Homework2

Classify MINIST data using CNN

Dept. of CSE, Yonsei University
2016323246 Junyoung Jang

In this report, I used Tensorflow Library which easily use Convolution Neural Network to train MNIST data set. (I agree with that it is necessary manually to make the algorithm and code, but we can expect more exact result as using Library.) This report introduce each steps that are involved to Computing Environment, Model Setting, Problem results.

- Computing Environment

Unlike MATLAB, Tensorflow is sensitive to computing environment. If you want to run my code, you should set the same environment at least.

- Python version : 3.6.0
- IDE : Pycharm Community 2016.3
- Library : Anaconda, Tensorflow 1.0.0

- Model Setting

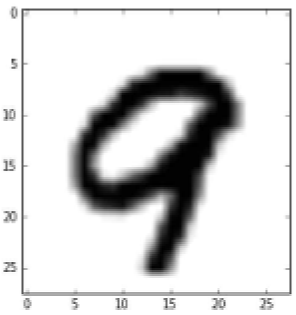
It used MNIST data included Tensorflow Library. The MNIST database of handwritten digits, available from this page, has a training set of 60,000 examples, and a test set of 10,000 examples.

I set model to classify MNIST data as below:

- Main Algorithm : CNN and Softmax regression
- Default #Convolution Layer : 2
- Default #Filter : Conv Layer 1(36) / Conv Layer 2(64)
- Default #Hidden Layer : 1
- #Neuron of each Hidden Layer : 1024 by 1024
- Training Epoch : 100
- Learning Rate : 0.001
- Batch Size : 128
- Drop rate : 50% in the train step.

Problem 1.

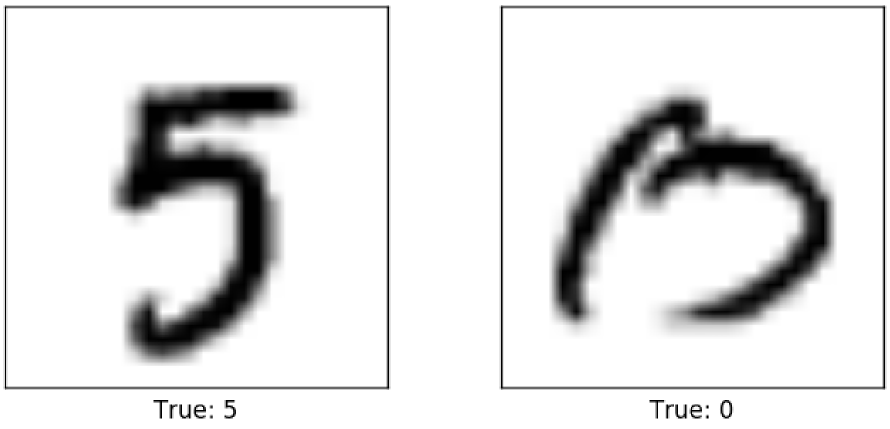
After reading the MNIST image file, please print it so that you can see it with



your eyes. The numbers below read the 17th digit of the train data.
Print out the 28th and 52nd numbers in the train data.

* the number of 28th : 5
* the number of 52nd : 0

solution.



Problem 2.

