Homework 8

Harrison Kiang (hkiang2), Umberto Ravaioli (urjav2), Annlin Sheih (sheih2)

1.

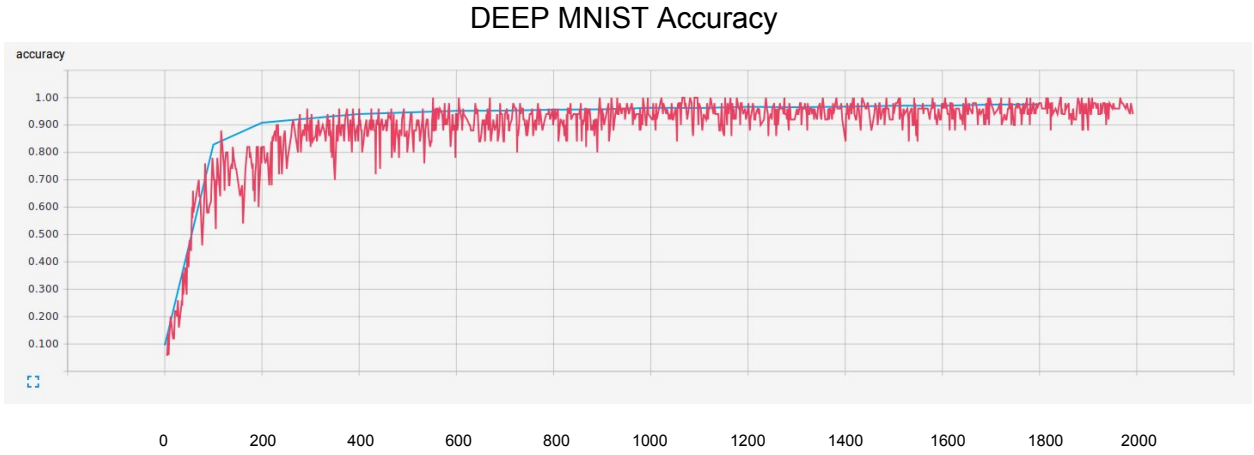


Figure 1.1: Accuracy for training (red) and test data (blue) for every 100 cycles. After approximately 2k steps, accuracy approached 97.44%.

2. Various modifications to the MNIST deep learning convolutional network tutorial architecture were made to demonstrate the various ways in which these parameters affect accuracy. All the variations followed the same basic model as the tutorial and do not represent a radical departure in architecture, but rather an investigation into the selection of network parameters. The investigation was limited to: convolution kernel size, convolution layer features size, network depth, and max pooling.

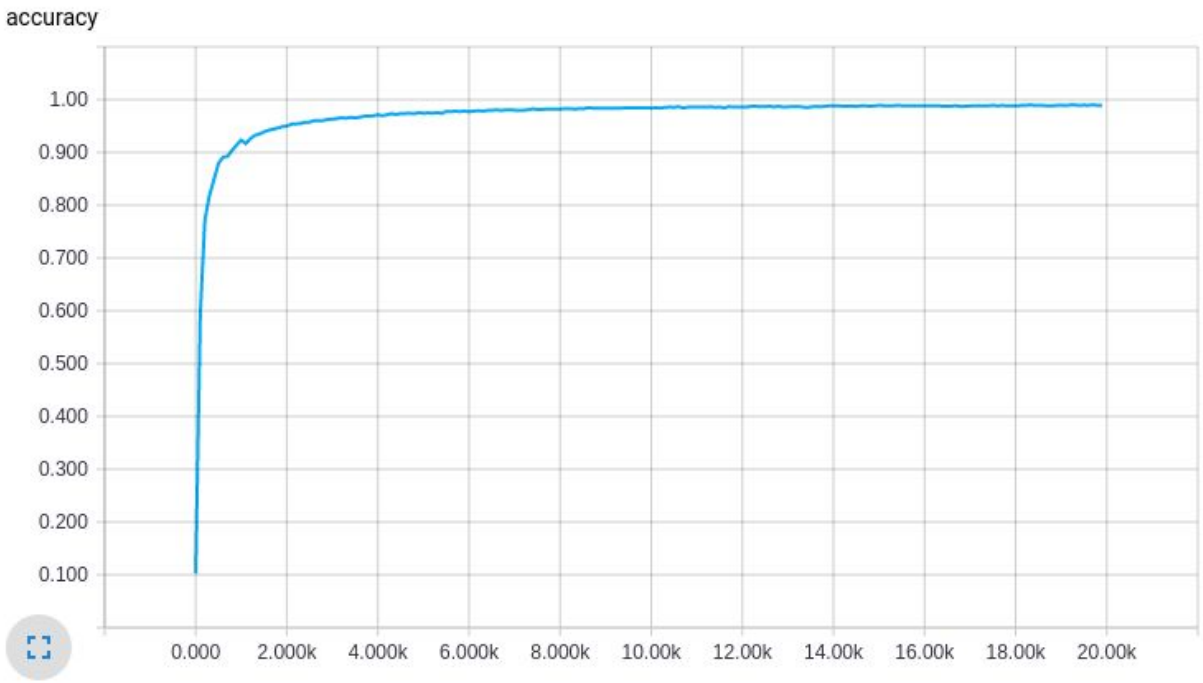
The results as seen below show that within this particular type of architecture, the given tutorial was very finely tuned. Shrinking the kernel size below 5x5 to 3x3 did not have a noticeable impact as observed in architectures iii, iv, vi, and vii. However as shown in v, increasing the kernel size to 7x7 broke the network reducing the accuracy to essentially guessing. Furthermore increasing the number of convolutional layers beyond 2 had little impact on accuracy, likely due to the small size and simplicity of the images.

