Deep Learning: Assignment 3

Anton Stråhle

April 25, 2020

Introduction

In this assignment we have constructed, and trained, a k-layer fully connected network with, and without batch-normalization, using data from CIFAR-10.

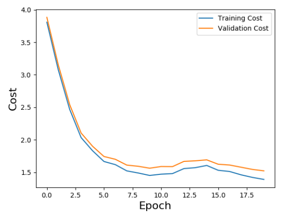
Gradient Checking

In order to check that my analytical gradients were correct, which I believe they are, I translated the given Matlab function `ComputeGradsNumSlow` to Python and then compared my analytcial gradients using the function `np.testing.assert_array_almost_equal` at 6 decimal places. For the 3-layer network without batch normalization this function yielded that the numerical and analytical gradients were equal up to 6 decimal places. When computing the gradients for the 9-layer network without batch normalization the gradients, as expected, start to vanish meaning that we can't learn, and as a result need batch normalization.

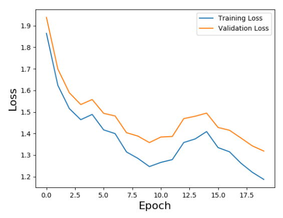
For the 9-layer network using batch normalization the initial layers turned out to equal at several decimal places when using the aforementioned function. However, the more we propagated the gradients backwards the more they started to differ from their analytical counterparts, as was mentioned in the assignment. This seems to be a natural occurrence and not a fault of my gradient implementation as the accuracies that were obtained turned out similar to those mentioned in the assignment.

3-Layer Network

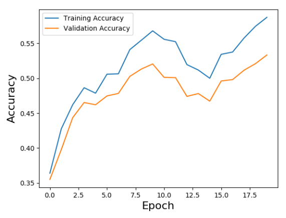
For the 3 layer network without batch normalization I achieved an accuracy of 53% with the following loss, cost and accuracy plots.



(a) Cost



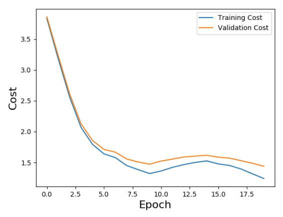
(b) Loss



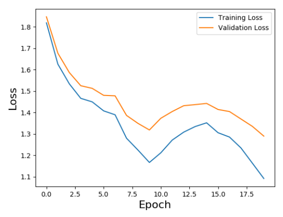
(c) Accuracy

Figure 1: 3 layer network without batch normalization

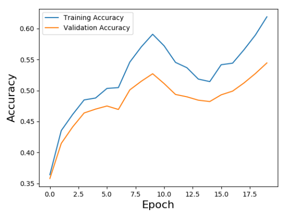
When using batch normalization I achieved an accuracy of 54% with the following loss, cost and accuracy plots.



(a) Cost



(b) Loss

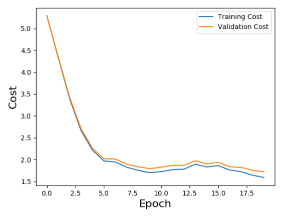


(c) Accuracy

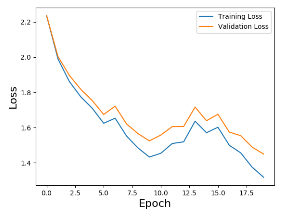
Figure 2: 3 layer network with batch normalization

9-Layer Network

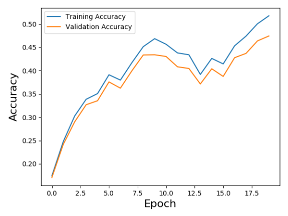
For the 9-layer network without batch normalization I achieved an accuracy of 47% with the following loss, cost and accuracy plots.



(a) Cost



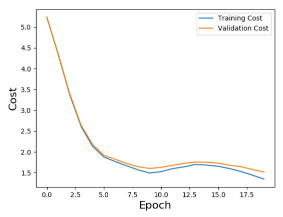
(b) Loss



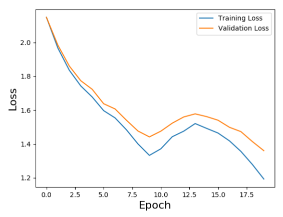
(c) Accuracy

Figure 3: 9 layer network without batch normalization

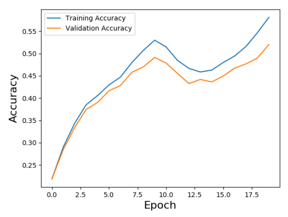
When using batch normalization I achieved an accuracy of 52% with the following loss, cost and accuracy plots.



