Deep Learning | Assignment 2

Michael Alvarino (maa2282), Richard Dewey (rld2126), Colby Wise (cjw2165)

October 31, 2017

1 Part 1

Convolutional Layers: We implemented two convolutional layers using TensorFlow's $tf.layers.conv2d()$ function.

Conv2d() Layer 1 Architecture
Filter Number: 96
Kernel Size: 3x3
Padding: Same
Activation: Relu

Conv2d() Layer 2 Architecture
Filter Number: 192
Kernel Size: 3x3
Padding: Same
Activation: Relu

Pooling Layers: We used two max pooling layers using TensorFlow's $tf.layers.max\_pooling2d()$ function following each of our Convolutional layers.

max pooling2d() Layer Architecture
Input: Conv2d() layers
Pool Size: 3x3
Stride Size: 2x2
Padding: Same

Normalization: We used *batch normalization* on the input data.

Optimizer: *Gradient Descent optimization* was used with a learning rate of $\alpha = 0.01$.

Fully-connected Layers: We have one fully-connected dense layer with 1024 neurons followed by Softmax transformation.

2 Part 2

Training/Test Data Size: 90% Train | 10% Test

Total Training Loss: 4.43

Training Accuracy: 100%

Test Accuracy: 78.2%

Duration of Training: 20 epochs | 450 iterations pp. epoch

Convolutional Layers: We experimented 2, 3, 4, 5, and 6 convolutional layers using TensorFlow's $tf.layers.conv2d()$ function. While we thought building a deeper network would lead to significantly improved accuracy on the test data we only noticed a marginal increase in accuracy while training time increased significantly. On the other hand, increasing the filter size from 32 to 192+ lead to an increase in accuracy. Greater than 192 filters lead to memory problems on a few runs as we struggled to fit our test data in memory. Additionally, we experimented with kernal sizes of 5, 7, and 9. We found that increasing the kernel size decreased the test accuracy on average. From this we started researching common kernel sizes (i.e. LeNet, ResNet, GoogleNet) and found out that most CNN networks keep kernel sizes below 5 and typically kernel sizes of 3 are used. As we decreased kernel size we found that test accuracies expectedly improved but more importantly the model required less epochs to converge.

Pooling Layers: After each Convolutional layer followed with a max pooling layer for down-sampling. The main reason behind this was that as we increased the number of layers and filters training time increased and we ran into memory problems. We continued to use TensorFlow's $tf.layers.max_pooling2d()$ function following each of Convolutional layers.

Normalization: We used *batch normalization* following each max pooling layer. We also experimented with *local response normalization*. With BN vs LRN our accuracy was 2% higher with batch normalization and we hit peak accuracies on the test data with less epochs. Additionally, we tested an exponential decay learning rate of $\alpha = .96$.

Dropout: We added *dropout* following each dense layer with a dropout rate of 50%.?????

Optimizer: *Adam Descent optimization* was used with a learning rate of $\alpha = 0.001$. We also experimented with TensorFlow's RMSPropOptimizer but accuracy on Adam Optimizer was meaningfully better.

Fully-connected Layers: We have one fully-connected dense layer with 1024 neurons followed by Softmax transformation.

3 Best Model

Loss | Accuracy: Total Loss: 3.73 | Train Accuracy: 100% | Test Accuracy: 78.62%

Training/Test Data Size: 90% Train | 10% Test

Duration of Training: 20 epochs | 450 iterations/epoch

Architecture Overview: We implemented two convolutional layers of 128 and 192 filters respectively and kernel size of 3, with two max pooling layers of kernel size 3 and stride 2. Moreover, we had one fully-connected layer of 1024 nuerons. In addition to input normalization and we used local response normalization after pooling with a 50% dropout rate per dense layer. During training we used AdaGrad optimizer.