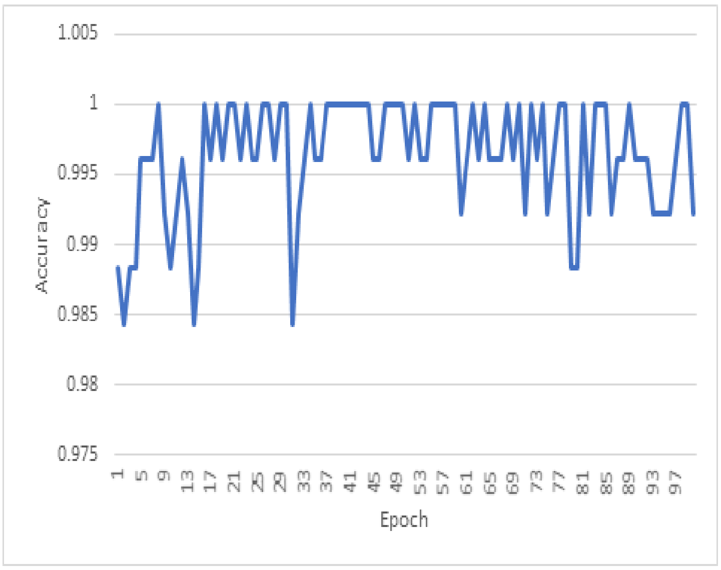
test accuracy : 95% or more accuracy

(The method of implementing machine learning is irrelevant.

Increase the accuracy by practicing various courses.)

solution.

I got test accuracy about 99.2% and accuracy at each epoch as below :



Epoch	Accuracy
1	0.988281
2	0.984375
3	0.988281
. . .	
98	1.000000
99	1.000000
100	0.992188
Test Accuracy	**0.992**

And I can know what is filter(weight) and image in each convolution layer (1&2) as below :

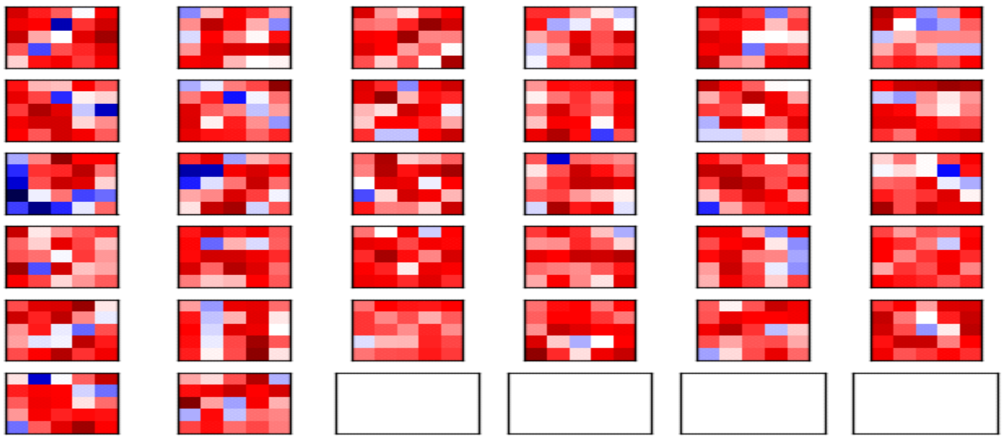


Figure 1. Optimized Filter(Weight) of Convolution Layer 1

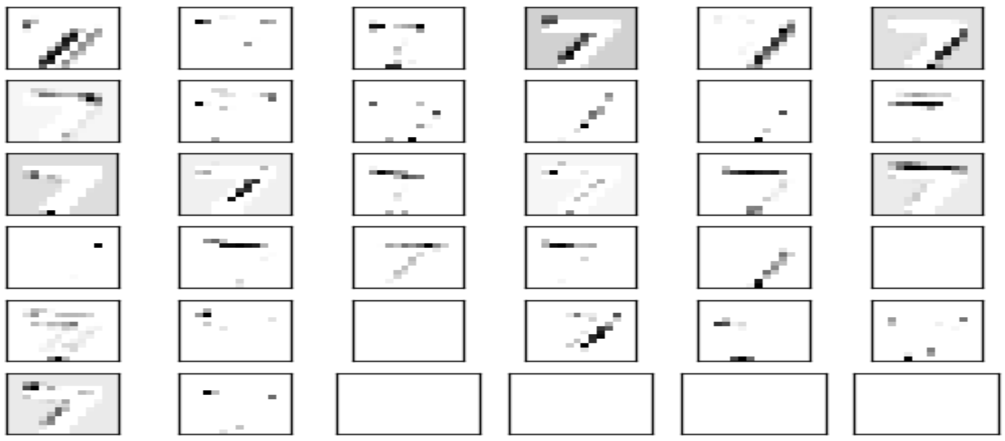


Figure 2. Image of Convolution Layer 1 using Optimized Filter(Weight)

