Programming Assignment No.3

Due Date: April 27, 2018

Q1. Referring to the example exercises in the class, design and implement a network to classify the MNIST hand written digits.

The code for our network is provided from material for ex5 (2018/04/06).

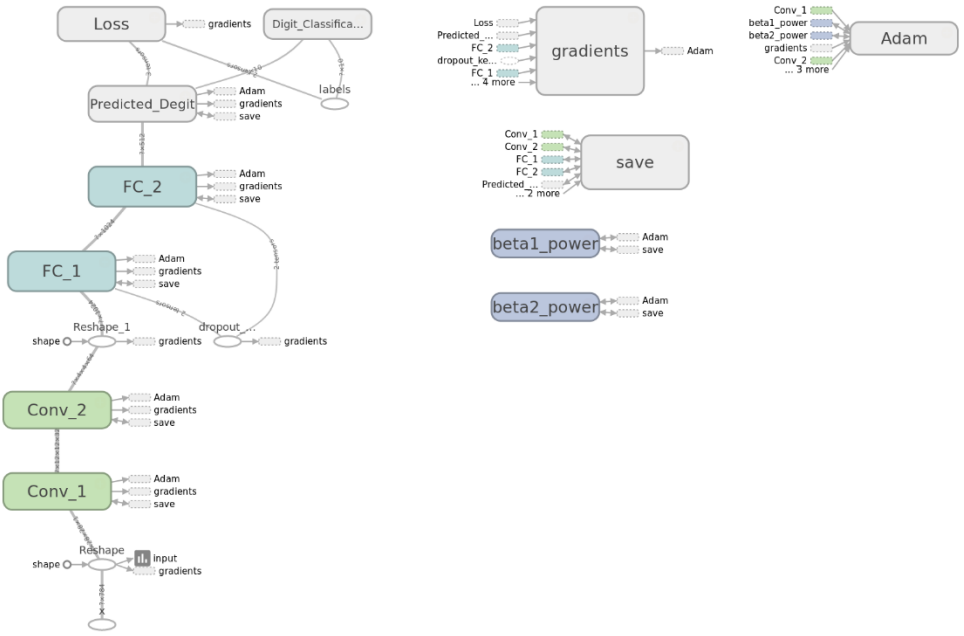

Figure 1 Network Structure for Submitted Code

This is a graph for our network. And this is the result of our network.

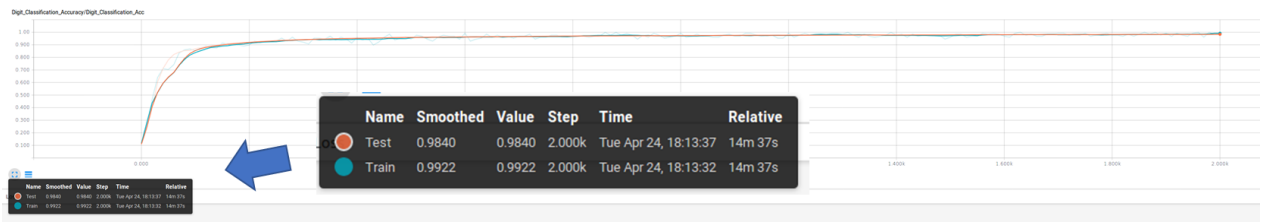

Figure 2 Training / Test Accuracy Result (Original)

Test classification accuracy is about 98.4%.

To enhance accuracy, I tried some modification on this network.

[Modification 1] Learning rate 0.0001 → 0.001

I multiplied `lRate` by 10, and observed how the network operates.

The graph of network doesn't change, and the result is as following.

