Assignment 3

Fred Lindahl (frlinda@kth.se), 950213-7772
Deep Learning in Data Science, DD2424

I. Checking gradients

I checked my analytic gradient computations by comparing the values to the numerical gradient computation provided for the lab using the standard formula for the course

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

with $\text{eps} = 10^{-10}$.

I performed the calculation above for the gradients of the weights, biases, stretch and shift, for each layer of networks with up to 4 hidden layers using 3 images and 20 dimensions of the images. I debugged my code and moved on to the next task when all errors were $\sim 10^{-10}$.

II. Loss function for 3-layer network

Using the given default parameters settings with Xavier initialization I trained a 3-layer network with [50 50] nodes in the hidden layers. The evolution of the loss function for both cases are shown in the figure below.

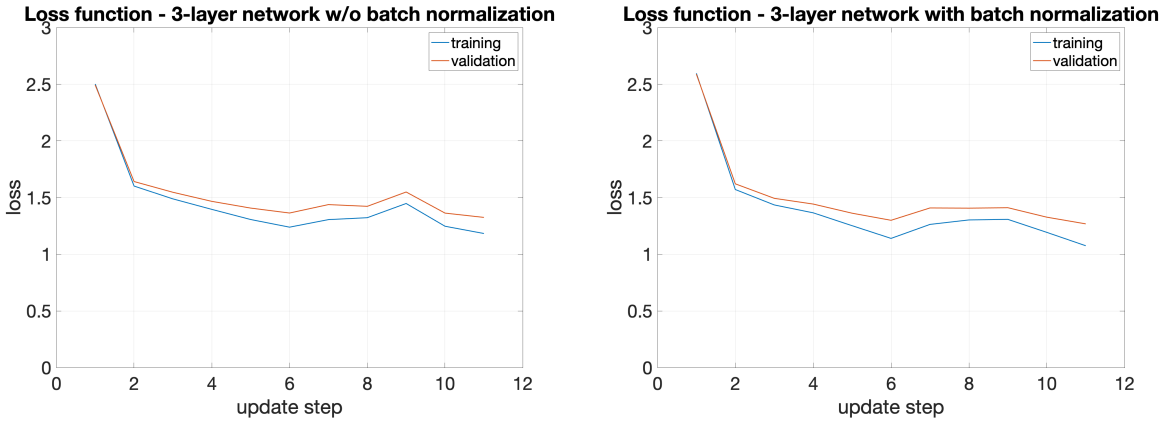

Figure 1: Evolution of loss function for 3-layer network with and without batch normalization

53.06% without batch normalization, 54.03% with batch normalization.

III. Loss function for 9-layer network

Same settings as with the 3-layer network and $[50, 30, 20, 20, 10, 10, 10, 10]$ nodes in the hidden layers. The evolution of the loss function for both cases are shown in the figure below.

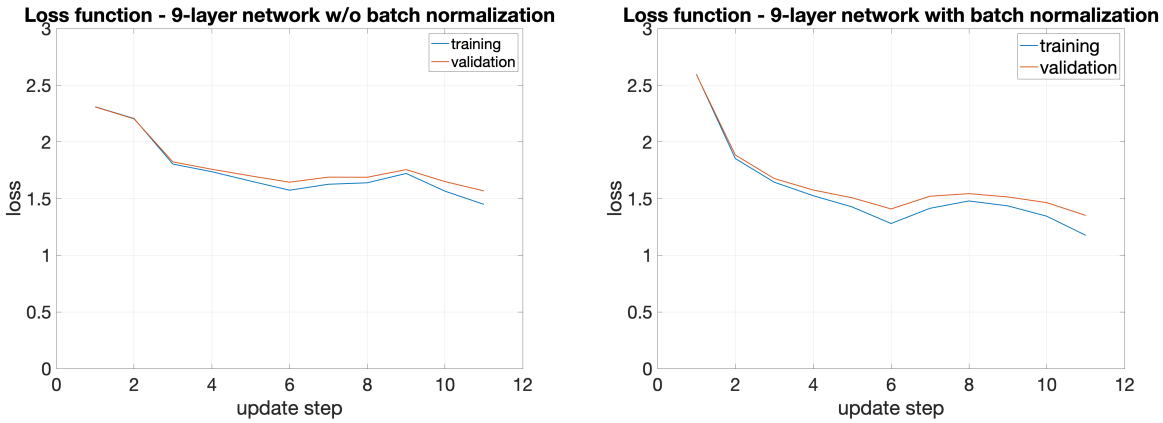

Figure 2: Evolution of loss function for 9-layer network with and without batch normalization

42.03% without batch normalization and 51.42% accuracy with batch normalization. Clearly batch normalization is much more important for larger networks.

