Deep Learning in Data Science, DD2424

Short report on assignment 2

Image classification with a two-layer network

Côme Lassarat

April 16, 2022

1 Introduction

This work aims to implement a k-layer network to classify images from the CIFAR-10 dataset. The network designed will be trained using mini-batch gradient descent and cyclical learning rates. First, the tests ran to verify if the different functions (such as the gradients computation) are correct will be presented. In a second place, the different scenarios with relevant plots will be described.

2 Implementation checking

One of the main difficulty of this assignment is to correctly implement the analytical gradients calculation. In order to do so, a separate function were used to compute the numerical gradients. Then, the relative error (element wise) between the numerical gradients and the analytical gradients (implemented in the network) was calculated on a batch from the dataset used. Finally, the maximum of these relative errors is analyzed.

For $\lambda = 0$

- For 2 layers (50 hidden nodes), the maximum relative error among all the parameters is $8.37e-7$
- For 3 layers (50 hidden nodes at every layer), the maximum relative error among all the parameters is $9.6e-7$

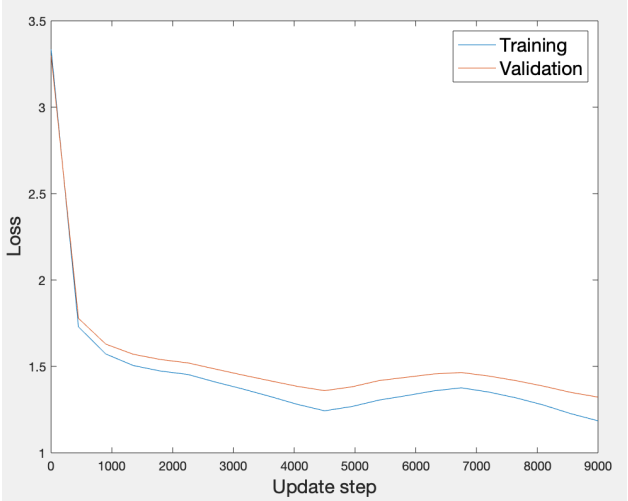
For λ not null, the same orders of magnitude is obtained. Given these small errors, we can then assumed our gradient implementation is correct.

3 Training the neural network with cyclical learning rates

3.1 Training a 3-layer neural network

3.1.1 Without batch normalization

The loss function plots for two training cycles can be seen on Figure 1 for the following configuration: $\eta_{min} = 10^{-5}$, $\eta_{max} = 10^{-1}$, $\lambda = 0.005$, $n_{batch} = 100$ and $n_s = 5 * 45000/n_{batch}$, length of the hidden layers: $[50, 50]$. The final test accuracy at the end of training is 52.09% (Table 1).



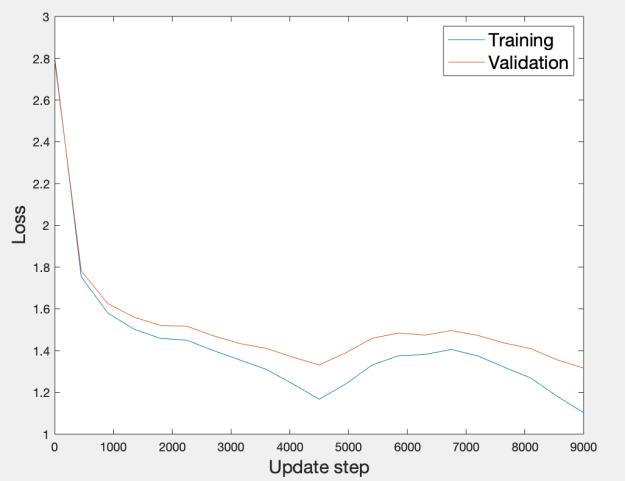
Figur 1: Plot of the loss function over the update steps

Test accuracy
52.09%

Tabell 1: Final accuracy

3.1.2 With batch normalization

The loss function plots for two training cycles can be seen on Figure 2 for the following configuration: $\eta_{min} = 10^{-5}$, $\eta_{max} = 10^{-1}$, $\lambda = 0.005$, $n_{batch} = 100$ and $n_s = 5 * 45000/n_{batch}$, length of the hidden layers: $[50, 50]$. The final test accuracy at the end of training is 52.13% (Table 2).