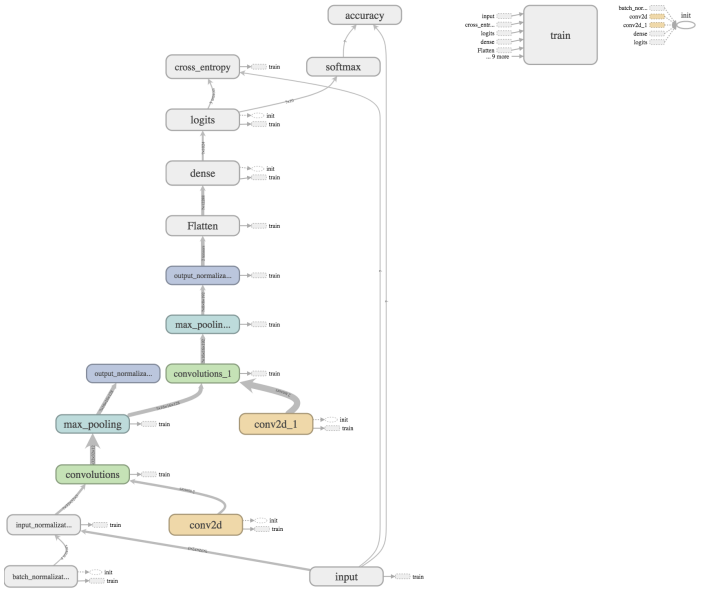

Figure 1: Tensorboard CNN Graph

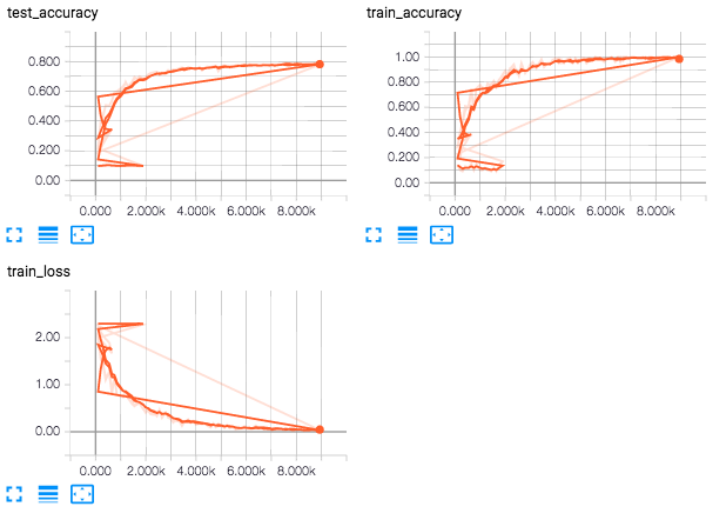

Figure 2: Loss and Accuracy Curves

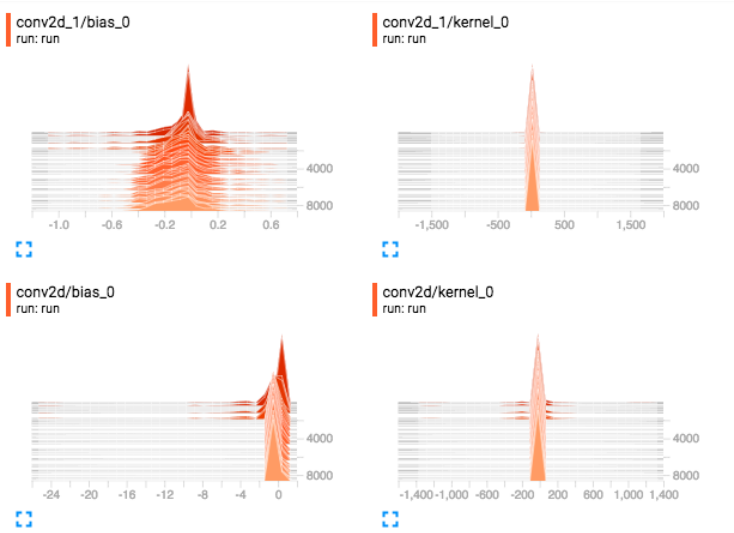

Figure 3: Hist of Conv Weight Gradients

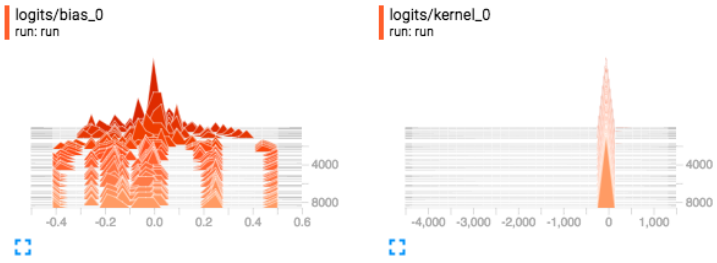

Figure 4: Hist of Fully-Connected Weight Gradients