DD2424 Deep Learning Assignment 1

Mikael Ljung (milju@kth.se)
940312–7435

May 2020

1. Introduction

In this paper a K-layer network with multiple outputs was trained to classify images from the CIFAR10-dataset. The CIFAR100 consists of 60 000 32x32 color images in 10 classes, with 6000 images per class.

2. Methods

The K-layer network was built from scratch in Python using mini-batch gradient descent on a cost function that calculated the cross-entropy loss. To further improve the algorithm, batch normalization was implemented in each layer. To make sure that the analytical gradient was calculated correctly it was compared to a numerical calculated gradient. Furthermore, the learning rate was set using a cyclical range of values, and the lambda parameter was found by first doing coarse search and after that a fine search to find its optimal value.

3. Results

3.1. Gradients

The analytically computed gradient for the network with batch normalization was compared to a numerically computed gradient to verify that it was correct. This comparison was conducted using the following formula:

$$\frac{|g_a - g_n|}{\max(\text{eps}, |g_a| + |g_n|)}$$

The values for the respective variable for a two-layer network are presented in the table below:

	First layer	Second layer
∑ gnW	-5.478684538706702	-3.552713678800501e-09
∑ gnb	2.220446049250313e-10	-6.661338147750939e-10
∑ gaW	-5.4570943687837214	-4.440892098500626e-16
∑ gab	-1.5513848886095086e-17	8.673617379884035e-18
∑ εw	**0.003516528722559895**	**2.887309333594692e-09**
∑ εb	**0.00022204471630403626**	**2.0000286752554976e-09**

The values for the respective variable for a three-layer network are presented in the table below:

	First layer	Second layer	Third layer
∑ gnW	-2.074731690226983	2.5772543872371045	3.9968028886505635e-09
∑ gnb	6.661338147750939e-10	0	4.440892098500626e-10
∑ gaW	-2.112196196099384	2.5831415587193858	0
∑ gab	3.848158770802801e-17	5.227806035290604e-17	-4.163336342344337e-17

∑ εw	**0.0036796813823165016**	**0.0027625857872718954**	**2.51094569232485e-09**
∑ εb	**0.0011102231724550386**	**2.677307160692255e-10**	**1.913662346627803e-09**

It can be seen from the tables that the difference between the numerical and the analytical gradient is small, and therefore it can be concluded that the analytical gradient is calculated correctly.

3.2. Loss

By setting the hyperparameters eta_min = 1e-5, eta_max = 1e-1, lambda=0.05, n_s=2250 and training for two cycle the following plots showcasing the loss were generated:

3.2.1. 3-layer network

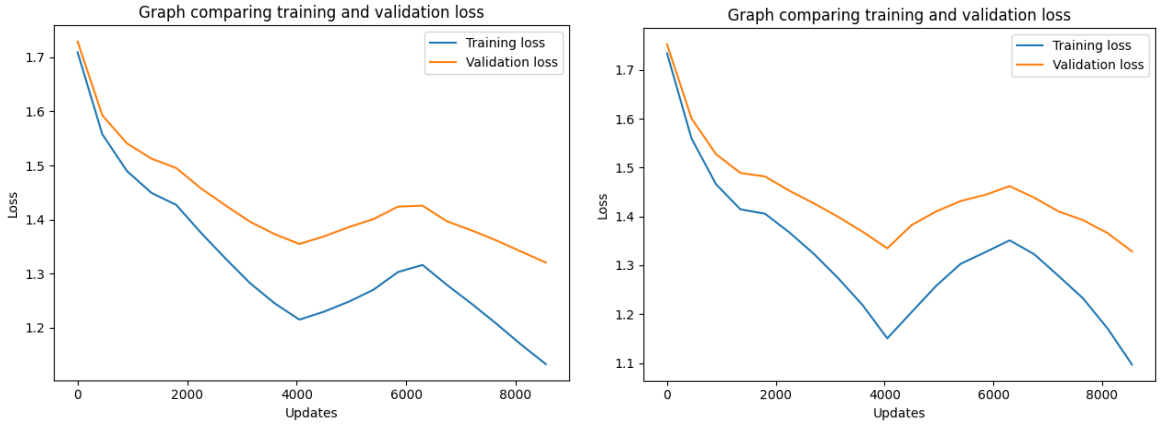

Figure 1: On the left the loss graph is visualized with batch normalization, and to the right it is visualized without batch normalization. This network has three layers.

