Introduction

In this lab, I created a neural network based on one layer for classification of 10 different images based on 32 x 32-pixel pictures, from a dataset called CIFAR 10. The training was done using mini-batch gradient descent for optimizing the weight matrix. As a cost function I used cross entropy, which takes the value of the true label and the given predicted value to calculate the loss.

The assignment

One of the most important parts of the network is the gradient descent implementation, which updates the weight matrix. For comparing the reliability of the computed gradients, I measured the Euclidian distance between the vectors and matrices. The numerical computations of the gradients were done by the function ComputeGradsNumSlow(). The Euclidian distance I got were:

$$\text{Euclidian distance}(\nabla W_{GradsNumSlow}, \nabla W) = 4.0789 * 10^{-6}$$

$$\text{Euclidian distance}(\nabla b_{GradsNumSlow}, \nabla b) = 7.6 * 10^{-6}.$$

The distance seems to be small, with a factor of 10^{-6}. Thus, can conclude that the implemented gradient function was working relatively well.

After checking if the gradient was working correctly, it was time for testing the network with different parameters.

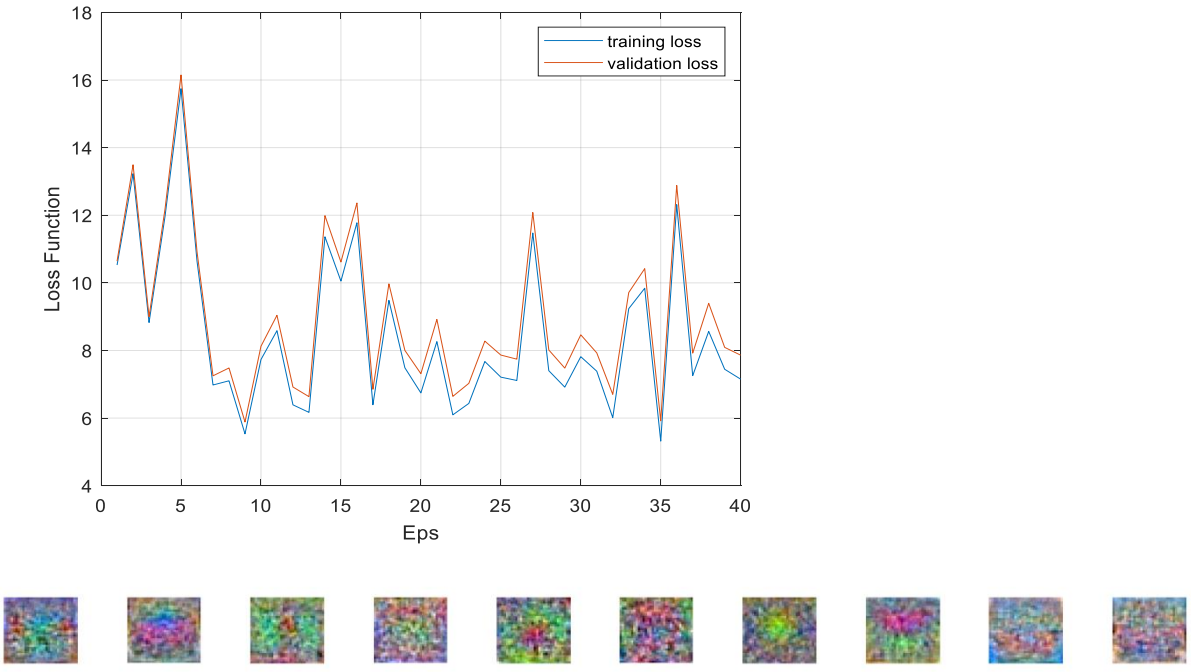

Fig 1. Graph of the loss with cross entropy and images of the weight matrix after training. The parameters used: $\lambda = 0, \mathrm{N}_{\mathrm{Epochs}} = 40, \mathrm{N}_{\mathrm{Batch\ size}} = 100, \eta = 0.1$