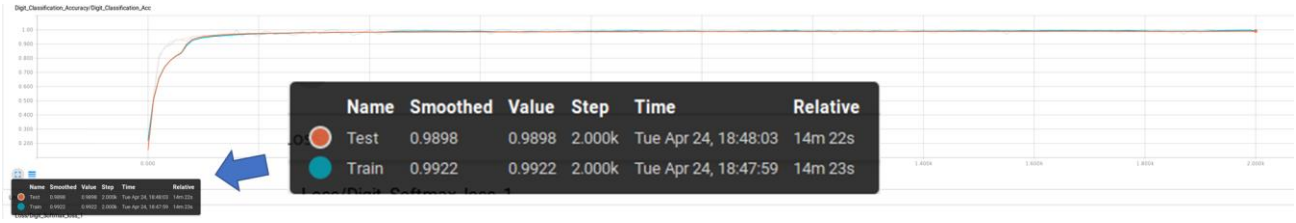

Figure 3 Training / Test Accuracy Result (Modification 1)

Test classification accuracy is about 98.98%.

[Modification 2] Adding a fc layer

I added an fc layer (fc3), by following code.

```
# ==========================================================================================================
#    This function creates the Fully connected Layer building block the building block
#     - Construct the CNN model as described in the slides
#         - input layer
#         - 2 convoltional layers
#         - 3 fully connected layers
#         - output layer
# ==========================================================================================================
def CNN_model(x, n_classes, nShape, nchannels, dropout ):
    # Reshape input picture
    x = tf.reshape(x, [-1, nShape, nShape, nchannels])
    tf.summary.image('input', x, 10)
    # first Convolution Layer
    conv1 = conv_layer(x, [3, 3, nchannels, 32], pool_kernel=[1, 3, 3, 1], pool_strides=[1, 2, 2, 1], name="Conv_1")
    # Second Convolution Layer
    conv2 = conv_layer(conv1, [3, 3, 32, 64], pool_kernel=[1, 3, 3, 1], pool_strides=[1, 2, 2, 1], name="Conv_2")

    conv2_shape = conv2.get_shape().as_list()
    flattened = tf.reshape(conv2, [-1,  conv2_shape[1] * conv2_shape[2] * conv2_shape[3]])

    flattened_shape = flattened.get_shape().as_list()
    # Fully connected layer 01
    # Reshape conv3 output to fit fully connected layer input
    fc1 = fc_layer(flattened, flattened_shape[1], 1024, activation_type="relu",  dropoutProb=dropout, name="FC_1")
    # Fully connected layer 02
    fc2 = fc_layer(fc1, 1024, 512, activation_type="relu",  dropoutProb=dropout, name="FC_2")
    # Fully connected layer 03
    fc3 = fc_layer(fc2, 512, 256, activation_type="relu",  dropoutProb=dropout, name="FC_3")
    # Output, class prediction
    Prediction = fc_layer(fc3, 256 , n_classes, activation_type=None, name="Predicted_Degit")
    return Prediction
```

Figure 4 Adding a fc layer

The graph for this network also changes, as we add a fc layer. This is the graph for modified network.