(a) Cost



(b) Loss



(c) Accuracy

Figure 4: 9 layer network with batch normalization

Coarse-to-Fine Search for λ

I did a coarse search using the same ranges as in Assignment 2

$$\mathrm{U} \sim \mathrm{Unif}(-5, -1)$$
$$\lambda = 10^{\mathrm{U}}$$

It turned out that the highest validation accuracies where generated closer to 10^{-1} than 10^{-5} and as such I narrowed down the search to $\mathrm{U} \sim \mathrm{Unif}(-3, -1)$. The final finer search was done using $\mathrm{U} \sim \mathrm{Unif}(-3, -2)$. Within this interval the accuracies did not vary that much at all and any differences could in some sense be attributed to the differing initializations of the parameters. In this search we did however get a validation accuracy of 55% which was obtained using $\lambda = 0.007671$.

Using $\lambda = 0.007671$ I achieved a test accuracy of about 54% on the 3-layer network when training for 3 cycles but also an accuracy of 52.5% on the 9-layer network when training for 3 cycles.

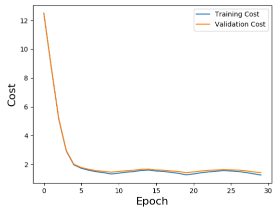
Sensitivity to Initialization

Lastly we want to examine the effects initialization has on the different training regimes. Specifically we want to compared He-initialization, as used throughout this assignment, with a fixed initialization which is independent of the size of each layer, specifically we use

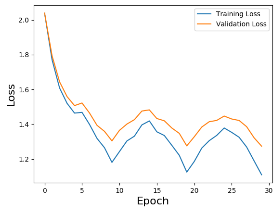
$$\mathbf{W}_i \sim \mathbf{N}(\mathbf{0}, \boldsymbol{\sigma}) \quad i = 1, ..., k$$

where $\boldsymbol{\sigma} = \text{dag}(\sigma, \sigma, ..., \sigma)$ for some fixed value of σ. Specifically we want to test $\sigma = 0.1, 0.001, 0.0001$.

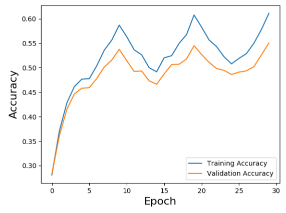
We begin by observing the 3-layer network using batch normalization, λ found in the coarse-to-fine search and $\sigma = 0.1$. This gave us a testing accuracy of about 53% as well as the following plots.



(a) Cost



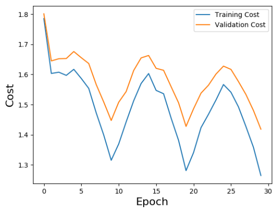
(b) Loss



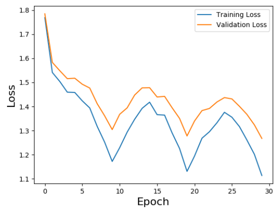
(c) Accuracy

Figure 5: 3 layer network with batch normalization and $\sigma = 0.1$

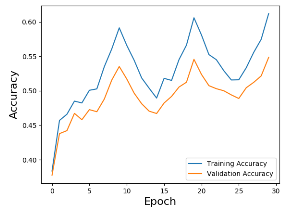
Secondly we observe the same parameters but for $\sigma = 0.001$ which gave us an accuracy of about 53%



(a) Cost



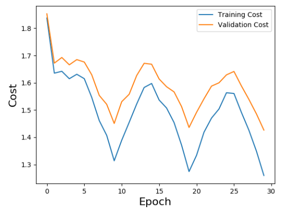
(b) Loss



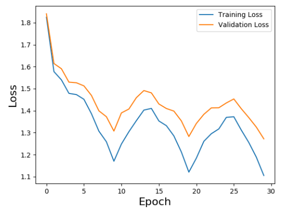
(c) Accuracy

Figure 6: 3 layer network with batch normalization and $\sigma = 0.001$

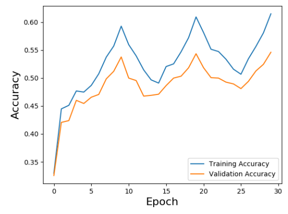
Secondly we observe the same parameters but for $\sigma = 0.0001$ which gave us an accuracy of about 54%



(a) Cost



(b) Loss

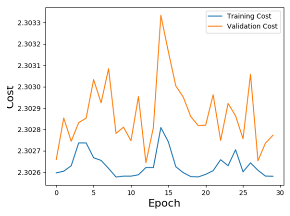


(c) Accuracy

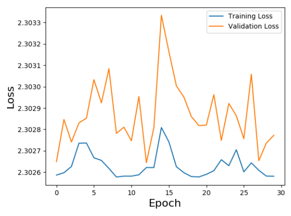
Figure 7: 3 layer network with batch normalization and $\sigma = 0.0001$

Now we want to go through the same procedure again but now without batch normalization.