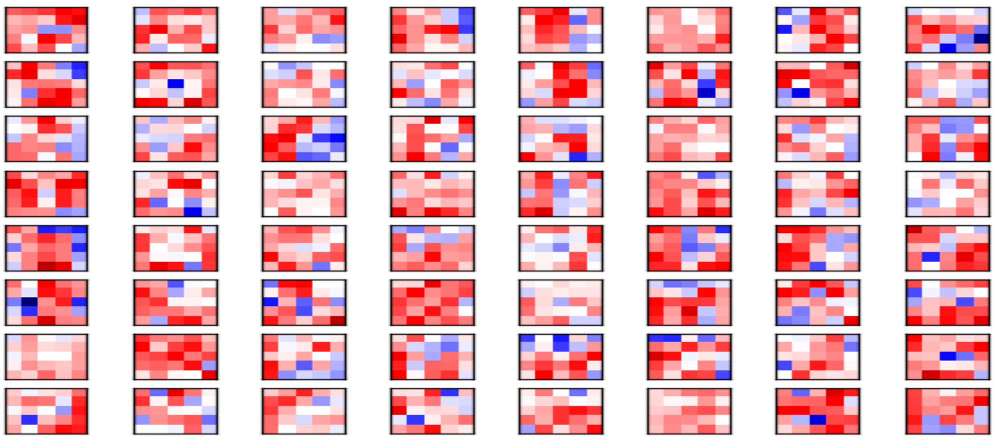


Figure 3. Optimized Filter(Weight) of Convolution Layer 1

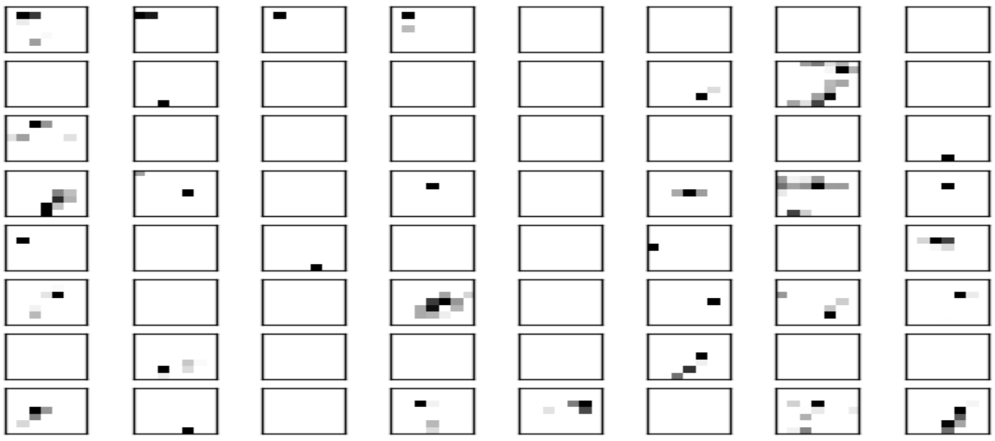


Figure 4. Image of Convolution Layer 1 using Optimized Filter(Weight)

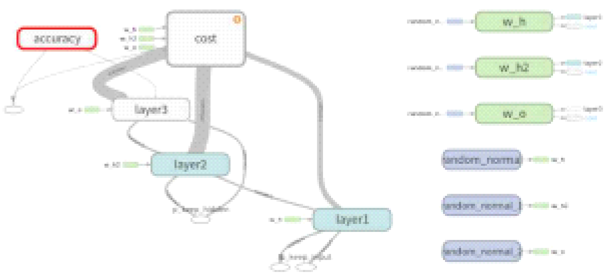
Problem 3.

Draw the flow chart.

1) If you use the tensorflow library, draw with tensorboad.

   The figure below is an example written with tensorboard.

