In [1]:

```
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
from tensorflow.contrib.learn.python.learn.datasets.mnist import read_data_set
s
from tensorflow.python.framework import ops
ops.reset_default_graph()

# Start a graph session
sess = tf.Session()

# Load data
data_dir = 'temp'
mnist = read_data_sets(data_dir)

# Convert images into 28x28 (they are downloaded as 1x784)
train_xdata = np.array([np.reshape(x, (28,28)) for x in mnist.train.images])
test_xdata = np.array([np.reshape(x, (28,28)) for x in mnist.test.images])

# Convert labels into one-hot encoded vectors
train_labels = mnist.train.labels
test_labels = mnist.test.labels

# Set model parameters
batch_size = 100
learning_rate = 0.005
evaluation_size = 500
image_width = train_xdata[0].shape[0]
image_height = train_xdata[0].shape[1]
target_size = max(train_labels) + 1
num_channels = 1 # greyscale = 1 channel
generations = 500
eval_every = 5
conv1_features = 25
conv2_features = 50
max_pool_size1 = 2 # NxN window for 1st max pool layer
max_pool_size2 = 2 # NxN window for 2nd max pool layer
fully_connected_size1 = 100

x_input_shape = (batch_size, image_width, image_height, num_channels)
x_input = tf.placeholder(tf.float32, shape=x_input_shape)
y_target = tf.placeholder(tf.int32, shape=(batch_size))

#summary for accuray
trainAccuracy = tf.Variable(0.0, name="trainAccuracy")
tf.summary.scalar(tensor=trainAccuracy, name="accuracy")

#summary for loss
trainLoss = tf.Variable(0.0, name="TrainLoss")
tf.summary.scalar(tensor=trainLoss, name="xent")



eval_input_shape = (evaluation_size, image_width, image_height, num_channels)
eval_input = tf.placeholder(tf.float32, shape=eval_input_shape)
```

```
eval_target = tf.placeholder(tf.int32, shape=(evaluation_size))


# Convolutional layer variables
print (num_channels)
print (conv1_features)
print (conv2_features)
conv1_weight = tf.Variable(tf.truncated_normal([4, 4, num_channels, conv1_feat
ures],
                                               stddev=0.1, dtype=tf.float32))
conv1_bias = tf.Variable(tf.zeros([conv1_features], dtype=tf.float32))

conv2_weight = tf.Variable(tf.truncated_normal([4, 4, conv1_features, conv2_fe
atures],
                                               stddev=0.1, dtype=tf.float32))
conv2_bias = tf.Variable(tf.zeros([conv2_features], dtype=tf.float32))


# fully connected variables
resulting_width = image_width // (max_pool_size1 * max_pool_size2)
resulting_height = image_height // (max_pool_size1 * max_pool_size2)
full1_input_size = resulting_width * resulting_height * conv2_features


print "full1_input_size", full1_input_size
print "fully_connected_size1", fully_connected_size1
print "target_size", target_size


full1_weight = tf.Variable(tf.truncated_normal([full1_input_size, fully_connec
ted_size1],
                           stddev=0.1, dtype=tf.float32))
full1_bias = tf.Variable(tf.truncated_normal([fully_connected_size1],
stddev=0.1, dtype=tf.float32))
full2_weight = tf.Variable(tf.truncated_normal([fully_connected_size1, target_
size],
                                               stddev=0.1, dtype=tf.float32))
full2_bias = tf.Variable(tf.truncated_normal([target_size], stddev=0.1,
dtype=tf.float32))



# Initialize Model Operations
def my_conv_net(input_data):
    # First Conv-ReLU-MaxPool Layer
    conv1 = tf.nn.conv2d(input_data, conv1_weight, strides=[1, 1, 1, 1], paddi
ng='SAME')
    relu1 = tf.nn.relu(tf.nn.bias_add(conv1, conv1_bias))
    max_pool1 = tf.nn.max_pool(relu1, ksize=[1, max_pool_size1,
max_pool_size1, 1],
                               strides=[1, max_pool_size1, max_pool_size1, 1],
 padding='SAME')

    # Second Conv-ReLU-MaxPool Layer