Figur 2: Plot of the loss function with batch normalization over the update steps

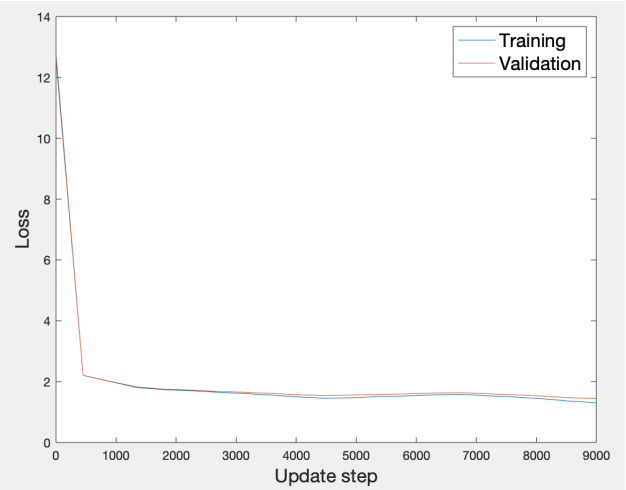
Test accuracy
52.13%

Tabell 2: Final accuracy

3.2 Training a 9-layer neural network

3.2.1 Without batch normalization

The loss function plots for two training cycles can be seen on Figure 3 for the following configuration: $\eta_{min} = 10^{-5}$, $\eta_{max} = 10^{-1}$, $\lambda = 0.005$, $n_{batch} = 100$ and $n_s = 5 * 45000/n_{batch}$, length of the hidden layers: $[50, 30, 20, 20, 10, 10, 10, 10]$, He Initialization. The final test accuracy at the end of training is 47.57% (Table 3).



Figur 3: Plot of the loss function over the update steps

Test accuracy
47.57%

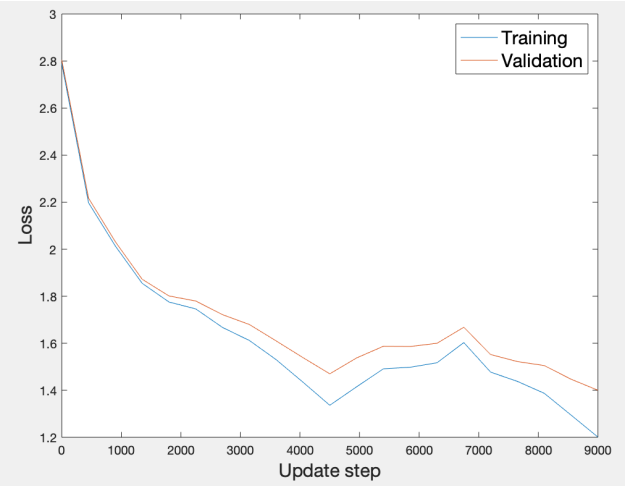
Tabell 3: Final accuracy

3.2.2 With Batch Normalization

The loss function plots for two training cycles can be seen on Figure 4 for the following configuration: $\eta_{min} = 10^{-5}$, $\eta_{max} = 10^{-1}$, $\lambda = 0.005$, $n_{batch} = 100$ and $n_s = 5 * 45000/n_{batch}$, length of the hidden layers: $[50, 30, 20, 20, 10, 10, 10, 10]$, He Initialization. The final test accuracy at the end of training is 51.44% (Table 4).

Test accuracy
51.44%

Tabell 4: Final accuracy



Figur 4: Plot of the loss function with batch normalization over the update steps

3.3 Observations

The results exposed above highlight the advantage of using batch normalization especially when the network has many layers (for 9 hidden layers, the test accuracy is about 47.57% without batch normalization (Table 3) and 51.44% with batch normalization (Table 4)). For two hidden layers, the performance are very similar.