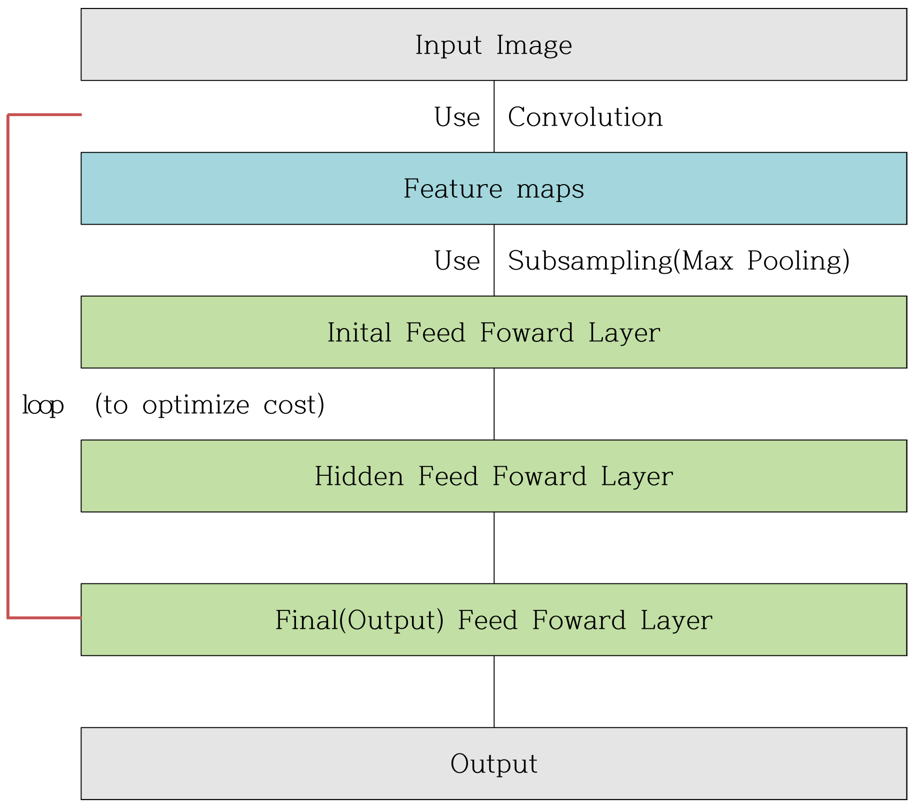
   Use the tensorboard as shown in the figure



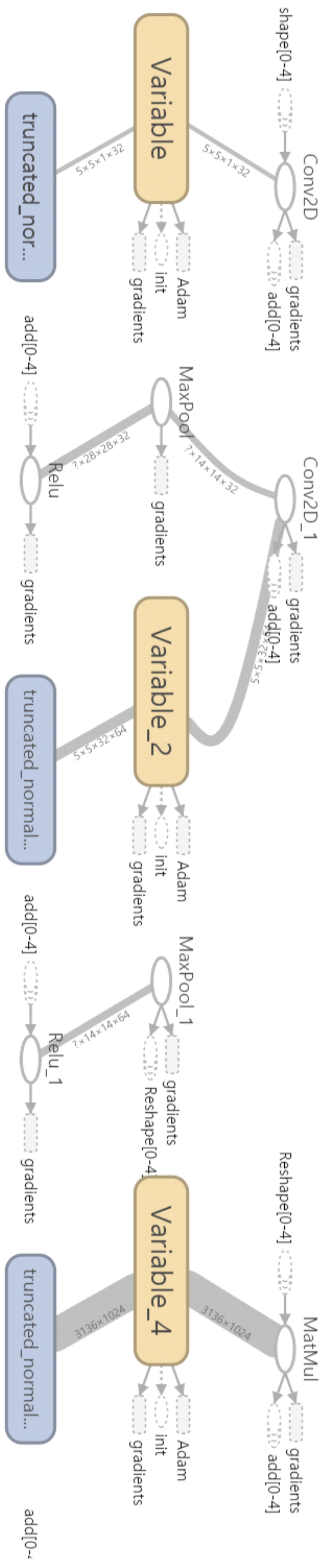
2) If you are writing a code without using the tensorflow library, draw the flowchart by hand or computer as shown in figure 1).

