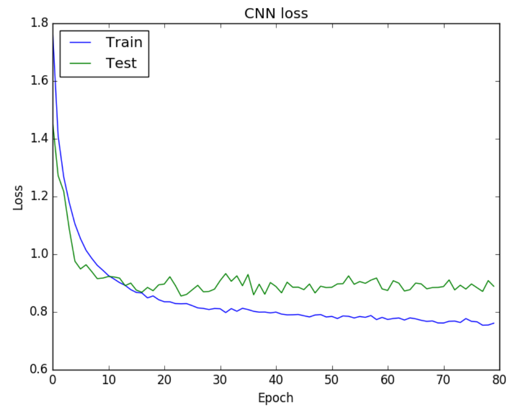
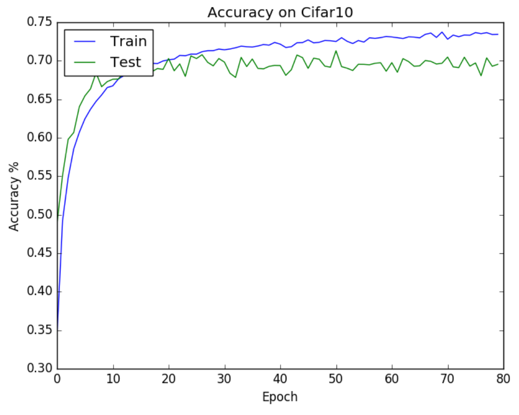
CS519 Assignment 3 Cifar10 Classification with CNNs
Neale Ratzlaff

1 Vanilla Model

a. Loss.



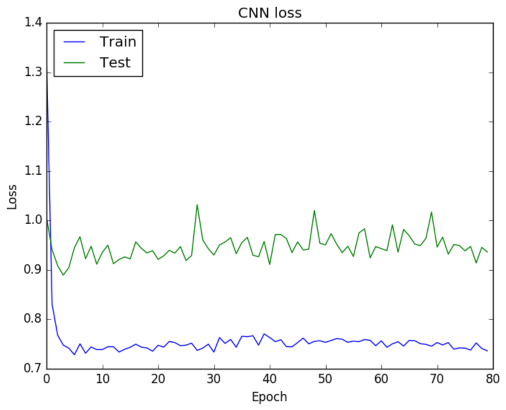
b. Accuracy



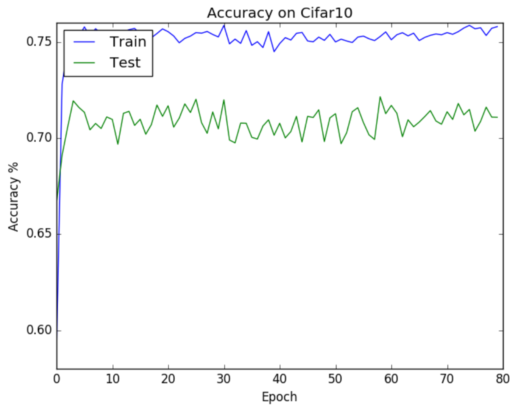
c. Analysis
This network was initialized without the dropout layer after the pre-classification FC layer. The network trained with nearly monotonic decrease in the loss and converged on percent accuracy

2 Adding an Additional FC Layer

a. Loss.