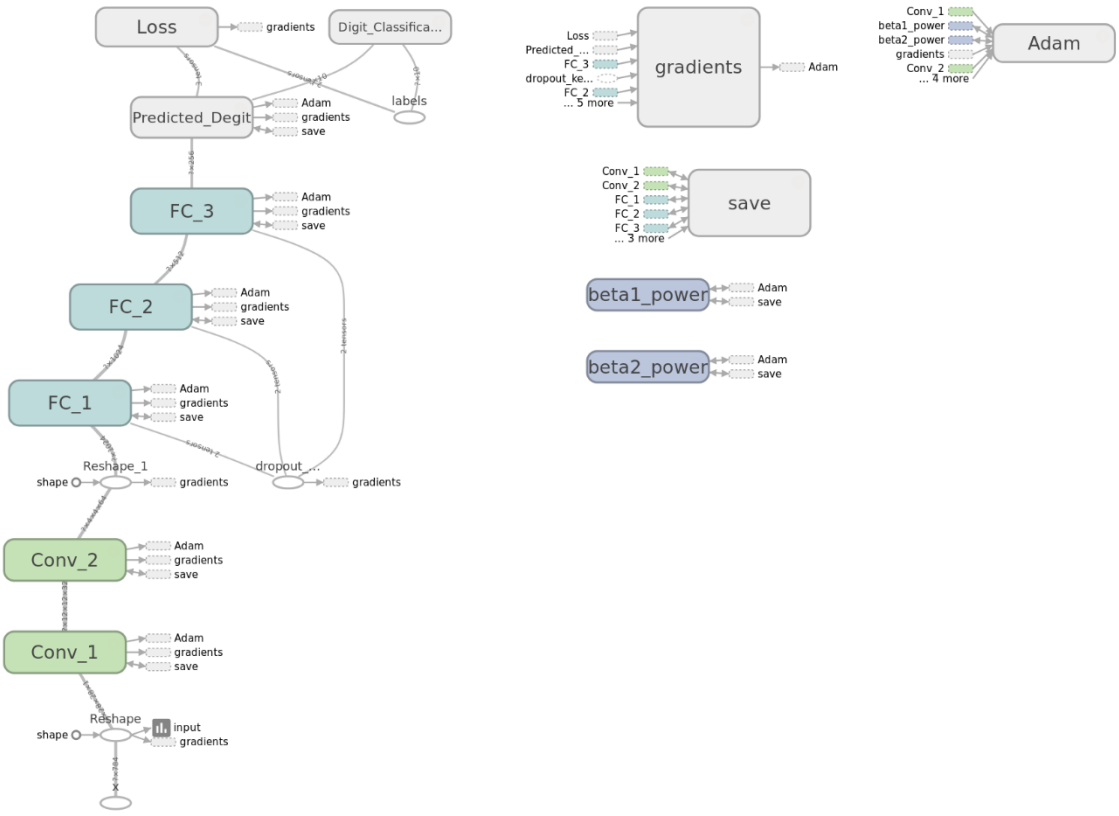

Figure 5 Network Structure of Modification 2

The result is as following.

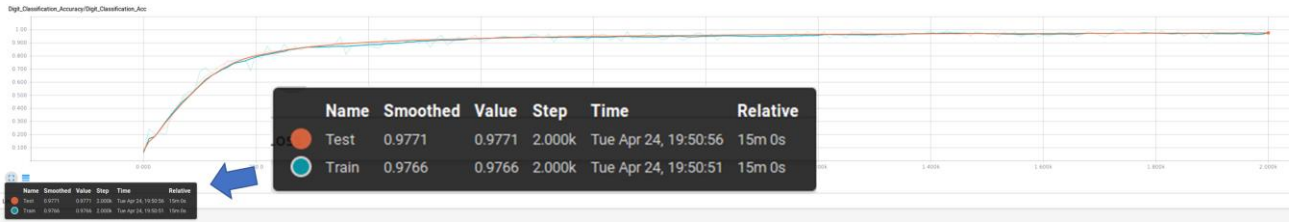

Figure 6 Training / Test Accuracy Result (Modification 2)

Test classification accuracy is about 97.71%

We can observe the training curve rises in slower rate, and accuracy has decreased. I decided not to increase number of layers.

Conclusion

I knew that if I increase epoch, higher accuracy can be acquired. So I took modification 1 (learning rate 0.01) and increased epoch from 2000 to 5000. This is the result.

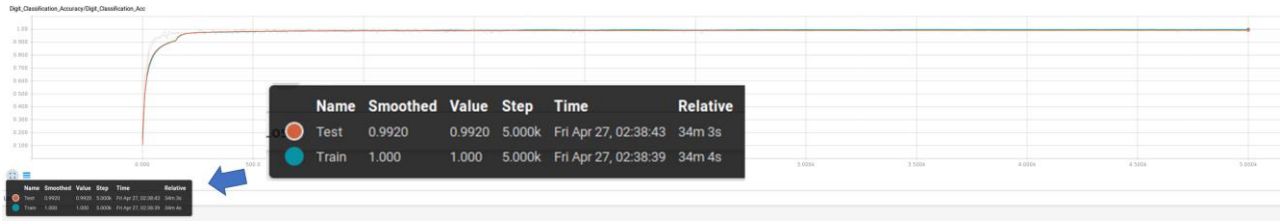

Figure 7 Training / Test Accuracy Result (Final)

Achieved test classification accuracy is about **99.2%**.

Code for this network is included in this zip file, with the log data folder "mnist_tutorial".