The greatest impact was on balancing the number of features per pixel and the number of pixels considered in the final layer. Max pooling destroys information from most of the images but allows a larger feature vector given the same computational resources. In these results, the architectures with max pooling and a larger number of features had the best accuracy showing that the max pooling trade-off is well worth it. This makes intuitive sense since max pooling allows the most prominent patterns in a region (which tend to be locally redundant) to be passed to the next layers freeing up computational resources to get a better characterization on the types of patterns present.

i. 3x3 kernel, 16 features, max pool

3x3x8 conv + 2x2 max pool -> 3x3x16 conv + 2x2 max pool ->
7*7*16x1024 fully connected

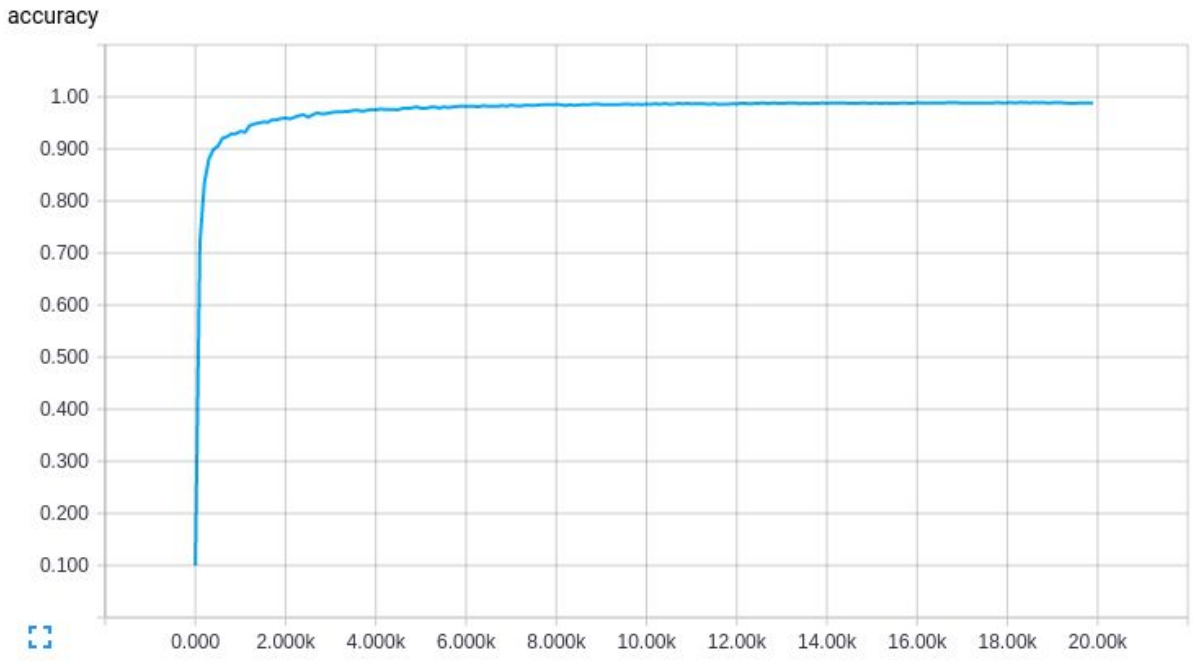
Final test acc. = 98.92%



ii. 3x3 kernel, 8 features, no max pool

3x3x8 conv -> 3x3x8 conv ->
28*28*8x1024 fully connected

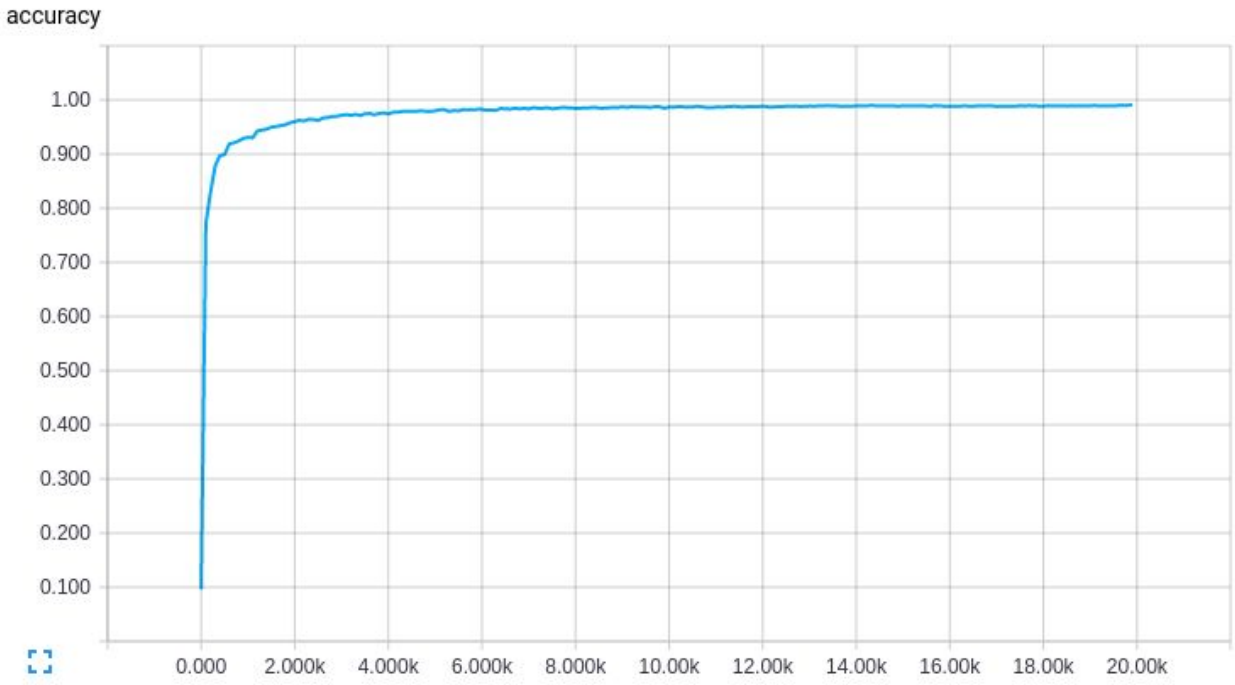
Final test acc. = 98.82%



iii. 3x3 kernel, 32 features, no max pool, 3 conv layers deep