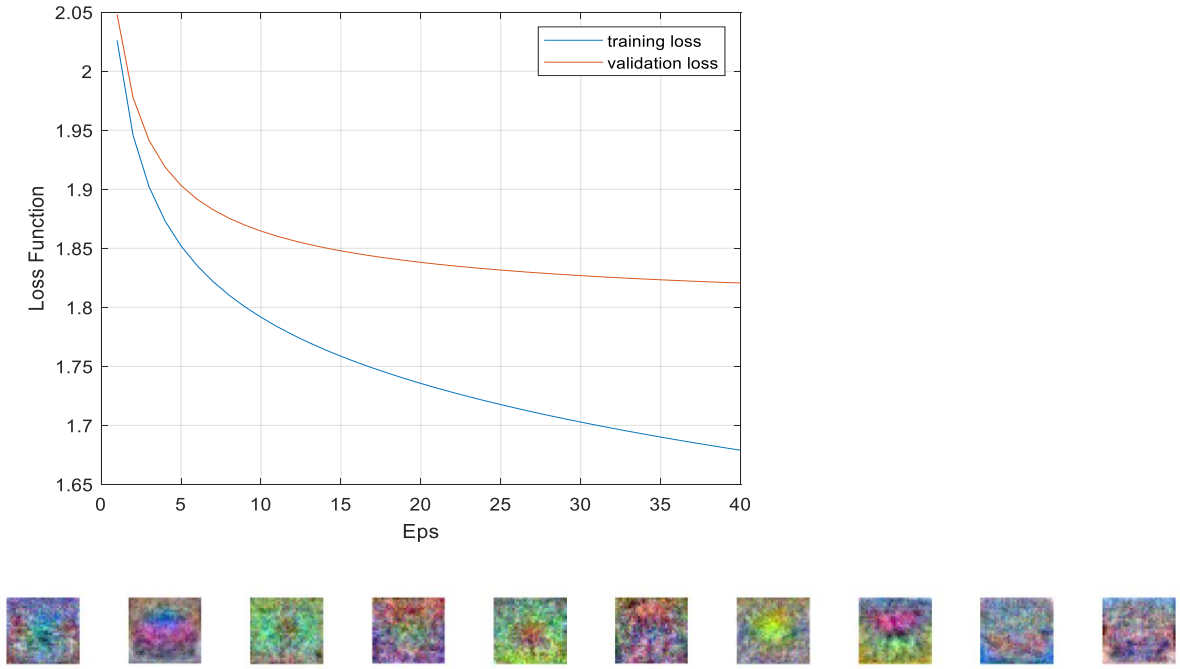

Fig 2. Graph of the loss with cross entropy and images of the weight matrix after training. The parameters used: $\lambda = 0, \mathrm{N}_{\mathrm{Epochs}} = 40, \mathrm{N}_{\mathrm{Batch\ size}} = 100, \eta = 0.01$

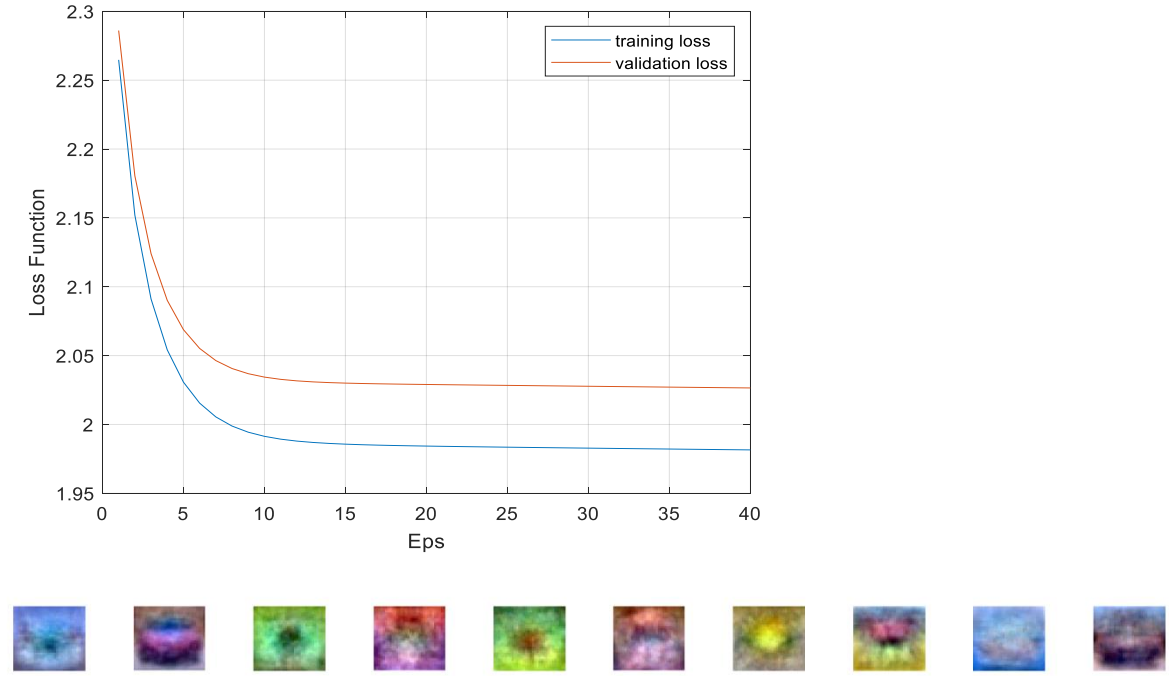

Fig 3. Graph of the loss with cross entropy and images of the weight matrix after training. The parameters used: $\lambda = 0.1, \mathrm{N}_{\mathrm{Epochs}} = 40, \mathrm{N}_{\mathrm{Batch\ size}} = 100, \eta = 0.1$

