DD2424 Deep Learning in Data Science: Assignment 3

Romain Trost

May 2022

The task for this assignment consisted of building and training a multi-layer layer network with multiple output labels. The network was trained using mini-batch gradient descent with batch normalisation to classify images contained in the CIFAR-10 dataset.

Analytical and numerical gradients
To make sure that the analytically computed gradients were accurate, the results were compared to those computed numerically over small batches. More specifically, the absolute error difference between both computations was measured. Two tests were implemented to make sure this absolute error was low enough. Firstly, the percentage of gradients for which the absolute error was below the threshold of $1e-6$ was recorded. Secondly, the maximum absolute error was recorded. The results are below.

2 layer network with 50 hidden nodes where only the first 10 dimensions were measured using no regularization:

```
Percentage of weight errors < 1e-6: [100.0, 99.6]
Percentage of bias errors < 1e-6: [100.0, 100.0]
Max weight error: [3.1077043632921075e-08, 1.3130278271322737e-06]
Max bias error: [2.0072670636750445e-08, 8.831480199922304e-08]
```

Figure 1: 2 layer gradient error discrepancy.

3 layer network with [50, 50] hidden nodes where only the first 10 dimensions were measured using no regularization:

```
Percentage of weight errors < 1e-6: [100.0, 100.0, 100.0]
Percentage of bias errors < 1e-6: [100.0, 100.0, 100.0]
Max weight error: [2.1835888958099403e-08, 2.3035743212940218e-07, 3.1549580603584815e-07]
Max bias error: [1.723493928129649e-08, 2.552848244796535e-08, 7.017868716707731e-08]
```

Figure 2: 3 layer gradient error discrepancy.

4 layer network with [50, 50, 10] hidden nodes where only the first 10 dimensions were measured using no regularization:

```
Percentage of weight errors < 1e-6: [100.0, 100.0, 100.0, 100.0]
Percentage of bias errors < 1e-6: [100.0, 100.0, 100.0, 100.0]
Max weight error: [5.4356530554411364e-08, 6.34607092209194e-07, 5.68887205787405e-07, 5.398578817317201e-07]
Max bias error: [4.850460477712204e-08, 5.495565486013376e-08, 1.226303896206815e-07, 1.0257734145424635e-07]
```

Figure 3: 4 layer gradient error discrepancy.

As shown, the error discrepancy between the analytical and numerical gradients is very low allowing us to assume that the analytical computations are bug free.

Training a 3 layer network with and without batch normalisation
Here, a 3 layer network with hidden nodes [50, 50] is trained with cyclical learning rates. The learning curves when using both batch normalisation and no batch normalisation are displayed below. A test accuracy of 52.2% is achieved without batch normalisation which is slightly below the 53.5% accuracy achieved when using batch normalisation.

Figure 4: Loss and accuracy performance without batch normalisation and parameters **eta_min = 1e-5**, **eta_max = 1e-1**, **lambda = 0.005**, $n\_s = 5 * 45000/n\_batch$
.

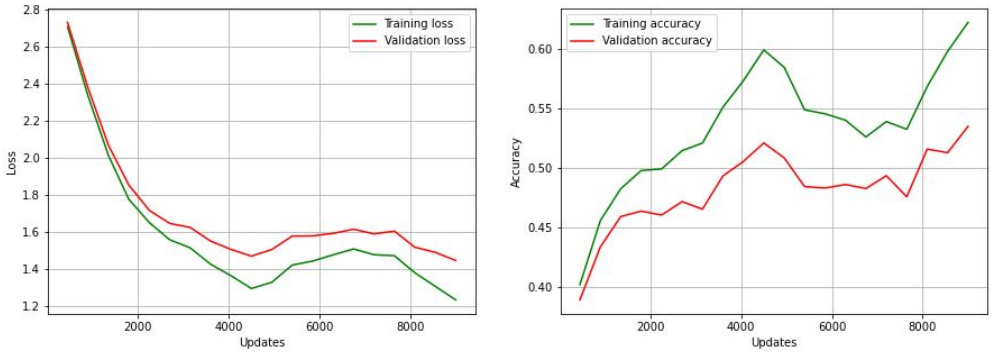


Figure 5: Loss and accuracy performance with batch normalisation and parameters **eta_min = 1e-5**, **eta_max = 1e-1**, **lambda = 0.005**, $n\_s = 5 * 45000/n\_batch$
.

