DD2424 Deep Learning in Data Science

Assignment 3 - Lab report, Valdemar Gezelius (vgez@kth.se)

January 31, 2023

1 About the lab

The lab focused on to what extent we can train a neural network, using mini-batch gradient descent, to be able to classify pictures. The data-set came in batches of `10000` data-points with `3072` features each, divided into `10` unique classes. Apart from the code I implemented for `Lab 2`, `Batch Normalization` functionality was added to the network. The task was accomplished using `Python3` with `NumPy` for mathematical computations and `Matplotlib` for data visualization.

2 Gradient examinations

With the value for `h` set to `1e-5`, the provided numerical gradient function updated to handle multiple layers, could be used to compute gradients to compare with the result of the analytical gradients function that is used in the network. I then used the built-in Python library `Unittest` to compare my results. First comparing the four gradients ($\texttt{grad}_{\texttt{weights}}$, $\texttt{grad}_{\texttt{bias}}$, $\texttt{grad}_{\texttt{gamma}}$ and $\texttt{grad}_{\texttt{beta}}$), directly using the built-in function $\texttt{assertAlmostEqual}(\texttt{g}_{\texttt{a}}, \texttt{g}_{\texttt{n}})$, where I measure that the analytically computed gradient and the numerical does not differ down to the seventh decimal place. Secondly by implementing `Eq.1`, where the relative error between the numerical gradient $\texttt{g}_{\texttt{n}}$ and the analytical gradient $\texttt{g}_{\texttt{a}}$, is computed (here `eps` is a very small positive number)

$$\frac{|g_a - g_n|}{max(eps, |g_a| + |g_n|)} \tag{1}$$

The function I then used was the built-in `assertLessEqual(g, num)`, that return `true iff g is less than or equal to num` (where `num` is a small number, for testing I set this to `1e-5`). When the network is shallow (2 layers), `1e-5` works as threshold, but when the network gets deeper (3+ layers) I needed to increase the threshold to `1e-4` and even `1e-3`. This shows that the discrepancy between the analytic and the numerical gradients increases with depth. The first `20 features` were selected, `batch_size 5`, and layer size `50 nodes`. Given these results I felt confident that I could move forward to the next part of the lab.

3 3-layer vs 9-layer Networks

When training shallower networks, like the 3-layer configuration, the implementation of batch normalization did not make a significant difference. I tried several parameter initialization, shuffling the training data for each epoch, and the network with batch normalization yielded slightly higher accuracy, however, it can be argued that the two networks performed similarly. Conversely, when training deeper networks like a 9-layer configuration, the implementation of batch normalization significantly improved accuracy, while the absence of batch normalization led to a decline in accuracy. This trend is even more pronounced when examining even deeper networks with a consistent number of nodes, where networks without batch normalization experienced a significant decrease in accuracy, while those with batch normalization showed significantly less decline.

3.1 3-layer Network

3.1.1 With batch normalization

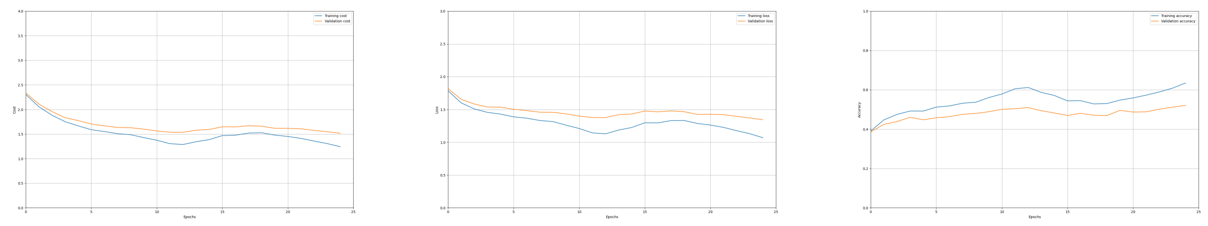

Figure 1: 3-layer network with batch normalization, 50 nodes in each hidden layer. **Left:** cost plot. **Middle:** loss plot. **Right:** accuracy plot. Two cycle ($\texttt{24 epochs}$) of training performed. Network settings: $\eta_{\texttt{min}} = \texttt{1e}-\texttt{5}, \eta_{\texttt{max}} = \texttt{1e}-\texttt{1}, \lambda = \texttt{0.005}, \texttt{batch\_size} = \texttt{100}$. The network yielded a test accuracy of $\approx \texttt{52.44}\%$.