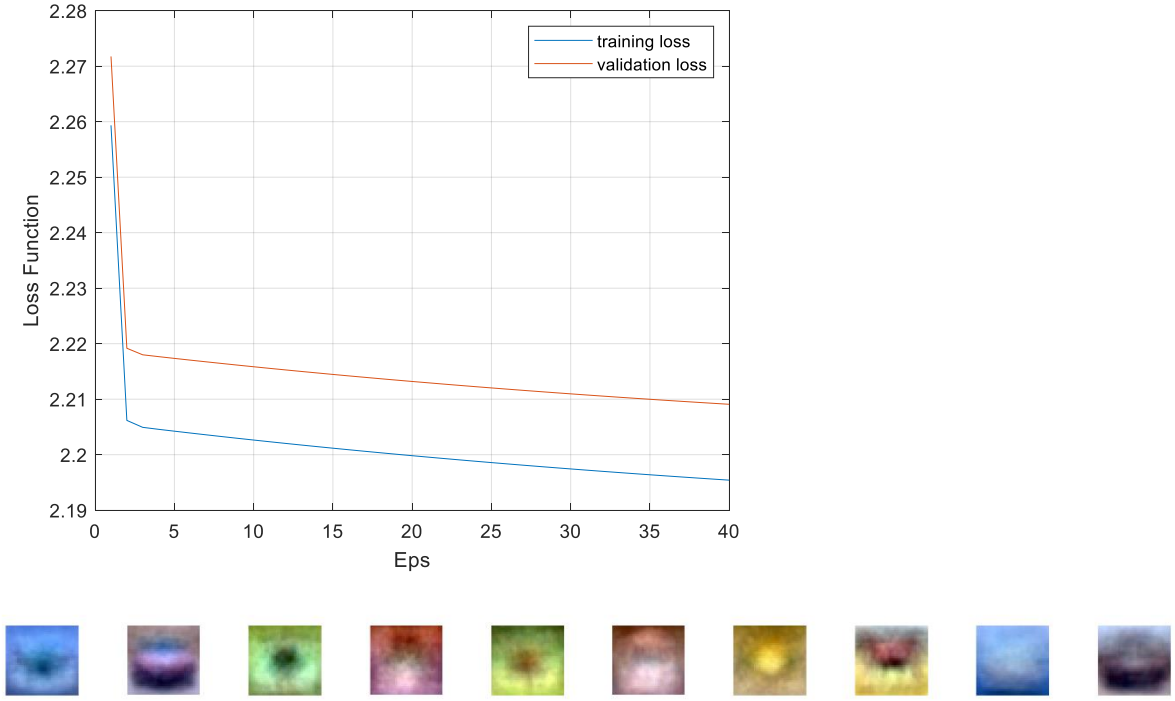

Fig 4. Graph of the loss with cross entropy and images of the weight matrix after training. The parameters used: $\lambda = 1, \mathrm{N}_{\mathrm{Epochs}} = 40, \mathrm{N}_{\mathrm{Batch\ size}} = 100, \eta = 0.1$

λ	$\mathrm{N}_{\mathrm{Epochs}}$	$\mathrm{N}_{\mathrm{Batch\ size}}$	η	Training Accuracy	Validation Accuracy	Test Accuracy
0	40	100	0.1	0.3100	0.2691	0.2664
0	40	100	0.01	0.4167	0.3634	0.3667
0.1	40	100	0.01	0.3416	0.3202	0.3336
1	40	100	0.01	0.2226	0.2129	0.2193

Table 1. Final accuracy for different setup of parameters

In the cost function I added a regularization term λ on the weight matrix. The aim is to minimize the loss. For instance, the problem with high values in W is solved by minimizing lambda. As we can see comparing the plots of the loss from fig 2, fig 3 and fig 4. For higher lambda values, it pushes down the loss for few epochs to a certain loss. Thus, as the loss is high the regularization term minimizes the loss immediately.

The right learning rate is important because for instance with a higher rate can make the iteration of finding some local minima of the gradient a bit harder. By skipping the smallest local minima and ending up at a local minimum which is not the most optimal for the system.