    conv2 = tf.nn.conv2d(max_pool1, conv2_weight, strides=[1, 1, 1, 1], paddin
g='SAME')
    relu2 = tf.nn.relu(tf.nn.bias_add(conv2, conv2_bias))
```

```
    max_pool2 = tf.nn.max_pool(relu2, ksize=[1, max_pool_size2,
max_pool_size2, 1],
                               strides=[1, max_pool_size2, max_pool_size2, 1],
 padding='SAME')

    # Transform Output into a 1xN layer for next fully connected layer
    final_conv_shape = max_pool2.get_shape().as_list()
    final_shape = final_conv_shape[1] * final_conv_shape[2] *
final_conv_shape[3]
    flat_output = tf.reshape(max_pool2, [final_conv_shape[0], final_shape])

    # First Fully Connected Layer
    fully_connected1 = tf.nn.relu(tf.add(tf.matmul(flat_output, full1_weight),
 full1_bias))

    # Second Fully Connected Layer
    final_model_output = tf.add(tf.matmul(fully_connected1, full2_weight), ful
l2_bias)

    return(final_model_output)

model_output = my_conv_net(x_input)
test_model_output = my_conv_net(eval_input)

# Declare Loss Function (softmax cross entropy)
loss = tf.reduce_mean(tf.nn.sparse_softmax_cross_entropy_with_logits(logits=mo
del_output, labels=y_target))

# Create a prediction function
prediction = tf.nn.softmax(model_output)
test_prediction = tf.nn.softmax(test_model_output)

# Create accuracy function
def get_accuracy(logits, targets):
    batch_predictions = np.argmax(logits, axis=1)
    num_correct = np.sum(np.equal(batch_predictions, targets))
    return(100. * num_correct/batch_predictions.shape[0])


# Create an optimizer
my_optimizer = tf.train.MomentumOptimizer(learning_rate, 0.9)
train_step = my_optimizer.minimize(loss)




# Initialize Variables
init = tf.global_variables_initializer()
summary_op = tf.summary.merge_all()

sess.run(init)

writer = tf.summary.FileWriter("problem1_cnn")
writer.add_graph(sess.graph)

# Start training loop
train_loss = []
```

```
train_acc = []
test_acc = []
for i in range(generations):
    rand_index = np.random.choice(len(train_xdata), size=batch_size)
    rand_x = train_xdata[rand_index]
    rand_x = np.expand_dims(rand_x, 3)
    rand_y = train_labels[rand_index]
    train_dict = {x_input: rand_x, y_target: rand_y}

    sess.run(train_step, feed_dict=train_dict)
    temp_train_loss, temp_train_preds = sess.run([loss, prediction],
feed_dict=train_dict)
    temp_train_acc = get_accuracy(temp_train_preds, rand_y)
    writer.add_summary(sess.run(summary_op, {trainAccuracy: temp_train_acc, tr
ainLoss: temp_train_loss}), global_step=i)


    if (i+1) % eval_every == 0:
        eval_index = np.random.choice(len(test_xdata), size=evaluation_size)
        eval_x = test_xdata[eval_index]
        eval_x = np.expand_dims(eval_x, 3)
        eval_y = test_labels[eval_index]
        test_dict = {eval_input: eval_x, eval_target: eval_y}
        test_preds = sess.run(test_prediction, feed_dict=test_dict)
        temp_test_acc = get_accuracy(test_preds, eval_y)

        # Record and print results
        train_loss.append(temp_train_loss)
        train_acc.append(temp_train_acc)
        test_acc.append(temp_test_acc)
        acc_and_loss = [(i+1), temp_train_loss, temp_train_acc, temp_test_acc]
        acc_and_loss = [np.round(x,2) for x in acc_and_loss]
        print('Generation # {}. Train Loss: {:.2f}. Train Acc (Test Acc): {:.2
f} ({:.2f})'.format(*acc_and_loss))



writer.flush()
writer.close()
print ('Done')
```

```
Successfully downloaded train-images-idx3-ubyte.gz 9912422 bytes.
Extracting temp/train-images-idx3-ubyte.gz
Successfully downloaded train-labels-idx1-ubyte.gz 28881 bytes.
Extracting temp/train-labels-idx1-ubyte.gz
Successfully downloaded t10k-images-idx3-ubyte.gz 1648877 bytes.