3.1.2 Without batch normalization

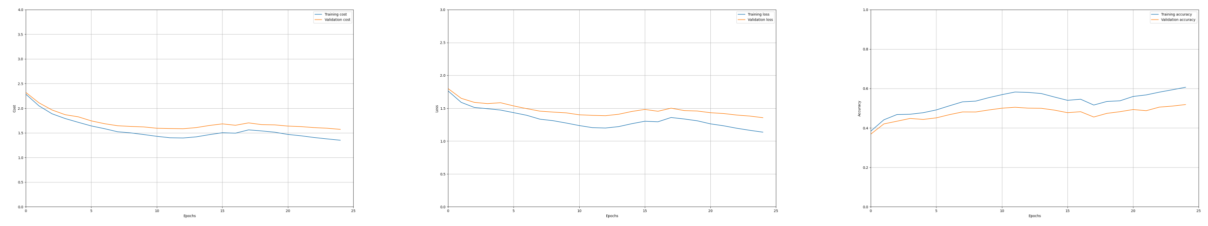

Figure 2: 3-layer network without batch normalization, 50 nodes in each hidden layer. **Left:** cost plot. **Middle:** loss plot. **Right:** accuracy plot. Two cycle ($\texttt{24 epochs}$) of training performed. Network settings: $\eta_{\texttt{min}} = \texttt{1e}-\texttt{5}, \eta_{\texttt{max}} = \texttt{1e}-\texttt{1}, \lambda = \texttt{0.005}, \texttt{batch\_size} = \texttt{100}$. The network yielded a test accuracy of $\approx \texttt{51.96}\%$.

3.2 9-layer Network

3.2.1 With batch normalization

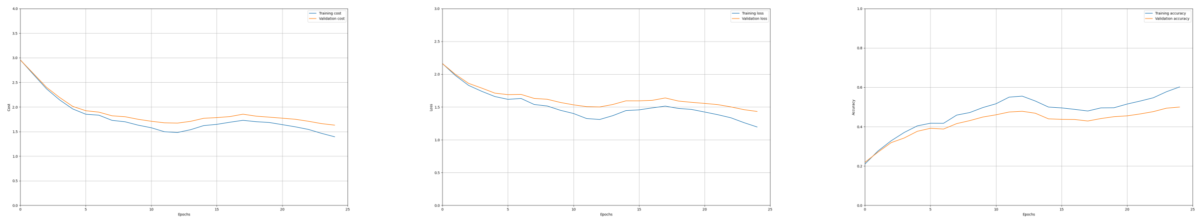

Figure 3: 9-layer network with batch normalization. Hidden Layer structure [50, 30, 20, 20, 10, 10, 10] **Left:** cost plot. **Middle:** loss plot. **Right:** accuracy plot. Two cycle (`24 epochs`) of training performed. Network settings: $\eta_{\texttt{min}} = \texttt{1e} - \texttt{5}, \eta_{\texttt{max}} = \texttt{1e} - \texttt{1}, \lambda = \texttt{0.005}, \texttt{batch\_size} = \texttt{100}$. The network yielded a test accuracy of $\approx \texttt{50.75}\%$.

3.2.2 Without batch normalization

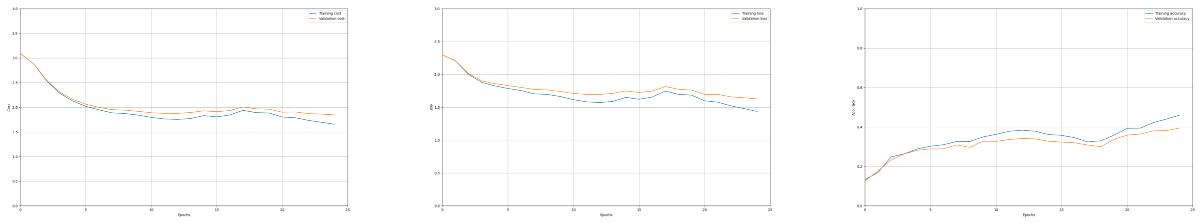

Figure 4: 9-layer network without batch normalization. Hidden Layer structure [50, 30, 20, 20, 10, 10, 10] **Left:** cost plot. **Middle:** loss plot. **Right:** accuracy plot. Two cycle (`24 epochs`) of training performed. Network settings: $\eta_{\texttt{min}} = \texttt{1e} - \texttt{5}, \eta_{\texttt{max}} = \texttt{1e} - \texttt{1}, \lambda = \texttt{0.005}, \texttt{batch\_size} = \texttt{100}$. The network yielded a test accuracy of $\approx \texttt{40.79}\%$.

4 Lambda optimization

Iteration	λ-value	Validation accuracy
3	0.009437	0.514879
2	0.002478	0.512378
7	0.018624	0.504376
9	0.020749	0.502876
10	0.014437	0.498375
1	0.031645	0.490623
4	0.040199	0.490123
8	0.032798	0.489872
5	0.038753	0.484871
6	0.044138	0.479870

Table 1: Coarse search for λ-value.

Iteration	λ-value	Validation accuracy
14	**0.003527**	0.520880
16	0.004391	0.514629
20	0.004240	0.512878
13	0.005397	0.512128
11	0.008290	0.512128
15	0.008363	0.510378
18	0.002996	0.507377
12	0.004074	0.506127
17	0.006934	0.504626
19	0.004151	0.504626

Table 2: Fine search for λ-value.