IV. Coarse-to-fine random search to find optimal value of `lambda`

For the coarse search for lambda I used 8 log-spaced values between 10^{-5} and 10^{-1}, He initialization and keeping the other parameters the same as inthe sections above. The best lambda found was ~ 0.007. Next, I used 8 log-spaced values between 0.006 and 0.008, which gave me a final value of lambda equal to 0.007072. With this value, using the default parameters and three cycles I achieved a test accuracy of 53.72%.

V. Sensitivity to initialization

Finally, to check the sensitivity of batch normalization I initialize all weights using a Gaussian with $\sigma = 0.1$, 0.001 and 0.0001 at all layers for both the 3-layer and the 9-layer networks. The results are presented in the figures below.

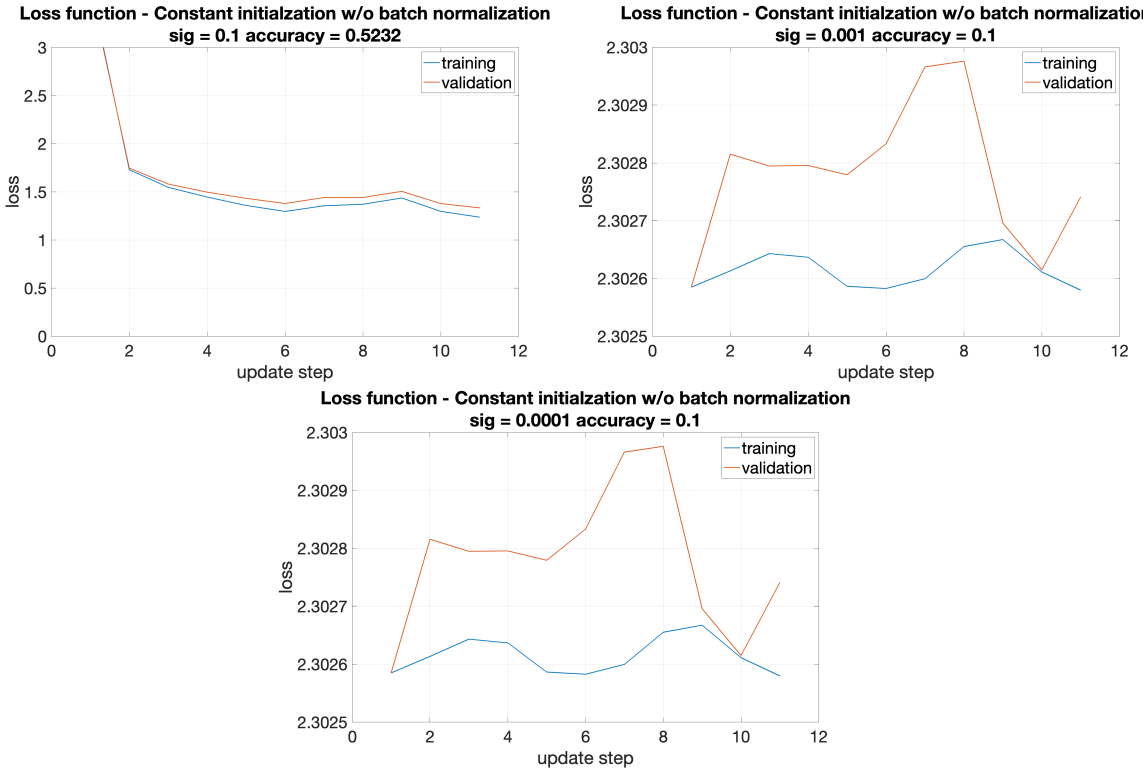

Figure 3: 3-layer networks with constant Gaussian initialization and no batch normalization