4 Searching Lambda

To improve the performance of the network, a coarse-to-fine search is done to find a good value of lambda. The search is led on 2 hidden layers network $[50, 50]$ with $\eta_{min} = 10^{-5}$, $\eta_{max} = 10^{-1}$, $n_{batch} = 100$, $n_{cycles} = 2$ and $n_s = 5 * 45000/n_batch$.

4.1 Coarse search

First, several values of lambda are tried on a uniform log-scale from 10^{-1} to 10^{-5}.

- Best λ values: $[0.0019,\ 0.0072,\ 0.0001]$
- Best associated validation accuracies: $[52.96\%,\ 52.86\%,\ 52.12\%]$

4.2 Fine search

Finally, the lambda search is focused on a narrower range around the values of λ previously found, for 2 cycles.

- Best λ values: $[0.0035,\ 0.0036,\ 0.0086$
- Best associated validation accuracy: $[53.80\%,\ 53.66\%,\ 53.48\%]$

Thus, we found $\lambda_{best} = 0.0035$.

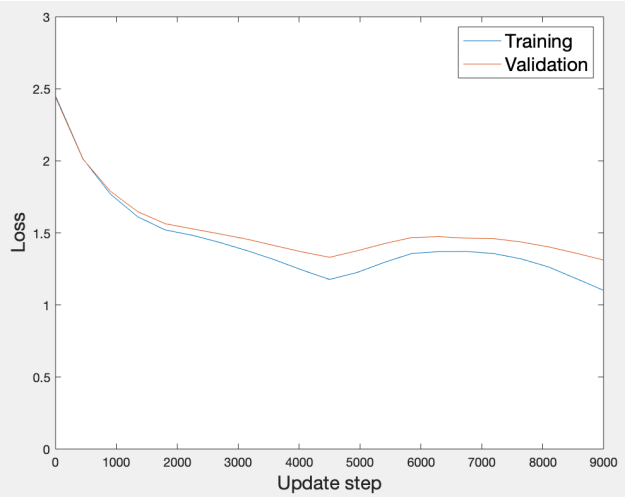
The test accuracy achieved is 52.29%.

5 Sensitivity to initialization

Instead of using He initialization, the weight parameters are initially normally distributed according to a varying standard deviation : $[0.1,\ 0.001,\ 0.0001]$. The performance a two hidden layers network $[50, 50]$ is tested, with $\eta_{min} = 10^{-5}$, $\eta_{max} = 10^{-1}$, $n_{batch} = 100$, $n_s = 5 * 45000/n_batch$ and $\lambda = 0.005$, with and without batch normalization.