A coarse-to-fine search for a optimized λ-value was conducted. Each network trained for 2 cycles with additional network settings (apart from $\lambda -$ value): batch_size=100, n_s=5$*(n/n_{batch})$, $\eta_{min} = 1e-5$, $\eta_{max} = 1e-1$, batch normalization and no shuffle.

My approach was to run 20 iterations, training each new network with a different λ-value sampled uniformly from a set range of values. After 10 iterations I evaluated the results (compiled in Table 1) and updated the range for the λ-values. The start range for the λ-values was (l_max=0.05, l_min=0.001), and after the evaluation, the range was updated to (l_max=0.009437 +eps, l_min=0.002478 -eps). I decided to add/remove a small number (eps=0.0001) to prevent the range to be too narrow since I automated the search and there could be a case where the range became to narrow for the two networks that yielded the best validation accuracy. As we see in Table 3 we overall higher accuracy with λ-values that were sampled from the refined search range. We can see that the network trained in the 14th iteration with λ=0.003527 yielded a validation accuracy of $\approx$ 52.1%, highest in the test and thus was chosen for the final network training.

5 Training best network

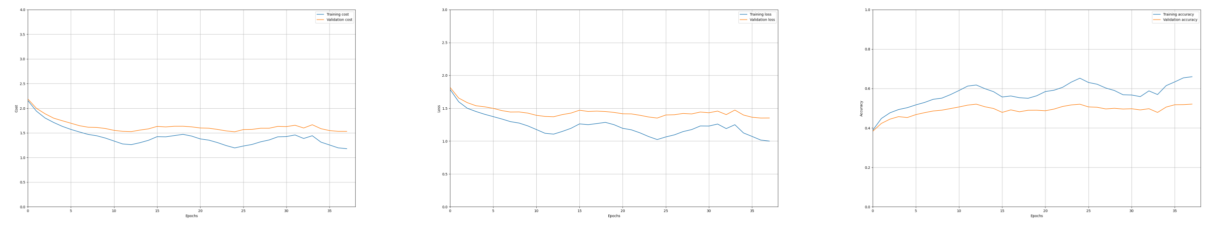

Figure 5: Best performing network, training- and validation plots. **Left:** cost plot. **Middle:** loss plot. **Right:** accuracy plot. Three cycle (`36 epochs`) of training performed. Parameter settings: η_{min}`=1e-5,` η_{max}`=1e-1, n_s=1200,` λ`=0.003038, batch_size=100`. The network yielded a test accuracy of ≈`52.5%`.

With the λ`-value` from the coarse-to-fine search, a best network was trained. The preferences for the network and training was: batch_size=100, n_s=5 $*$ $(n/batch\_size)$, η_{min}=1e-5, η_{max}=1e-1, `batch normalization` and `shuffle`. The network was trained for `3` cycles and yielded a test accuracy of ≈`52.5%`. `45000` samples was used for training and `5000` samples for validation, testing was done on the `test_batch`.

6 Sensitivity to Initialization - Experimental Findings

Network	Batch norm	Sigma value	Val. Accuracy	Test Accuracy
3-layer	True	0.1	0.5206	0.5167
3-layer	True	0.001	0.5314	0.5267
3-layer	True	0.0001	0.5229	0.5257
3-layer	False	0.1	0.5154	0.5261
3-layer	False	0.001	0.1038	0.1
3-layer	False	0.0001	0.5206	0.1

Table 3: Experimental findings regarding sensitivity in initialization of the networks weights.

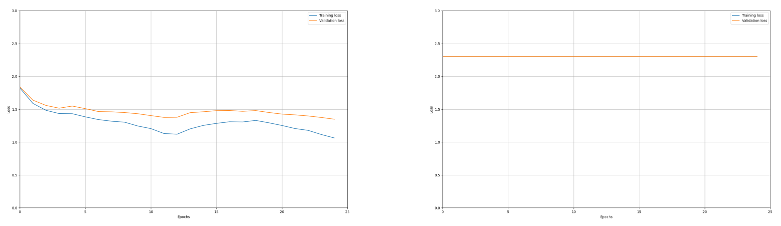

Figure 6: **Left:** loss plot with batch normalization. **Right:** loss plot without batch normalization.

Several experiments were performed on the proposed 3-layer network, combining different sigma values and batch norm options. The results, displayed in `Table 3`, indicate that the network failed to converge with small sigma values without batch norm. This was unexpected as the hypothesis was that the network without batch norm would not converge for any sigma value. Further experiments were run on the 9-layer network, where the configuration without batch norm and a sigma of `sigma of 0.1` resulted in poor performance with a `validation accuracy of 0.1 and a test accuracy of 0.095`. The results suggest that deep and shallow networks are more susceptible to initialization issues when batch normalization is not utilized. Batch normalization compensates for improper weight matrix initialization to some extent, but it's still recommended to use reliable initialization methods like Xavier or He, even with batch normalization.

The loss plots in `Figure 6` demonstrate the distinction between two networks that employ the same sigma value, one with the implementation of batch normalization and the other without.