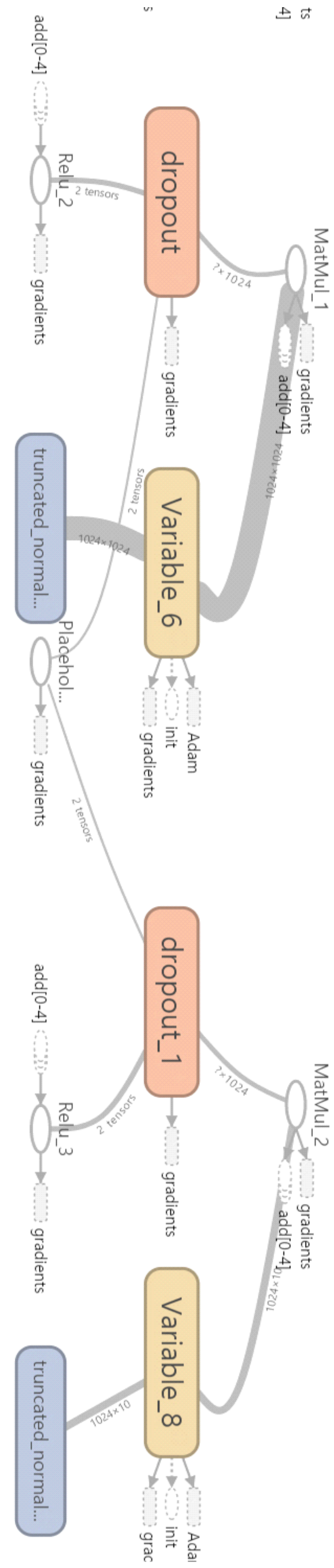
solution.

CNN is constructed as below:

