DD2424 Deep Learning in Data Science: Assignment 1

Romain Trost

April 2022

The task for this assignment consisted of building and training a two layer network with multiple output labels. The network was trained using mini-batch gradient descent and cyclical learning rates 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 (20 first samples). 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.

```
Percentage of weight 1 errors < 1e-6: 100.0
Percentage of weight 2 errors < 1e-6: 100.0
Percentage of bias 1 errors < 1e-6: 100.0
Percentage of bias 2 errors < 1e-6: 100.0
Max weight 1 error: 2.1387933624161803e-07
Max weight 2 error: 3.803785206951682e-09
Max bias 1 error: 1.581129628058786e-07
Max bias 2 error: 4.7170880906888257e-07
```

Figure 1: Error between analytically and numerically computed gradients.

Performance using cyclical learning rates
To train the network, cyclical learning rates were implemented. Two different network configurations were deployed with the learning curves for these displayed below.

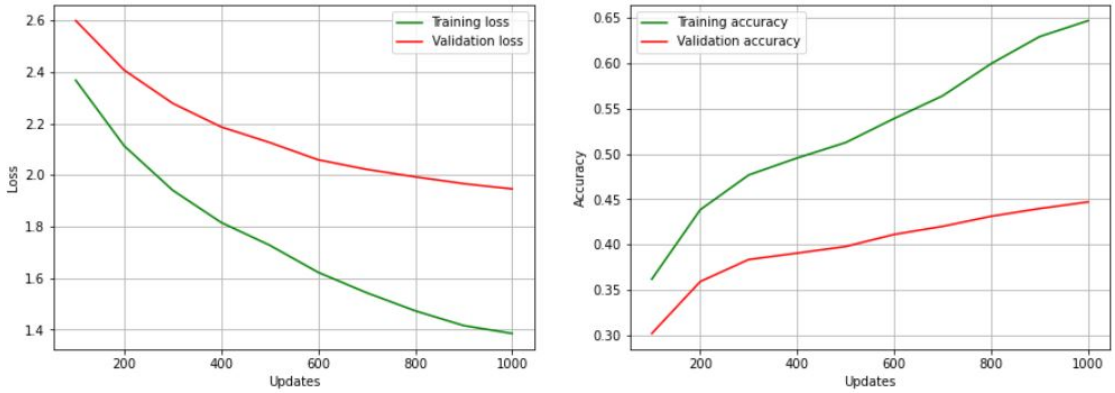

Figure 2: Loss and accuracy performance with **eta_min = 1e-5**, **eta_max = 1e-1**, **lambda = 0.01**, **n_s = 500**. Since we're using a batch size of 100 and the network is trained for one cycle, 10 epochs are required to fully train the network.

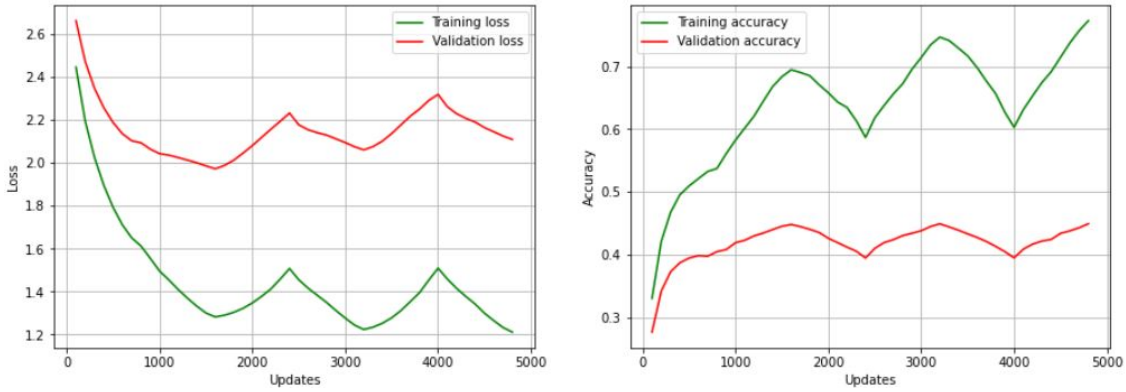

Figure 3: Loss and accuracy performance with **eta_min = 1e-5**, **eta_max = 1e-1**, **lambda = 0.01**, **n_s = 800**. Since we're using a batch size of 100 and the network is trained for 3 cycle, 48 epochs are required to fully train the network.

As shown, using more cycles gives more wavy loss and accuracy curves. These triangular patterns are due to the learning rate linearly increasing and then decreasing once every cycle. The first configuration yields a validation accuracy of 44.71% whereas the second configuration sees a small improvement with a validation accuracy of 44.94%.

Course search to set lambda
To find the best possible lambda for the network, a course search was first implemented, where 45000 images were used for training and 5000 for validation. Ten different initial values of lambda within the range of 1.707^{-5} and 0.635 were tested. The network was then trained for 2 cycles with each one of these values for 5 different initializations. The validation accuracies were then averaged to get the most indicative validation accuracy for each lambda. The results are the following:

- Highest validation accuracies: [0.5161, 0.5154, 0.5151]
- Respective lambdas: [2.481^{-3}, 4.208^{-5}, 4.207^{-5}]

Fine search to set lambda
For the fine search, the range of lambdas was narrowed down to be between the range 0 and 0.004 (5 values were tested in total). The exact same training approach as for the coarse search was implemented, resulting in the following:

- Highest validation accuracies: [0.5169, 0.5152, 0.5136]
- Respective lambdas: [0.001, 0.002, 0.003]

As shown, the fine search allowed us to find the best lambda = 0.001 which yields a higher validation accuracy (0.5169) than any of the lambdas used in the coarse search.

Training the network with the best lambda
Finally, the network is trained using the best lambda. For this training, 49000 of the images are used for training and 1000 for validation. The learning curves are displayed below.

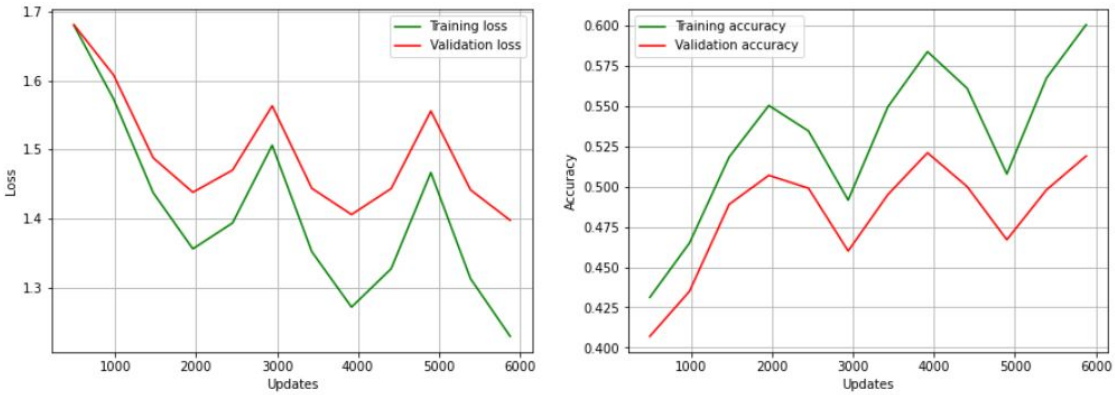

Figure 4: Learning curves when the network is trained with the best lambda

As expected, since we are using more training data and an optimal lambda, the validation accuracy sees an improvement to 51.9%.