b. Accuracy

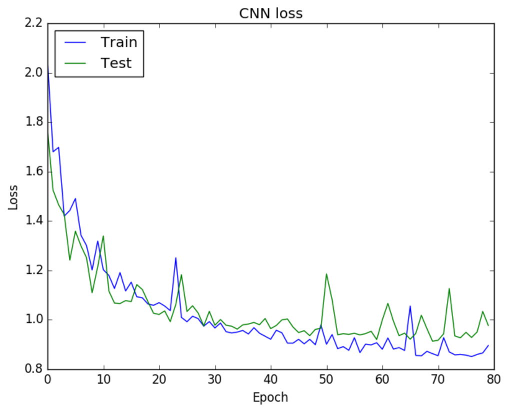


c. Analysis

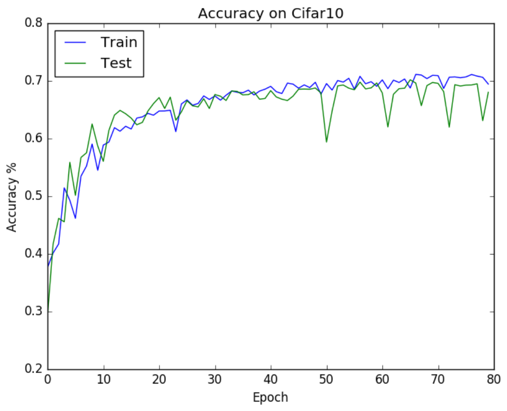
Here we altered the design of the model that we used in the first part. By initializing from weights achieved in the first run, we will start from the point in optimization that the model was at when 80 epochs elapsed in the first training run. As such, we start from a lower loss level. The loss is higher than the final loss in part 1 because there is a new layer with parameters that have not been tuned on the dataset. The model takes time to adjust to the new layer. Arguably this does not work very well as neither loss nor accuracy improve over the first model. Adding another FC layer to a network as small as our Cifar10 one isn't likely to gain any accuracy points.

3 Using Adam Adaptive Optimizer

a. Loss.



b. Accuracy

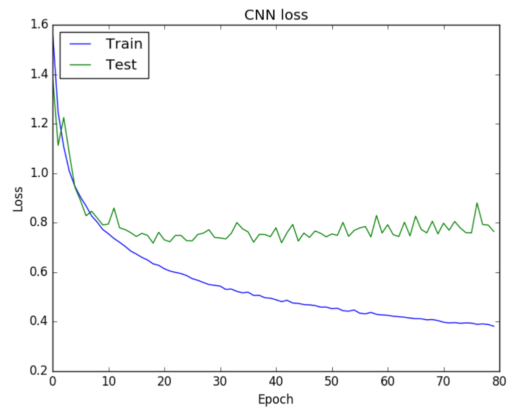


c. Analysis

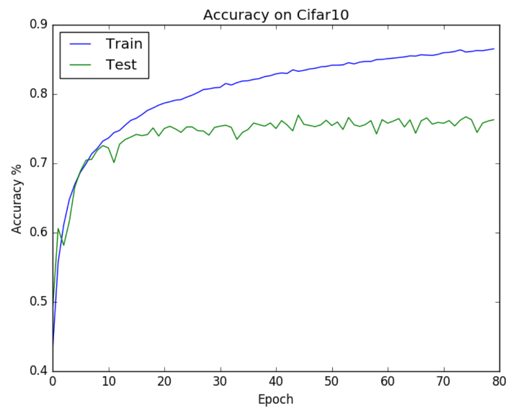
By using an adaptive optimizer instead of momentum or learning rate annealing, a network should be able to converge better and faster. Adam works by storing past subgradients and adjusting the step size with a finer level of granularity than momentum would be able to. Adam is widely viewed as the best choice of optimizer in a neural network, along with RMSProp. RMSProp here did very badly, and caused the network to completely diverge.

4 Addition of Batch Norm, and of additional, larger conv layers

a. Loss.



b. Accuracy



c. Analysis

Batch normalization nearly always has positive results when used in a convnet, it keeps the network from incurring high bias in deep layers. Similar to how normalization of input data allows the model to learn an easier distribution; by using batchnorm we can again simplify the target and improve optimization with respect to classification accuracy. Because of batchnorm I also take advantage of more convolutional layers without worrying about overfitting. The result is a better network than was originally posed, with an improvement of about six percent accuracy. In order to further increase classification accuracy, conv-batchnorm-relu layers would be repeated with more filters in each layer.