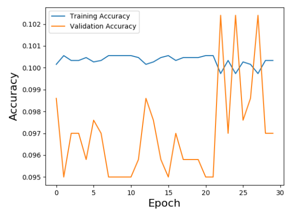
For $\sigma = 0.1$ without batch normalization we got an accuracy of 10% and the following plots.



(a) Cost



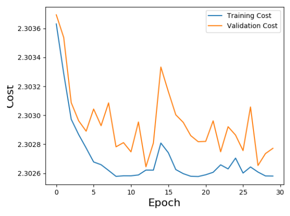
(b) Loss



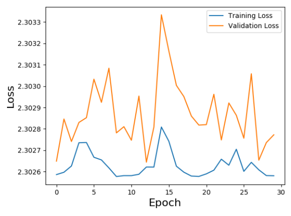
(c) Accuracy

Figure 8: 3 layer network without batch normalization and $\sigma = 0.1$

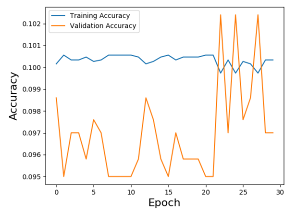
For $\sigma = 0.001$ without batch normalization we got an accuracy of 10% and the following plots.



(a) Cost



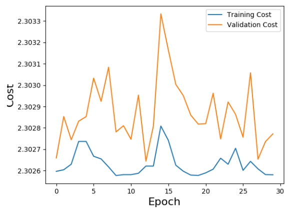
(b) Loss



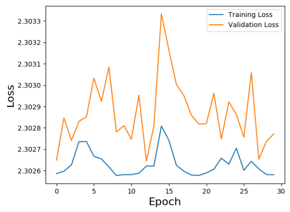
(c) Accuracy

Figure 9: 3 layer network without batch normalization and $\sigma = 0.001$

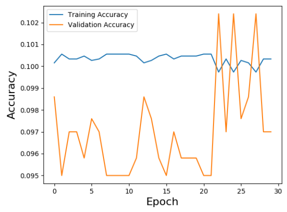
For $\sigma = 0.0001$ without batch normalization we got an accuracy of 10% and the following plots.



(a) Cost



(b) Loss



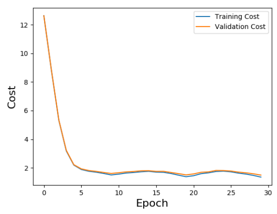
(c) Accuracy

Figure 10: 3 layer network without batch normalization and $\sigma = 0.0001$

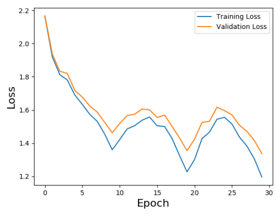
For $\sigma = 0.1, 0.001, 0.0001$ we clearly notice that the lack of batch normalization makes the network more sensitive to initialization as the usage of σ-initialization with the aforementioned values reduces the predictive power of the network to that of random guessing.

As we get a deeper network one would assume that the outcomes would be the same, i.e. that the training will fail without proper initialization when not using batch normalization. To verify this we examine the 9 layer network as well.

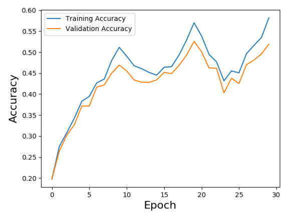
For $\sigma = 0.1$ we achieved an accuracy of 52% and got the following plots.