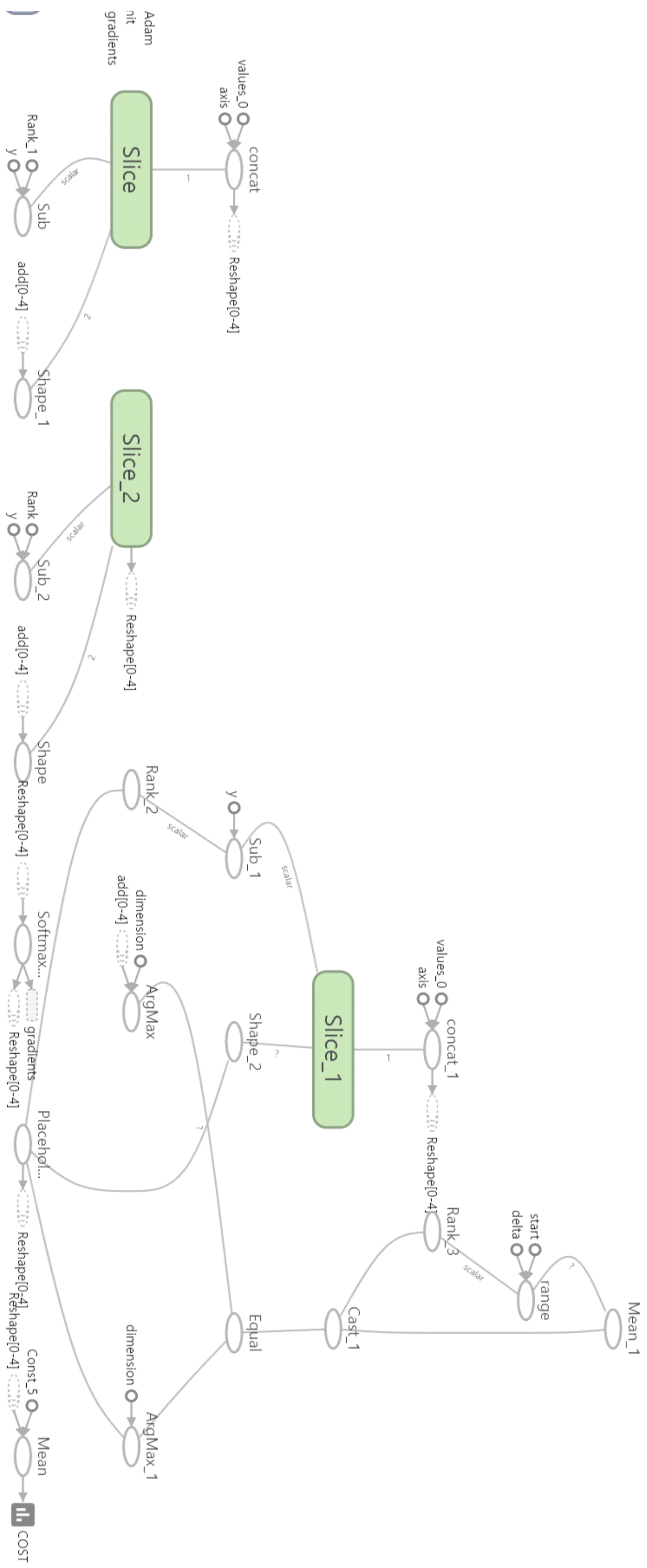


It is same flow chart of Tensorboard as next page:


shape[0-4]
Conv2D
5×5×1×32
Variable
5×5×1×32
truncated_nor...
gradients
add[0-4]
Adam
init
gradients
MaxPool
?×14×14×32
Conv2D_1
gradients
add[0-4]
gradients
add[0-4]
?×28×28×32
Relu
gradients
5×5×32×64
Variable_2
5×5×32×64
truncated_normal...
Adam
init
gradients
MaxPool_1
gradients
Reshape[0-4]
add[0-4]
?×14×14×64
Relu_1
gradients
Reshape[0-4]
MatMul
3136×1024
Variable_4
3136×1024
truncated_normal...
gradients
add[0-4]
Adam
init
gradients
add[0-



dropout
MatMul_1
?×1024
2 tensors
Relu_2
add[0-4]
gradients
gradients
gradients
add[0-4]
gradients
1024×1024
Variable_6
truncated_normal...
1024×1024
2 tensors
Placehol...
gradients
init
Adam
gradients
2 tensors
dropout_1
MatMul_2
?×1024
2 tensors
Relu_3
add[0-4]
gradients
gradients
add[0-4]
gradients
1024×10
Variable_8
truncated_normal...
1024×10
grac
init
Adai



If you want to see it in the detail, I include the tensorboard file in zip file. You can see it input "tensorboard —logdir='Location of the folder of my tensorboar' in command windows(ex. cmd, terminal)

[Appendix] Python Code

Step1. Load Library

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
import math
import time
import os
from tensorflow.examples.tutorials.mnist import input_data
```

Step2. Define plot function to use Problem 1

```
def Hw2PlotP1(images, cls_true, cls_pred=None):
    m = int(np.sqrt(images.shape[1]))
    img_shape = (m, m)
    fig, axes = plt.subplots(1, 2)
    fig.subplots_adjust(hspace=0.3, wspace=0.3)
    for i, ax in enumerate(axes.flat):
        ax.imshow(images[i].reshape(img_shape), cmap='binary')
        if cls_pred is None:
            xlabel = "True: {0}".format(cls_true[i])
        else:
            xlabel = "True: {0}, Pred: {1}".format(cls_true[i], cls_pred[i])
        ax.set_xlabel(xlabel)
        ax.set_xticks([])
        ax.set_yticks([])
    fig = plt.gcf()
    plt.show()
    fig.savefig("HW2Problem1.png")
```

Step3. Define plot function to draw weight&image of Conv