3x3x8 conv -> 3x3x16 conv -> 3x3x32 conv ->
28*28*32x1024 fully connected

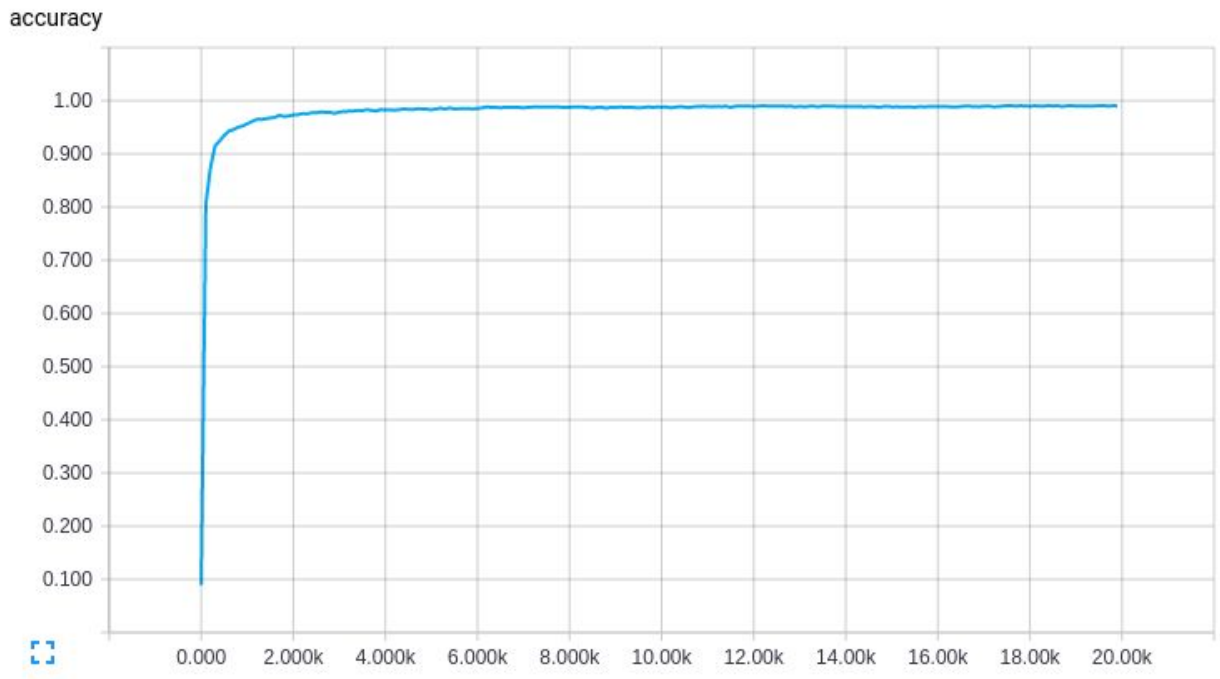
Final test acc. = 99.02%



iv. 5x5 kernel, 32 features, no max pool, 3 conv layers deep

5x5x8 conv -> 5x5x16 conv -> 5x5x32 conv ->
28*28*32x1024 fully connected

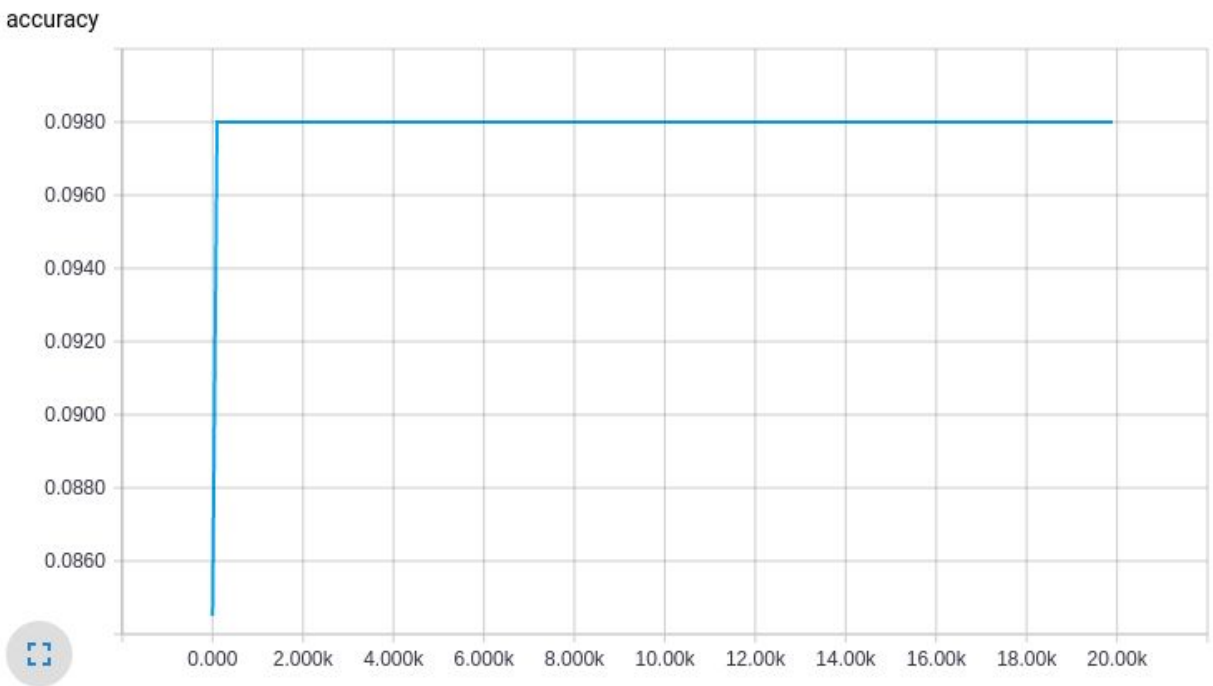
Final test acc. = 99.02%



v. 7x7 kernel, 32 features, no max pool, 3 conv layers deep

7x7x8 conv -> 7x7x16 conv -> 7x7x32 conv ->
28*28*32x1024 fully connected

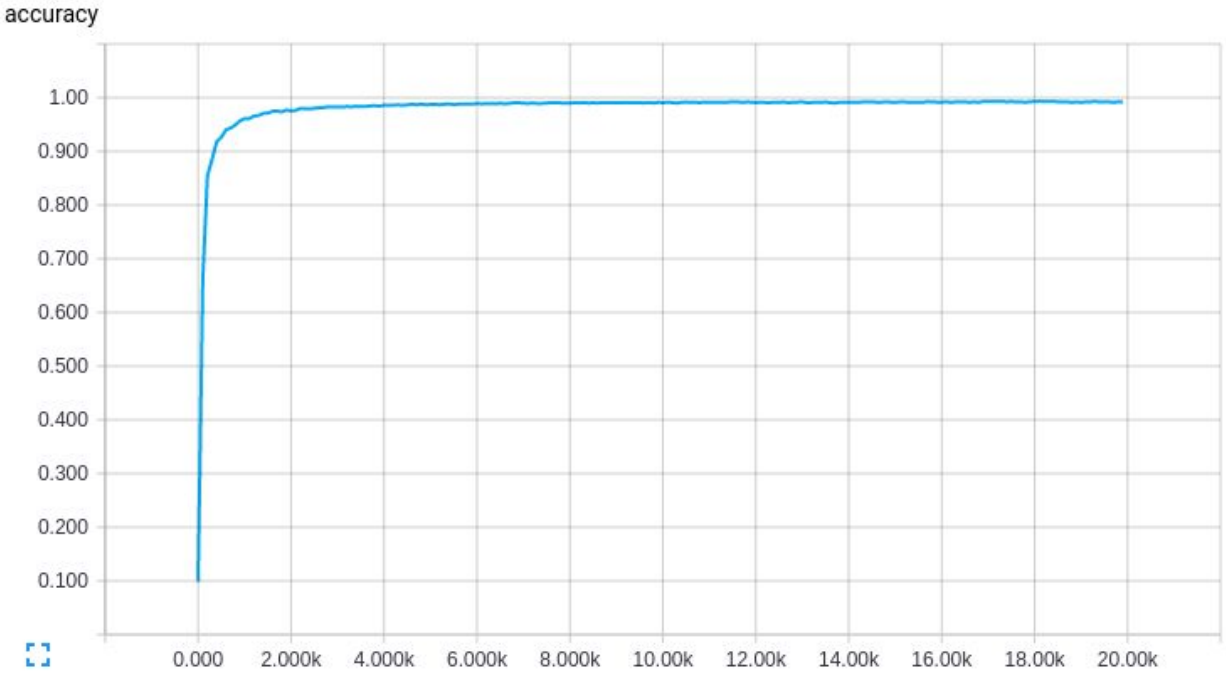
Final test acc. = 9.80%



vi. 3x3 kernel, 64 features, max pool, 4 conv layers deep