(a) Cost



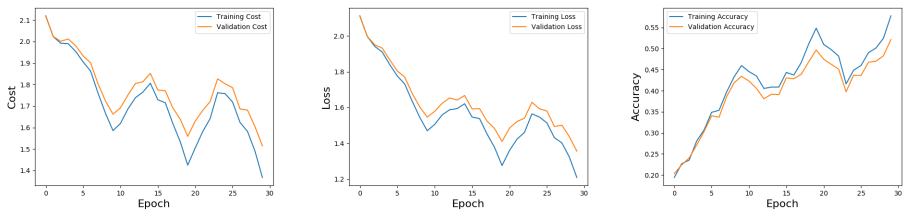
(b) Loss



(c) Accuracy

Figure 11: 9 layer network with batch normalization and $\sigma = 0.1$

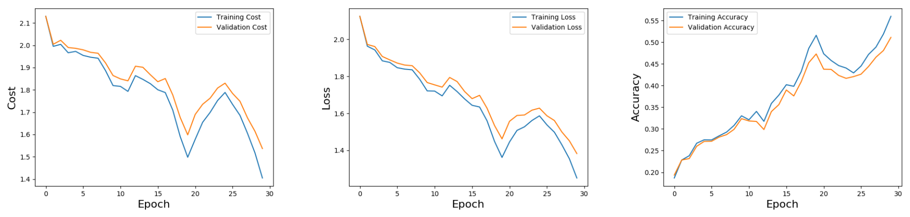
For $\sigma = 0.001$ we achieved an accuracy of 51.5% and got the following plots.



(a) Cost (b) Loss (c) Accuracy

Figure 12: 9 layer network with batch normalization and $\sigma = 0.001$

For $\sigma = 0.0001$ we achieved an accuracy of 50.5% and got the following plots.

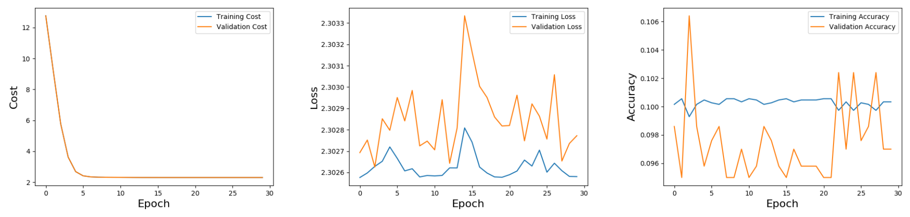


(a) Cost (b) Loss (c) Accuracy

Figure 13: 9 layer network with batch normalization and $\sigma = 0.0001$

Once again we now want to examine what happens when training the same networks without batch normalization.

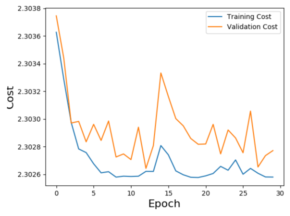
For $\sigma = 0.1$ we achieved an accuracy of 10% and got the following plots.



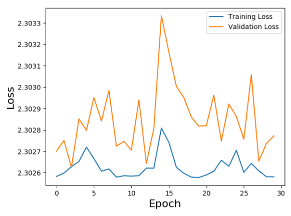
(a) Cost (b) Loss (c) Accuracy

Figure 14: 9 layer network without batch normalization and $\sigma = 0.1$

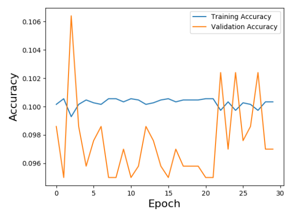
For $\sigma = 0.001$ we achieved an accuracy of 10% and got the following plots.



(a) Cost



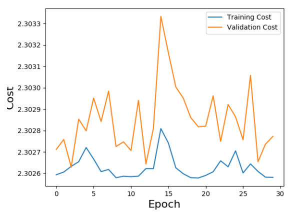
(b) Loss



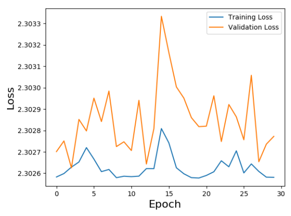
(c) Accuracy

Figure 15: 9 layer network without batch normalization and $\sigma = 0.001$

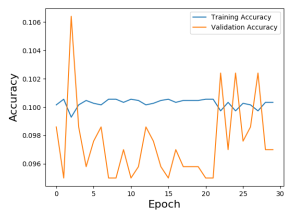
For $\sigma = 0.0001$ we achieved an accuracy of 10% and got the following plots.



(a) Cost



(b) Loss



(c) Accuracy

Figure 16: 9 layer network without batch normalization and $\sigma = 0.0001$

Conclusion

In conclusion we can clearly see from the aforementioned experiments that batch normalization seems to decrease the dependency on a good initialization. This is important as the network will still be able to learn even though it was initialized poorly which a network without batch normalization will not. Even when initialized poorly the network using batch normalization still performs at and adequate level compared to a He-initialized network.