```
def Hw2PlotP2_convweight(weights, NAME):
    w = weights
    w_min = np.min(w)
    w_max = np.max(w)
    num_filters = w.shape[3]
    num_grids = math.ceil(math.sqrt(num_filters))
    fig, axes = plt.subplots(num_grids, num_grids)
    for i, ax in enumerate(axes.flat):
        if i < num_filters:
            img = w[:, :, 0, i]
            ax.imshow(img, vmin=w_min, vmax=w_max,
                                 interpolation='nearest', cmap='seismic')
        ax.set_xticks([])
        ax.set_yticks([])
    fig = plt.gcf()
    plt.show()
    fig.savefig(NAME)

def Hw2PlotP2_convimage(values, NAME):
    values = values
    num_filters = values.shape[3]
    num_grids = math.ceil(math.sqrt(num_filters))
    fig, axes = plt.subplots(num_grids, num_grids)
    for i, ax in enumerate(axes.flat):
        if i < num_filters:
            img = values[0, :, :, i]
            ax.imshow(img, interpolation='nearest', cmap='binary')
        ax.set_xticks([])
        ax.set_yticks([])
    fig = plt.gcf()
    plt.show()
    fig.savefig(NAME)
```

Step4. Define plot function to draw weight&image of Conv

```
def weight_variable(shape):
    return tf.Variable(tf.truncated_normal(shape=shape, stddev=0.1))

def bias_variable(shape):
    return tf.Variable(tf.constant(0.1, shape=shape))

def conv2d(x, W):
    return tf.nn.conv2d(x, W, strides=[1, 1, 1, 1], padding='SAME')

def max_pool_2x2(x):
    return tf.nn.max_pool(x, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')
```

Step5. Set parameters

```
LEARNING_RATE = 1e-3
DECAY_RATE = 0.9
EPOCH_TRAIN = 100
BATCH_SIZE = 128
TEST_SIZE = 256
HIDDEN_SIZE = 1
```

Step6. Load MNIST data and Set image size

```
mnist = input_data.read_data_sets("/tmp/data/", one_hot = True)
mnist.train.classes = np.argmax(mnist.train.labels, axis=1)
mnist.test.classes = np.argmax(mnist.test.labels, axis=1)
mnist.validation.classes = np.argmax(mnist.validation.labels, axis=1)
print("#Training set :\t\t{}".format(len(mnist.train.labels)))
print("#Test set :\t\t\t{}".format(len(mnist.test.labels)))
print("#Validation set :\t{}".format(len(mnist.validation.labels)))


img_size = 28
img_size_flat = img_size * img_size
img_shape = (img_size, img_size)
num_classes = 10
```

Step7. Plot Problem1

```
# Problem 1 of Homework 2
Problem1 = [28, 52]
images = mnist.train.images[Problem1]
classes_true = mnist.train.classes[Problem1]
Hw2PlotP1(images=images, cls_true=classes_true)
```

Step8. Set variables of Tensorflow session

```
x = tf.placeholder(tf.float32)
y = tf.placeholder(tf.float32)
keep_prob = tf.placeholder(tf.float32)
x_image = tf.reshape(x, shape=(-1, 28, 28, 1))
```

Step9. Convolution Layer 1 & 2

```
# Convolution layer 1 - 32 filters of shape (5,5,1)
#                     - input shape (-1,28,28,1)
#                     - output shape (-1,14,14,32)
W_conv1 = weight_variable(shape=(5, 5, 1, 32))
b_conv1 = bias_variable(shape=(1, 32))
h_conv1 = tf.nn.relu(conv2d(x_image, W_conv1) + b_conv1)
h_pool1 = max_pool_2x2(h_conv1)

# Convolution layer 2 - 64 filters of shape (5,5,32)
#                     - input shape (-1,14,14,32)
#                     - output shape (-1,7,7,64)
W_conv2 = weight_variable(shape=(5, 5, 32, 64))
b_conv2 = bias_variable(shape=(1, 64))
h_conv2 = tf.nn.relu(conv2d(h_pool1, W_conv2) + b_conv2)
h_pool2 = max_pool_2x2(h_conv2)
h_pool2_flat = tf.reshape(h_pool2, shape=(-1, 7 * 7 * 64))
```

Step10. Initial & Hidden & Final Layer