3x3x8 conv-> 3x3x16 conv ->
3x3x32 conv + 2x2 max pool -> 3x3x64 conv + 2x2 max pool ->
7*7*64x1024 fully connected

Final test acc. = 99.23%

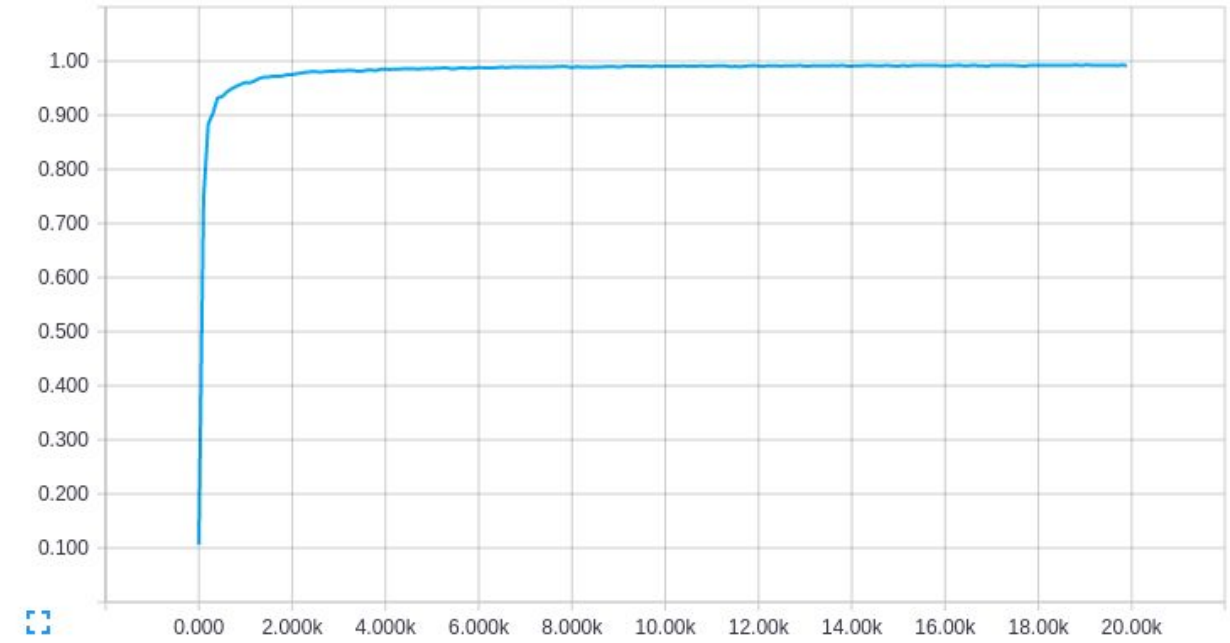


vii. 5x5 kernel, 64 features, max pool, 4 conv layers deep

5x5x8 conv-> 5x5x16 conv ->
5x5x32 conv + 2x2 max pool -> 5x5x64 conv + 2x2 max pool ->
7*7*64x1024 fully connected

Final test acc. = 99.19%

accuracy


1.00
0.900
0.800
0.700
0.600
0.500
0.400
0.300
0.200
0.100
0.000
2.000k
4.000k
6.000k
8.000k
10.00k
12.00k
14.00k
16.00k
18.00k
20.00k

3.