Training a 9 layer network with and without batch normalisation
Here, a 9 layer network with hidden nodes [50, 30, 20, 20, 10, 10, 10, 10] is trained with cyclical learning. The learning curves when using both batch normalisation and no batch normalisation are displayed below. A test accuracy of 49.8% is achieved without batch normalisation which is slightly below the 51.6% accuracy achieved when using batch normalisation.

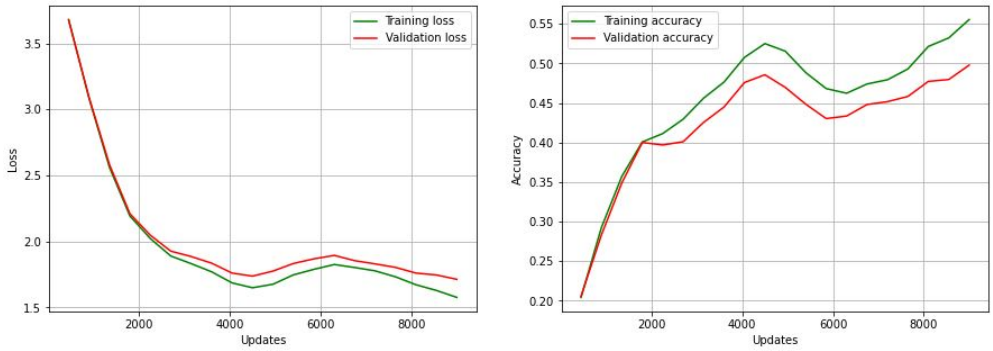


Figure 6: Loss and accuracy performance without batch normalisation and parameters **eta_min = 1e-5**, **eta_max = 1e-1**, **lambda = 0.005**, $\boldsymbol{n\_s = 5 * 45000/n\_batch}$
.

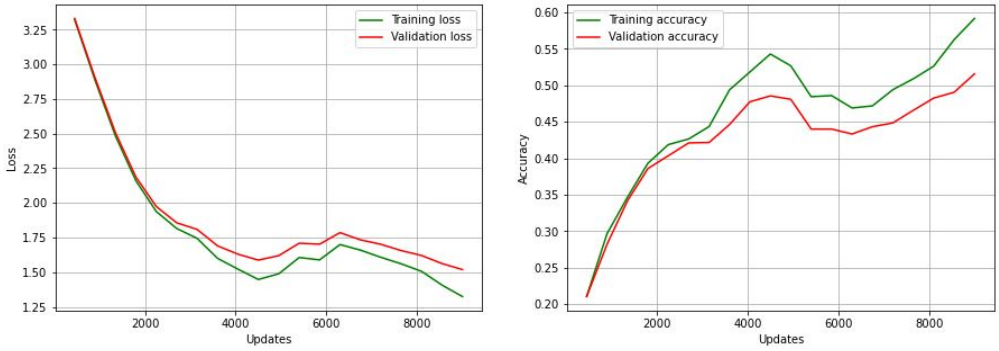


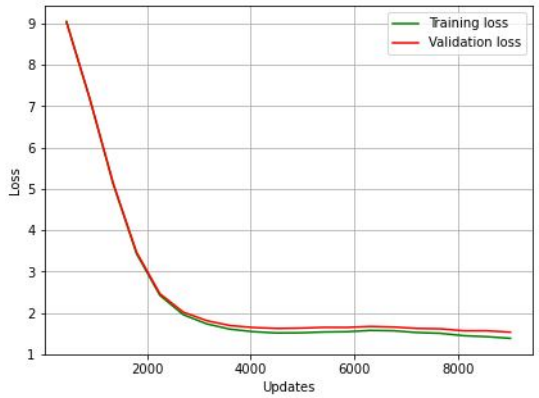
Figure 7: Loss and accuracy performance with batch normalisation and parameters **eta_min = 1e-5**, **eta_max = 1e-1**, **lambda = 0.005**, $\boldsymbol{n\_s = 5 * 45000/n\_batch}$
.

Coarse-to-fine search to find a good value for λ
Here, coarse-to-fine search was applied to find an optimal value of λ to train a 3 layer network with batch normalisation. Ten different random values of λ within the range of $1e-1$ and $1e-4$ were tested. The best performing network achieved a test accuracy of 53.7% with a λ value of $3.83e-3$.

