Report for 181130\_ CNN.py

1 Explain each line of code and put figure if you need

import tensorflow as tf

import numpy as np

import matplotlib.pyplot as plt

from tensorflow.examples.tutorials.mnist import input\_data

mnist = input\_data.read\_data\_sets("./mnist/data/", one\_hot=True)

#making nueral model

#to use cnn make it in 2d

X = tf.placeholder(tf.float32, [None, 28, 28, 1])

Y = tf.placeholder(tf.float32, [None, 10])

keep\_prob = tf.placeholder(tf.float32)

# each var and layer are set

# W1 [3 3 1 32] -> [3 3]: size of kernell, 1: inputed X , 32: number of filter

# L1 Conv shape=(?, 28, 28, 32)

# Pool ->(?, 14, 14, 32)

W1 = tf.Variable(tf.random\_normal([3, 3, 1, 32], stddev=0.01))

#using tf.nn.conv2d we can make a conv that move one point

#padding='SAME' is moving one more at the edge in kernell sliding

L1 = tf.nn.conv2d(X, W1, strides=[1, 1, 1, 1], padding='SAME')

L1 = tf.nn.relu(L1)

#Pooling can also use tf.nn.max\_pool to set easily

L1 = tf.nn.max\_pool(L1, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')

L1 = tf.nn.dropout(L1, keep\_prob)

#L2 Conv shape is=(?, 14, 14, 64)

# Pool is ->(?, 7, 7, 64)

#W2's [3, 3, 32, 64] 32 is the size of filter that is printed int w1 in L1

W2 = tf.Variable(tf.random\_normal([3, 3, 32, 64], stddev=0.01))

L2 = tf.nn.conv2d(L1, W2, strides=[1, 1, 1, 1], padding='SAME')

L2 = tf.nn.relu(L2)

L2 = tf.nn.max\_pool(L2, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')

L2 = tf.nn.dropout(L2, keep\_prob)

#input 7x7x64 -> output 256

#For full connection we bring the Pool size(?,7,7,64)from just before tom make dimension

#Reshapeing -> (?,256)

W3 = tf.Variable(tf.random\_normal([7 \* 7 \* 64, 256], stddev=0.01))

L3 = tf.reshape(L2, [-1, 7 \* 7 \* 64])

L3 = tf.matmul(L3, W3)

L3 = tf.nn.relu(L3)

L3 = tf.nn.dropout(L3, keep\_prob)

#get the final ouput from L3, get 256 as input date make 10 printout in 0~9label

W4 = tf.Variable(tf.random\_normal([256, 10], stddev=0.01))

model = tf.matmul(L3, W4)

# change the func to RMSPropOptimizer and check the result

cost = tf.reduce\_mean(tf.nn.softmax\_cross\_entropy\_with\_logits(logits=model, labels=Y))

optimizer = tf.train.AdamOptimizer(0.001).minimize(cost)

#optimizer = tf. train.RMSPropOptimizer(0.001,0.9).minimize(cost)

#########

# training

######

init = tf.global\_variables\_initializer() # init = op that initialize global var

sess = tf.Session() # a class for running tensorflow opertation

sess.run(init) #run session

batch\_size = 100 # number of picture worked in one time

total\_batch = int(mnist.train.num\_examples / batch\_size) # all example / batch size

for epoch in range(15): # after using one batch it means one epoch has past

total\_cost = 0

for i in range(total\_batch):

batch\_xs, batch\_ys = mnist.train.next\_batch(batch\_size)

# reshape to [28 28 1] to use in cnnm model

batch\_xs = batch\_xs.reshape(-1, 28, 28, 1) # reshapeing the tensor

\_, cost\_val = sess.run([optimizer, cost],

feed\_dict={X: batch\_xs, # feed values to the tensorflow

Y: batch\_ys,

keep\_prob: 0.7}) # give keep\_prob to feed\_dict

total\_cost += cost\_val

print('Epoch:', '%04d' % (epoch + 1),

'Avg. cost =', '{:.3f}'.format(total\_cost / total\_batch)) #one epoch done

print('Finish!')