3.2.2. 9-layer network

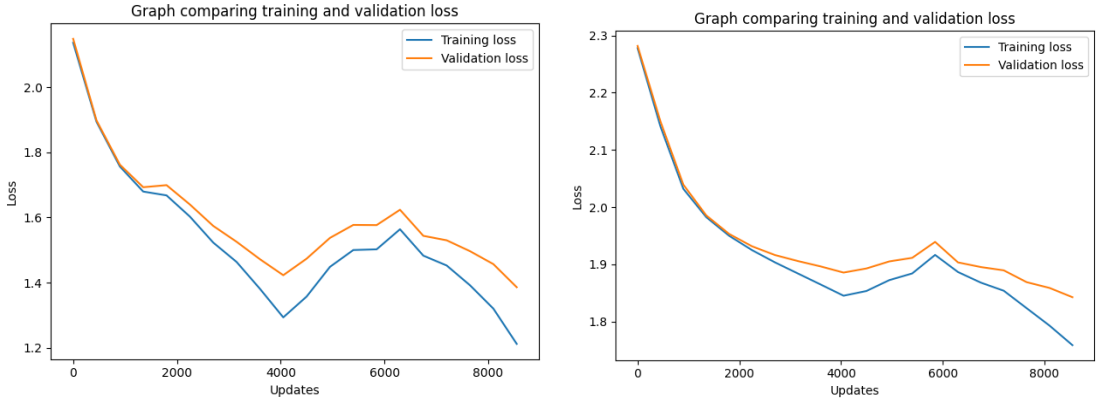

Figure 2: On the left the loss graph is visualized with batch normalization, and to the right it is visualized without batch normalization. This network has 9-layers.

3.3. Coarse search

The coarse search was conducted with values ranging from 10^-5 to 10^-1 for the lambda parameter. The training was done with 2 cycles and the best values on lambda was the following:

Lambda = 8.01666767e-05	Accuracy=0.503
Lambda= 3.91618605e-05	Accuracy=0.507
Lamda= 1.11736111e-05	Accuracy=0.521

3.4. Fine search

Based on the coarse search, the fine search was conducted with values ranging between 10^-3 and 10^-2. The number of cycles were increased to 3 and the best accuracies were achieved with the following values:

Lambda = 0.00466368	Accuracy=0.5366
Lambda= 0.00597247	Accuracy=0.537
Lambda= 0.00723267	Accuracy=0.539

3.5 Final search

In the final version the best lambda value from the fine search, 0.00723267, were used to evaluate the model. In this final round the cycles were increased to 4, and the following results were achieved on the test dataset:

Lambda	Accuracy
0.00723267	0.5344

3.6 Sensitivity to initialization

In this experiment the weight initializations were varied to evaluate its effect on the loss function on a nine-layer network. The same lambda was used that were found in the fine search, and the model was trained for two cycles.