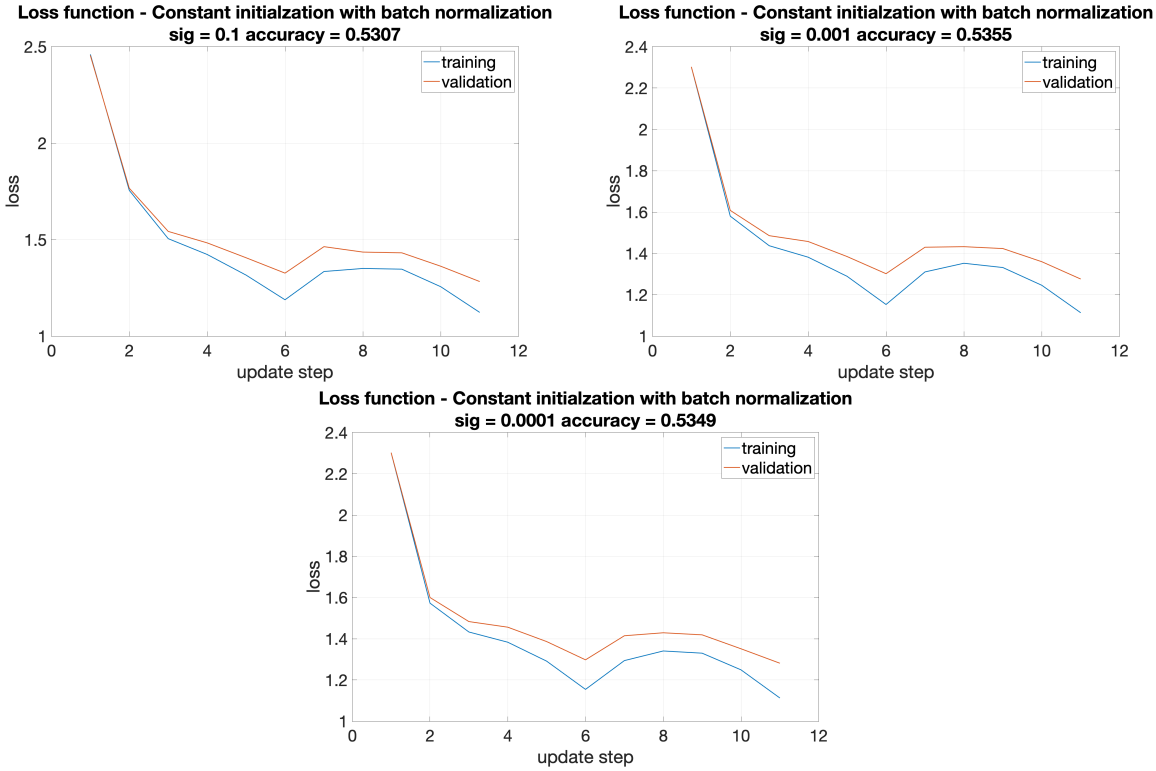

Figure 4: 3-layer networks with constant Gaussian initialization with batch normalization