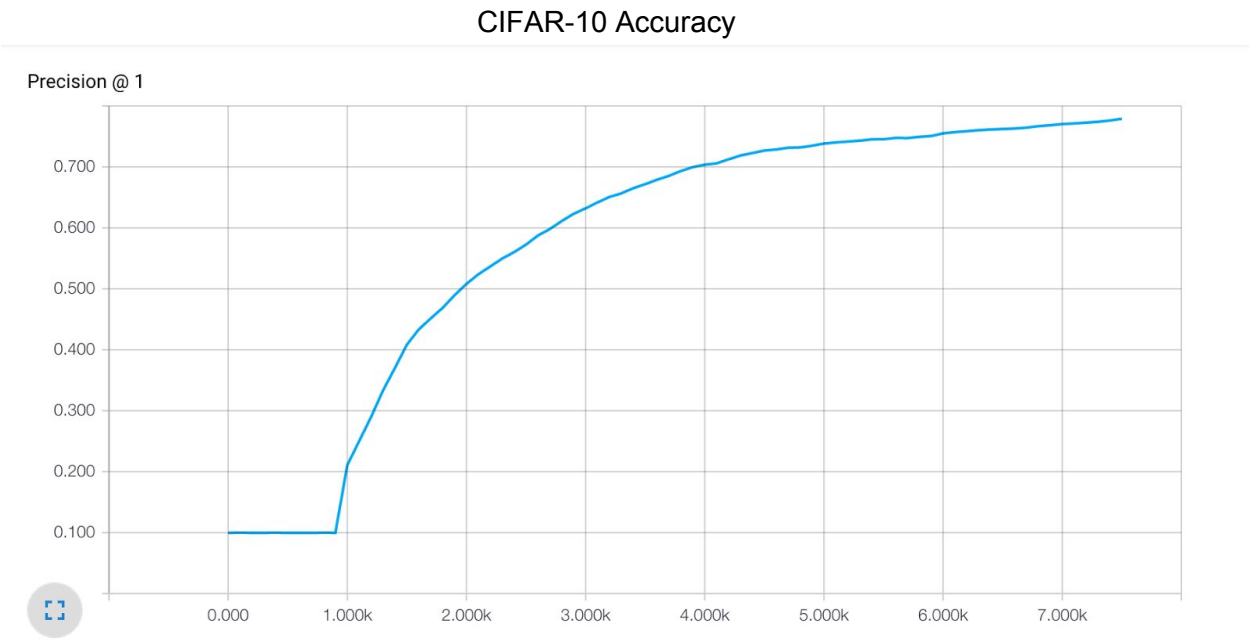


Figure 3.1: CIFAR unmodified. Accuracy is known to approach 86% according to tutorial. In the interest of time and due to the fact that the accuracy of the tutorial is known, only 7k steps were executed.

4.

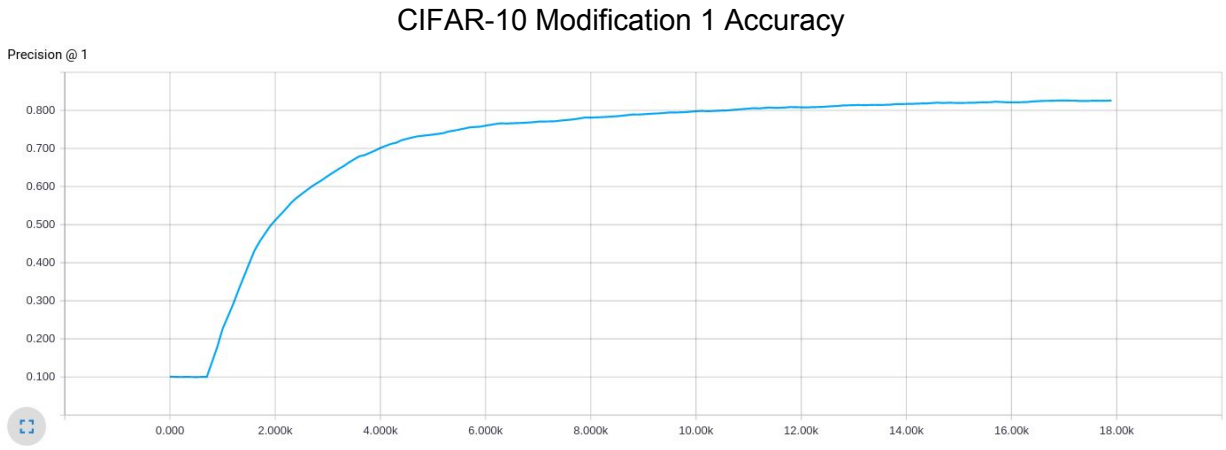


Figure 4.1: CIFAR with pool1 and pool2 disabled. Accuracy approached 82.5% after approx 18k steps (see inference2 in code). Performance is similar to that of the unmodified CIFAR algorithm.

