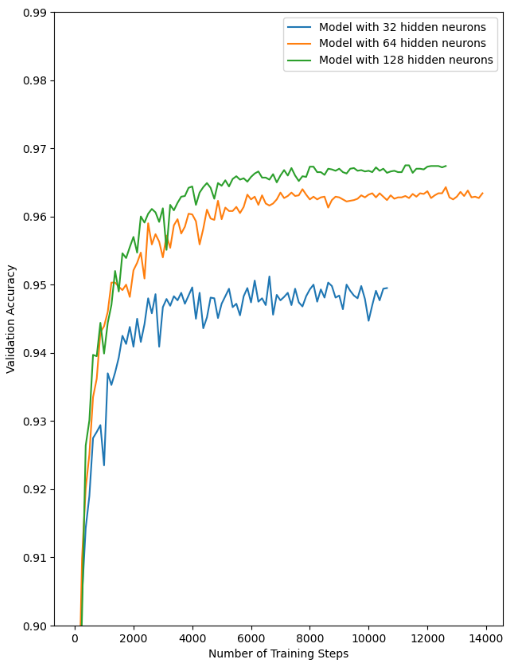
We can notice that the use at the same time of improved sigmoid and weight initialization overfit data if we don't stop the learning with early stopping, because the validation cost improves after a certain number of epoch. It is also the case for the model with improved momentum and all the improvements.

Regarding the final accuracy, the model with improved weight initialization and momentum has the best score with a accuracy of 0.9643. Therefore, the best models in terms of learning speed and validation loss are the model with improved sigmoid and weight initialization as well as improved weight initialization and momentum.

Task 4

Task 4a)





Reducing the number of neurons in the hidden layer to 32 results in a smaller accuracy and higher training loss. This is caused by a smaller capacity of the model (number of training parameters) that is not able to generalize enough.

Task 4b)

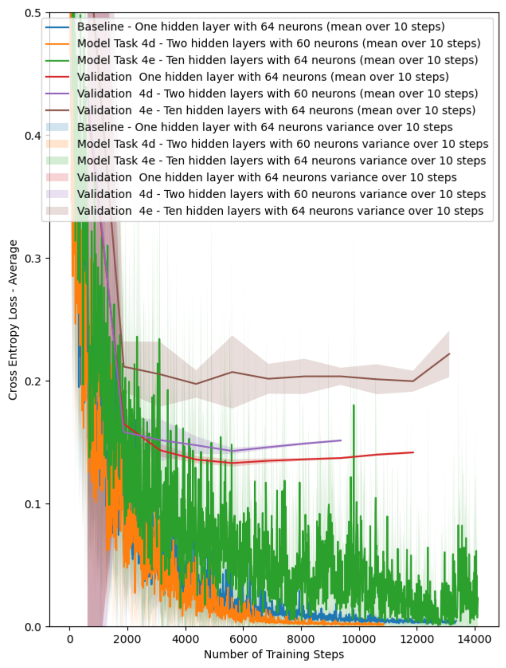
The accuracy of the model with 128 hidden neurons is increased compared to the model with 64 hidden neurons, as shown in the previous plot. The increased capacity of the model made it possible to generalize better, but it is also increasing the computation time. Therefore, we have to find a trade-off between increased complexity and improved performance. Increasing the capacity further can also lead to overfitting, which we did not notice in our experiments.

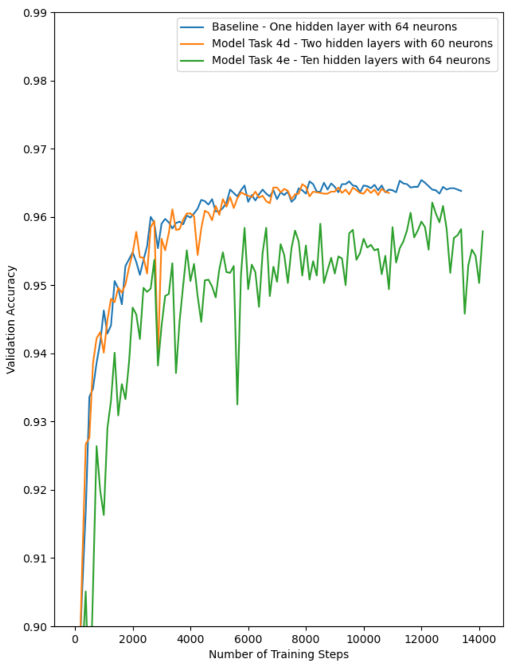
Task 4d)

We are looking for a number of neurons per hidden layer equal to the same number parameter than the previous model.

$$\text{Number of parameter} = \text{number of weights} + \text{number of biases}$$
$$50880 = 785 \times x + x \times +x \times 10 + 0$$
$$0 = x^2 + 795x - 50880$$
$$x = \frac{795}{2} + \sqrt{(\frac{795}{2})^2 + 50880}$$
$$x = 59.5407$$

Therefore, we train the model with two hidden layer with 60 neurons each, resulting in 51 300 parameters.





The accuracy of the model with two hidden layers is compareable to the model with one hidden layer. This shows, that in our case the capacity of a model (number of parameters) has a higher influence than the architecture of the model. Apart from the accuracy we notice a lower training loss with two hidden layers. Perhaps the deep structure allow a better optimization of the loss function.

Task 4e)

As presented in the previous plot, the model with ten hidden layers has a worse performance compared to models with less layers. The complexity of the model is too high to learn the problem appropriate. This can be caused for example by vanishing gradient. The accuracy and the loss fluctuate a lot, showing a unstable training. The validation loss is also increasing, showing signs of overfitting and results in earnly stopping of the training process. All in all, the high number of hidden layers is not optimal for our implementation.