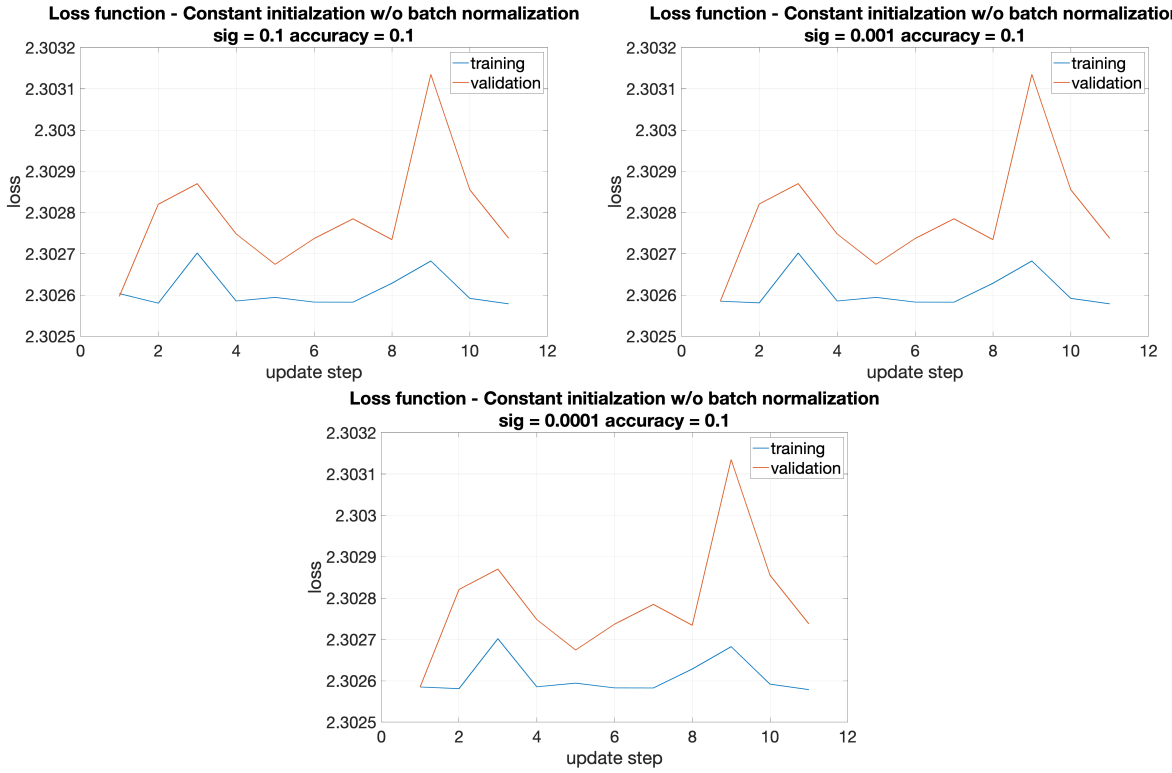

Figure 5: 9-layer networks with constant Gaussian initialization and no batch normalization

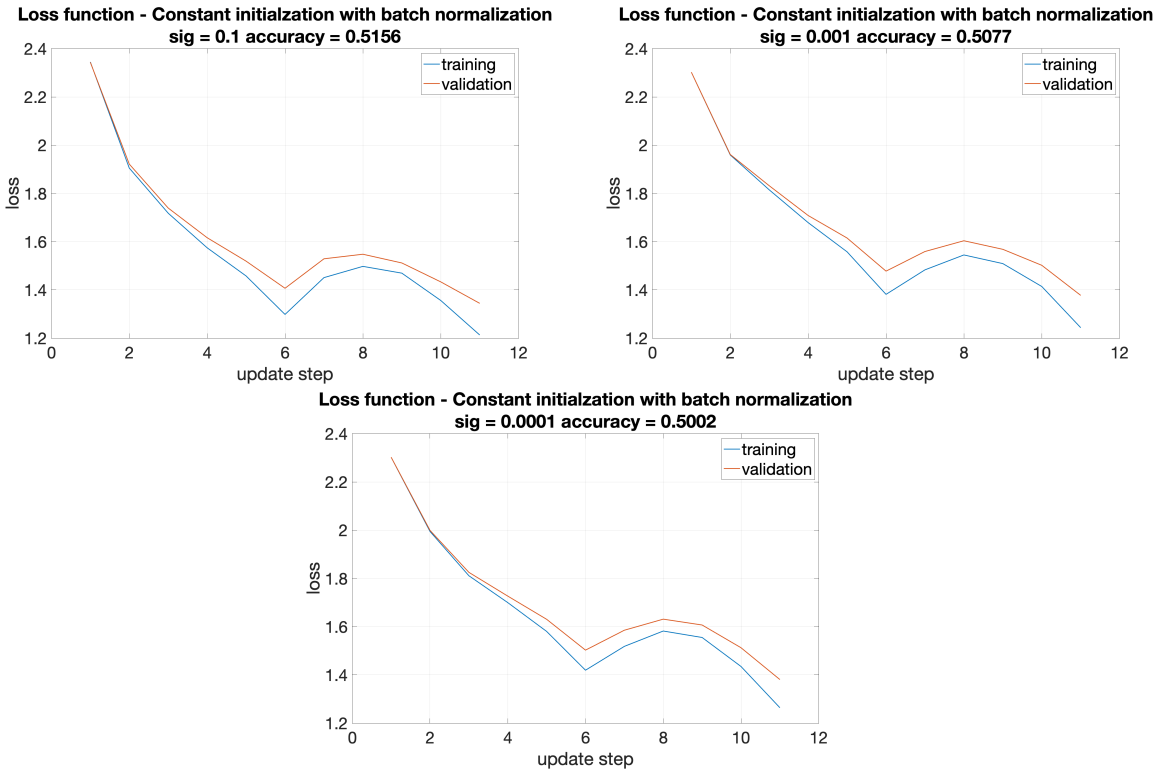

Figure 6: 9-layer networks with constant Gaussian initialization with batch normalization

All figures show clear improvements after applying batch normalization. Without it, the final test

accuracy is just 10% and then up to $> 50\%$ with it, i.e. from useless to comparable performance to that of careful initialization. This shows the usefulness of batch normalization.