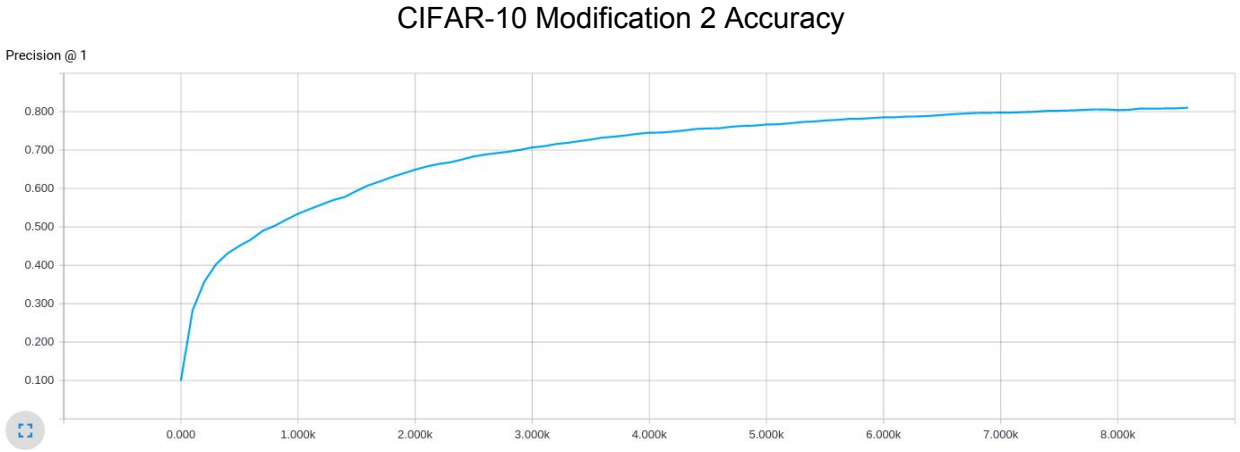


Figure 4.2: CIFAR with pool1 and pool2 disabled in addition to an expanded conv1 scope containing two convolutional layers. The first layer uses 5x5 tiles, 3 input channels, and 32 output channels. The second layer uses 5x5 tiles, 32 input channels, and 64 output channels. The interaction between these layers is identical to that of the deep MNIST tutorial. At 8.5k steps, accuracy approached 80.8%. This seems slightly better than the above figure with only pool1 and pool2 disabled for the number of steps run and how quickly the “fast start” (term borrowed from networking) of the accuracy curve progresses (see 2k steps in the above figure and the one above that).

CIFAR-10 Modification 3 Accuracy

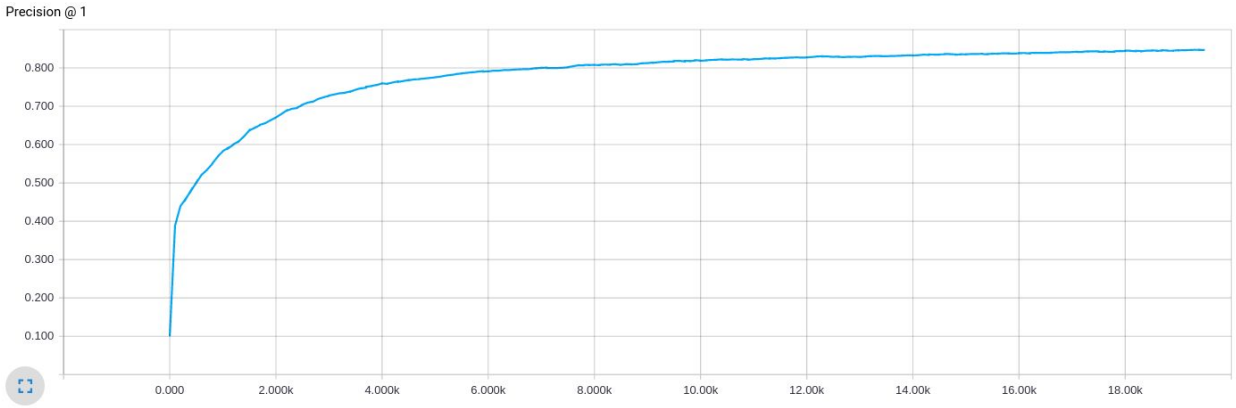


Figure 4.3: CIFAR with batch normalization in place of norm2 layer in addition to the expanded conv1 layer with pool1 and pool2 disabled. The curve seems similar to that of the previous graph above. Accuracy approaches 84.7% after approximately 19k steps. (See inference4 in cifar10.py)