#########

# Check result

######

is\_correct = tf.equal(tf.argmax(model, 1), tf.argmax(Y, 1))

accuracy = tf.reduce\_mean(tf.cast(is\_correct, tf.float32))

print('Accuracy:', sess.run(accuracy,

feed\_dict={X: mnist.test.images.reshape(-1, 28, 28, 1),

Y: mnist.test.labels,

keep\_prob: 1}))

#########

# Check result (matplot)

######

labels = sess.run(model,

feed\_dict={X: mnist.test.images.reshape(-1, 28, 28, 1),

Y: mnist.test.labels,

keep\_prob: 1})

fig = plt.figure()

for i in range(10):

subplot = fig.add\_subplot(2, 5, i + 1)

subplot.set\_xticks([])

subplot.set\_yticks([])

subplot.set\_title('%d' % np.argmax(labels[i]))

subplot.imshow(mnist.test.images[i].reshape((28, 28)),

cmap=plt.cm.gray\_r)

plt.show()

2 **Describe the size change of the feature map as the neural network flows (important)  
ex> Use this layer's ~ operation to change the size of the feature map from N** **x N to n x n.**

-----------------------------------------------------

Input data shape (28,28,1)

Filter=(3,3,32)

Stride =1

Convolution layer 1

(28 +(2\*1) – 3) /1 +1 = 28

(28 +(2\*1)- 3) /1 +1 = 28

Output data shape : (28,28,32)

Leraning Parameter: 288 (3\*3\*32)

-----------------------------------------------------

Max Pooling Layer 1

28 /2 = 14

28 /2 = 14

Output data shape : (14,14,32)

Leraning Parameter: 0

----------------------------------------------------

Input data shape (14,14,32)

filter=(3,3,64)

stride = 1

Convolution layer 2

(14 +(2\*1) – 3) /1 +1 = 14

(14 +(2\*1) – 3) /1 +1 = 14

Output data shape: (14,14,64)

Leraning Parameter: 576 (3\*3\*64)

----------------------------------------------------

Max Pooling Layer 2

14 /2 =7

14 /2 =7

Output data shape: (7,7,64)

Learning Parameter : 0

----------------------------------------------------

Flatten layer

Input data shape : (7,7,64)

Output data shape : (3136,256)

----------------------------------------------------

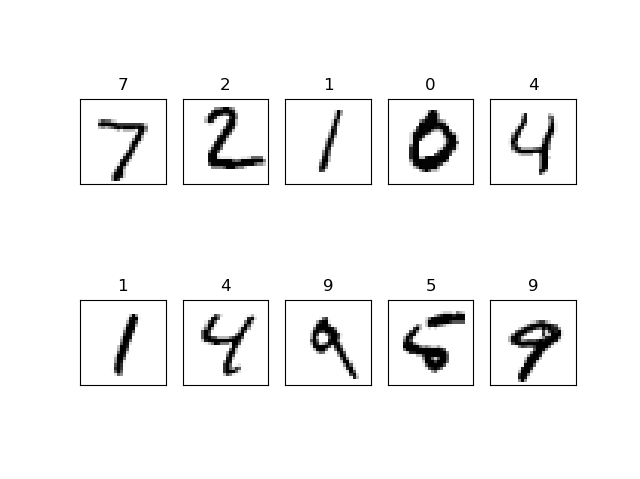
Softmax layer

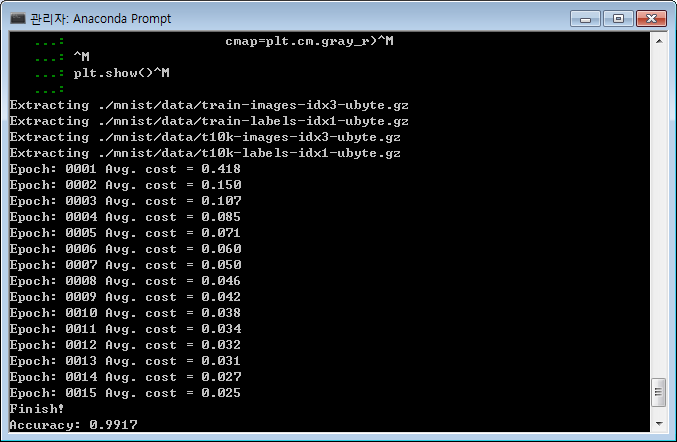
Input data shape : (3136,256)

Net : (256,10)

= 31360

3 Put the test results screen (Accuracy output)





Appendix

http://taewan.kim/post/cnn/

http://pythonkim.tistory.com/52

<https://github.com/golbin/TensorFlow-Tutorials/blob/master/07%20-%20CNN/01%20-%20CNN.py>