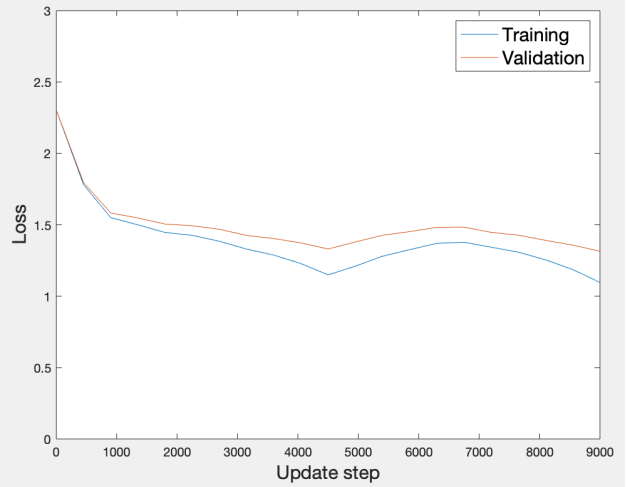
5.1 With batch normalization

5.1.1 $\sigma = 0.1$



Figur 5: Plot of the loss function over the update steps

5.1.2 $\sigma = 0.001$



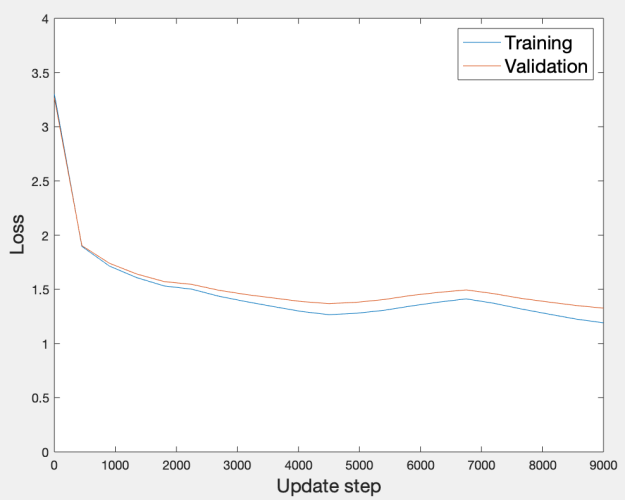
Figur 6: Plot of the loss function over the update steps

5.1.3 $\sigma = 0.0001$

Figur 7: Plot of the loss function over the update steps

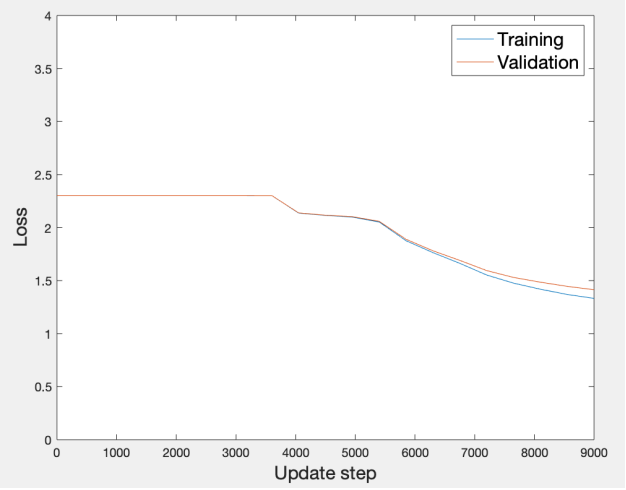
5.2 Without batch normalization

5.2.1 $\sigma = 0.1$

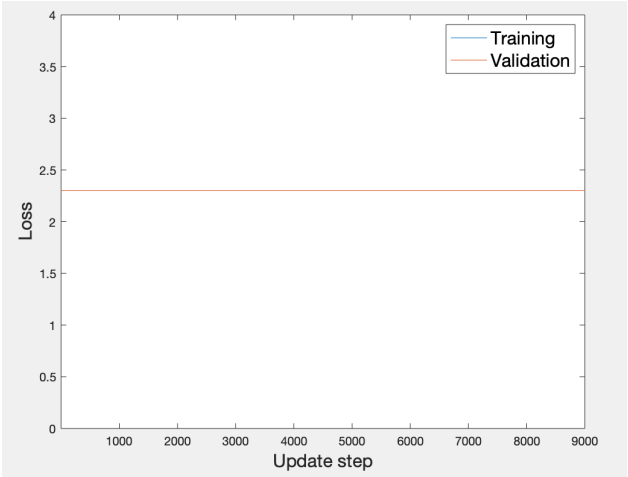


Figur 8: Plot of the loss function over the update steps

5.2.2 $\sigma = 0.001$



Figur 9: Plot of the loss function over the update steps



Figur 10: Plot of the loss function over the update steps

5.2.3 $\sigma = 0.0001$

5.3 Observations

It can be seen on the previous figures (from 5 to 10) another advantage of using batch normalization: the network performance is much less sensible to the initialization of weights when batch normalization is applied. On Figure 5 to 8, decreasing σ doesn't seem to have much impact on the gradient descent process and thus on the learning process (the loss functions decrease in almost the same way). On the other hand, without batch normalization (Figure 8 to 10), when σ is very small, the network doesn't learn anything: the loss functions are constant.