CIFAR-10 Modification 4 Accuracy

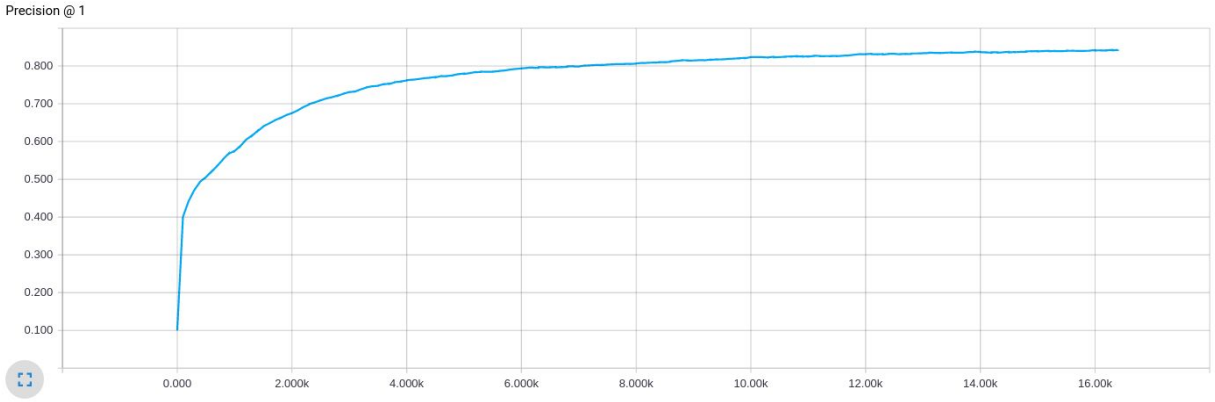


Figure 4.4: CIFAR with batch normalization after norm2 layer, before local3 layer in addition to the expanded conv1 layer with pool1 and pool2 disabled. The curve seems similar to that of the previous graph above. Accuracy approaches 84.3% after approximately 16.5k steps. (See inference5 in cifar10.py). Difference between this and the above: norm2 is included in its original format here, although the accuracy curve is nearly the same.

CIFAR-10 Modification 5 Accuracy

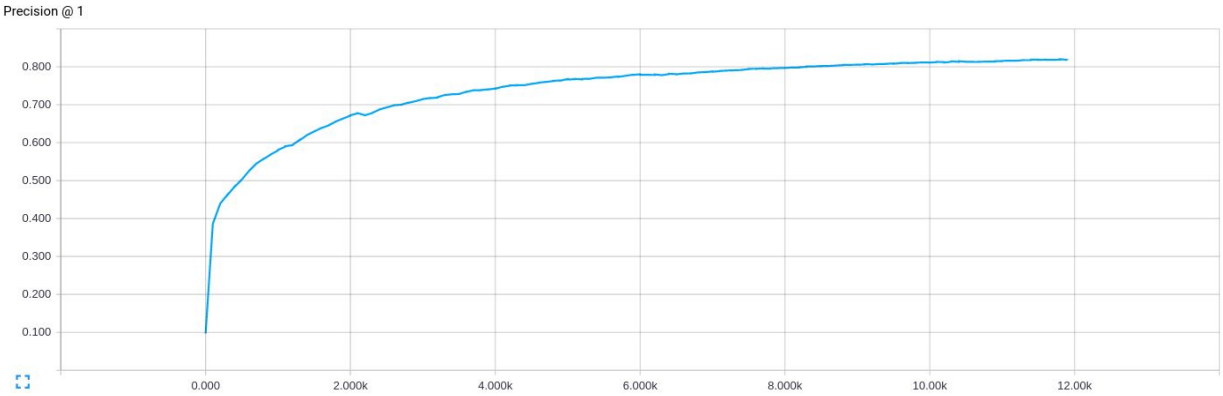


Figure 4.5: CIFAR with batch normalization after local3 layer, before local4 layer in addition to the expanded conv1 layer with pool1 and pool2 disabled. The curve seems similar to that of the previous graph above. Accuracy approaches 82.0% after approximately 12k steps. (See inference6 in cifar10.py).

More experiments were to be run, but Harrison Kiang's PSU started to smell like something was going to catch on fire. He decided not to risk it. RIP Harrison Kiang's PSU.

Observations: taking into account different run times, there does not seem to be sufficient difference in expanding the width of layer conv1 or adding batch normalization either in place of a normalization layer or between the original layers (save for conv1) of the CIFAR architecture. If a CUDA-enabled GPU was on hand, perhaps a more steps per experiment would have been run. The experiments were instead run until they reached a point of convergence of accuracy by inspection.