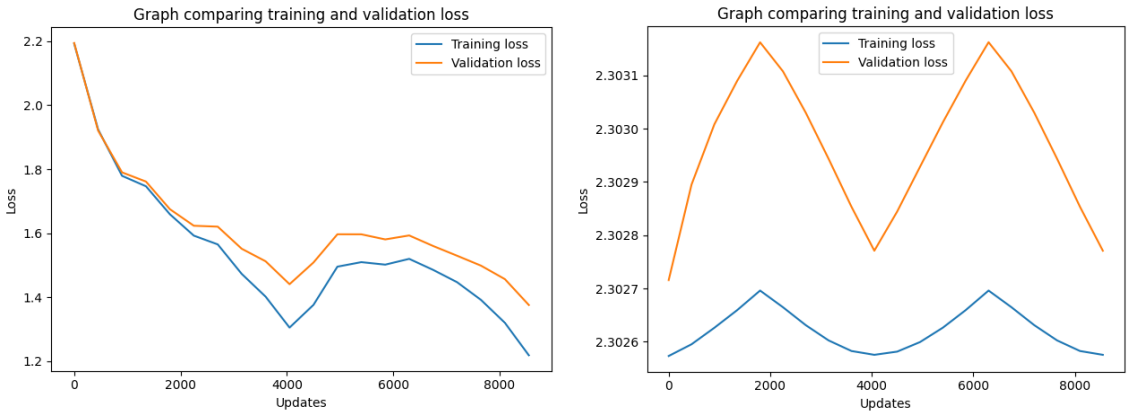

Figure 3: On the left the loss graph is visualized with batch normalization, and to the right it is visualized without batch normalization. Sigma is set to 0.1.

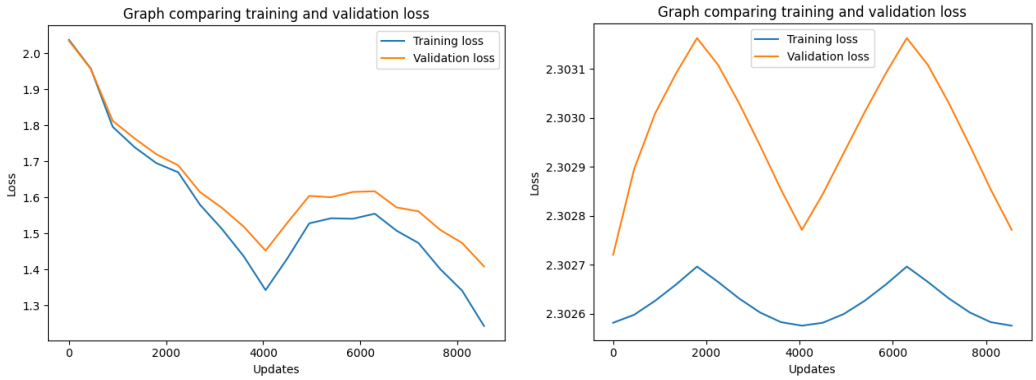

Figure 4: On the left the loss graph is visualized with batch normalization, and to the right it is visualized without batch normalization. Sigma is set to 0.01

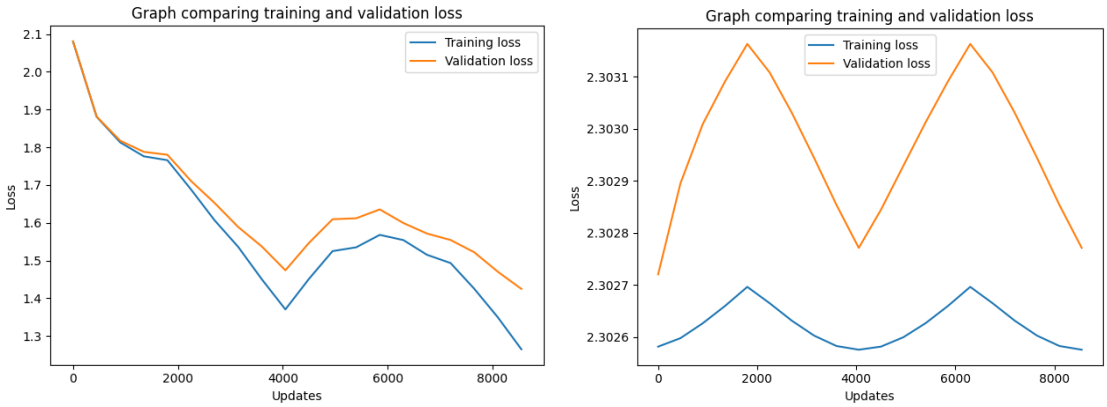

Figure 5: On the left the loss graph is visualized with batch normalization, and to the right it is visualized without batch normalization. Sigma is set to 0.001

By observing the graphs, it is clear that by implementing batch normalization the network becomes less sensitive towards weight initialization. The graphs describing the loss without normalization is rather constant with a high loss value, meaning the model is not learning properly, while the graphs with batch normalization has a loss that is decreasing cyclically.