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) #reading the dataset

fig,ax = plt.subplots(10,10)

#display the train data

k = 0

for i in range(10):

for j in range(10):

ax[i][j].imshow(mnist.train.images[k].reshape(28,28), aspect='auto')

k += 1

plt.show()

print("Shape of feature matrix:", mnist.train.images.shape)

print("Shape of target matrix:", mnist.train.labels.shape)

print("One-hot encoding for 1st observation:\n", mnist.train.labels[0])

x = tf.placeholder("float", [None, 784]) #train set

W = tf.Variable(tf.zeros([784,10])) #weight

b = tf.Variable(tf.zeros([10])) #bias

y = tf.nn.softmax(tf.matmul(x,W) + b) #Finding weighted sum of inputs + bias

y\_ = tf.placeholder("float", [None,10])

cross\_entropy = -tf.reduce\_sum(y\_\*tf.log(y))

train\_step = tf.train.GradientDescentOptimizer(0.01).minimize(cross\_entropy) #minimise cost

init = tf.initialize\_all\_variables()

sess = tf.Session()

sess.run(init)

for i in range(1000):

batch\_xs, batch\_ys = mnist.train.next\_batch(100)

sess.run(train\_step, feed\_dict={x: batch\_xs, y\_: batch\_ys})

correct\_prediction = tf.equal(tf.argmax(y,1), tf.argmax(y\_,1)) #prediction value

accuracy = tf.reduce\_mean(tf.cast(correct\_prediction, "float")) #accuracy

print (sess.run(accuracy, feed\_dict={x: mnist.test.images, y\_: mnist.test.labels}))