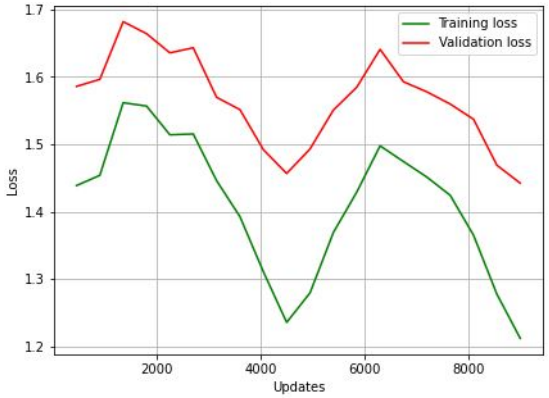
Sensitivity to initialization experiment

In this experiment, the weights of the neural network are initialised with zero mean and a select amount of different variances (1e-1, 1e-3, 1e-4). The parameters of the network are the

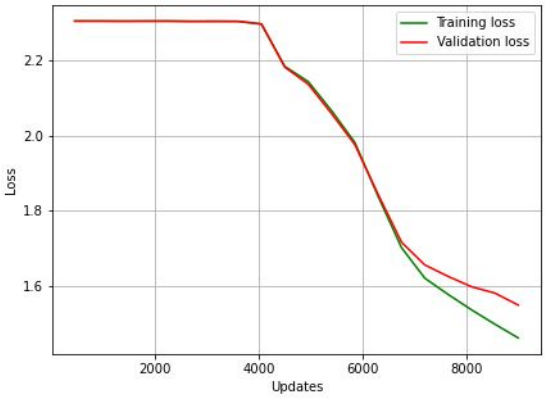
following: **eta_min = 1e-5**, **eta_max = 1e-1**, **lambda = 0.005**, $\boldsymbol{n\_s = 5 * 45000/n\_batch}$. Below is displayed the loss plots when using both batch normalisation and no batch normalisation over 2 cycles.



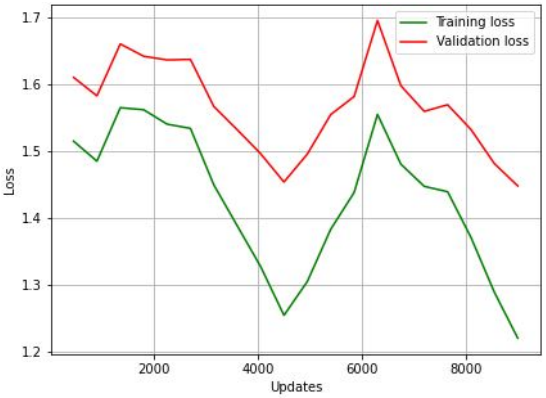
(a) sigma = 1e-1, no batch normalization



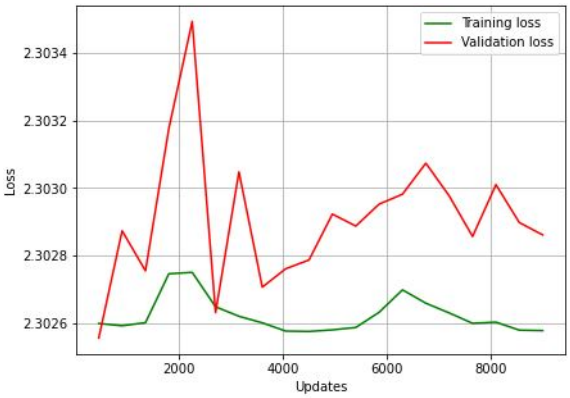
(b) sigma = 1e-1, batch normalization



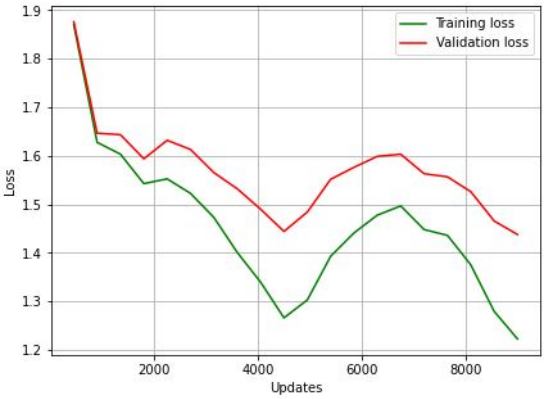
(a) sigma = 1e-3, no batch normalization



(b) sigma = 1e-3, batch normalization



(a) sigma = 1e-4, no batch normalization



(b) sigma = 1e-4, batch normalization

In general, these plots show that when no batch normalisation is used, a smaller sigma causes a higher loss. However, when using batch normalisation, the performance is improved, as the resulting loss is lower than without using batch normalisation.