```
# Initial fully connected layer   - input shape (-1,7*7*64)
#                                 - output shape (-1,1024)
W = weight_variable(shape=(7 * 7 * 64, 1024))
b = bias_variable(shape=(1, 1024))
h = tf.nn.relu(tf.matmul(h_pool2_flat, W) + b)
h = tf.nn.dropout(h, keep_prob)

# Hidden fully connected layer    - input shape (1024,1024)
#                                 - output shape (-1,1024)
for i in range(HIDDEN_SIZE):
    W = weight_variable(shape=(1024, 1024))
    b = bias_variable(shape=(1, 1024))
    h = tf.nn.relu(tf.matmul(h, W) + b)
    h = tf.nn.dropout(h, keep_prob)

# Model - Final fully connected layer  - input shape (-1,1024)
#                                      - output shape (-1,10)
W = weight_variable(shape=(1024, 10))
b = bias_variable(shape=(1, 10))
score = tf.matmul(h, W) + b
```

Step11. Set Run Optimization method for train

```
# Calculate cost
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(labels=y, logits=score))

# Plot COST value using tensorboard
tf.summary.scalar('COST', cost)

# Optimize cost using Adam
train = tf.train.AdamOptimizer(LEARNING_RATE, DECAY_RATE).minimize(cost)

# Performance measures
prediction = tf.argmax(score, 1)
correct_prediction = tf.equal(prediction, tf.argmax(y, 1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
```

Step12-1. Run session of TensorFlow 1

```
# Create TensorFlow session
with tf.Session() as sess:
    # Initialize variables
    tf.global_variables_initializer().run()
    savedir_tensorboard = '%s/tensorboard'%os.getcwd()
    train_writer = tf.summary.FileWriter(savedir_tensorboard, sess.graph)

    # train
    for i in range(EPOCH_TRAIN):
        start_time = time.time()  # Start-time used for printing time-usage below.
        training_batch = zip(range(0, len(mnist.train.labels), BATCH_SIZE),
                             range(BATCH_SIZE, len(mnist.train.labels), BATCH_SIZE))
        for start, end in training_batch:
            _, cost_now = sess.run([train, cost],
            feed_dict={x: mnist.train.images[start:end, :],
                       y: mnist.train.labels[start:end, :], keep_prob: 0.5})
        end_time = time.time()
        time_dif = end_time - start_time
        test_indices = np.arange(len(mnist.test.labels))
        np.random.shuffle(test_indices)
        test_indices = test_indices[0:TEST_SIZE]
        print("Accuracy: %g" % sess.run(accuracy, feed_dict={x: mnist.test.images[:TEST_SIZE, :],
                       y: mnist.test.labels[:TEST_SIZE, :], keep_prob: 1.0}))
```

Step12-2. Run session of TensorFlow 2 for plot

```
    # Plot - Convolution layer 1 weights
    image1 = mnist.test.images[0:1]
    weights_conv1 = sess.run(W_conv1)
    bias_conv1 = sess.run(b_conv1)
    pltNAME = 'HW2Problem2_ConvL1weight.png'
    Hw2PlotP2_convweight(weights=weights_conv1, NAME=pltNAME)
    image1_conv1 = sess.run(
        max_pool_2x2(tf.nn.relu(
        tf.add(conv2d(tf.reshape(image1, shape=(-1, 28, 28, 1)), weights_conv1), bias_conv1))))
    pltNAME = 'HW2Problem2_ConvL1image.png'
    Hw2PlotP2_convimage(values=image1_conv1, NAME=pltNAME)
    # Plot - Convolution layer 2 weights
    weights_conv2 = sess.run(W_conv2)
    bias_conv2 = sess.run(b_conv2)
    pltNAME = 'HW2Problem2_ConvL2weight.png'
    Hw2PlotP2_convweight(weights=weights_conv2, NAME=pltNAME)
    image1_conv2 = sess.run(max_pool_2x2(
      tf.nn.relu(tf.add(conv2d(image1_conv1, weights_conv2), bias_conv2))))
    pltNAME = 'HW2Problem2_ConvL2image.png'
    Hw2PlotP2_convimage(values=image1_conv2, NAME=pltNAME)
```