Extracting temp/t10k-images-idx3-ubyte.gz
Successfully downloaded t10k-labels-idx1-ubyte.gz 4542 bytes.
Extracting temp/t10k-labels-idx1-ubyte.gz
1
25
50
full1_input_size 2450
fully_connected_size1 100
target_size 10
Generation # 5. Train Loss: 2.32. Train Acc (Test Acc): 5.00 (12.80)
Generation # 10. Train Loss: 2.13. Train Acc (Test Acc): 24.00 (18.60)
Generation # 15. Train Loss: 2.02. Train Acc (Test Acc): 43.00 (44.80)
Generation # 20. Train Loss: 1.85. Train Acc (Test Acc): 56.00 (55.80)
Generation # 25. Train Loss: 1.66. Train Acc (Test Acc): 66.00 (61.40)
Generation # 30. Train Loss: 1.43. Train Acc (Test Acc): 72.00 (76.60)
Generation # 35. Train Loss: 1.17. Train Acc (Test Acc): 69.00 (75.00)
Generation # 40. Train Loss: 0.84. Train Acc (Test Acc): 78.00 (78.00)
Generation # 45. Train Loss: 0.79. Train Acc (Test Acc): 77.00 (79.80)
Generation # 50. Train Loss: 0.73. Train Acc (Test Acc): 76.00 (80.40)
Generation # 55. Train Loss: 0.50. Train Acc (Test Acc): 78.00 (80.40)
Generation # 60. Train Loss: 0.49. Train Acc (Test Acc): 85.00 (80.00)
Generation # 65. Train Loss: 0.67. Train Acc (Test Acc): 81.00 (84.60)
Generation # 70. Train Loss: 0.44. Train Acc (Test Acc): 87.00 (84.40)
Generation # 75. Train Loss: 0.37. Train Acc (Test Acc): 86.00 (87.20)
Generation # 80. Train Loss: 0.55. Train Acc (Test Acc): 80.00 (86.80)
Generation # 85. Train Loss: 0.41. Train Acc (Test Acc): 88.00 (90.20)
Generation # 90. Train Loss: 0.34. Train Acc (Test Acc): 88.00 (88.40)
Generation # 95. Train Loss: 0.27. Train Acc (Test Acc): 93.00 (87.80)
Generation # 100. Train Loss: 0.41. Train Acc (Test Acc): 88.00 (87.80)
Generation # 105. Train Loss: 0.24. Train Acc (Test Acc): 93.00 (86.40)
Generation # 110. Train Loss: 0.37. Train Acc (Test Acc): 86.00 (89.60)
Generation # 115. Train Loss: 0.20. Train Acc (Test Acc): 95.00 (89.20)
Generation # 120. Train Loss: 0.46. Train Acc (Test Acc): 85.00 (92.40)
Generation # 125. Train Loss: 0.33. Train Acc (Test Acc): 92.00 (92.20)
Generation # 130. Train Loss: 0.30. Train Acc (Test Acc): 92.00 (92.80)
Generation # 135. Train Loss: 0.44. Train Acc (Test Acc): 90.00 (91.00)
Generation # 140. Train Loss: 0.30. Train Acc (Test Acc): 91.00 (92.60)
Generation # 145. Train Loss: 0.25. Train Acc (Test Acc): 93.00 (95.60)
Generation # 150. Train Loss: 0.18. Train Acc (Test Acc): 96.00 (93.00)
Generation # 155. Train Loss: 0.41. Train Acc (Test Acc): 89.00 (93.40)
Generation # 160. Train Loss: 0.22. Train Acc (Test Acc): 93.00 (93.60)
Generation # 165. Train Loss: 0.23. Train Acc (Test Acc): 90.00 (95.20)
Generation # 170. Train Loss: 0.11. Train Acc (Test Acc): 98.00 (93.80)
Generation # 175. Train Loss: 0.25. Train Acc (Test Acc): 91.00 (92.20)
Generation # 180. Train Loss: 0.23. Train Acc (Test Acc): 93.00 (93.20)
Generation # 185. Train Loss: 0.25. Train Acc (Test Acc): 94.00 (93.00)
Generation # 190. Train Loss: 0.18. Train Acc (Test Acc): 94.00 (95.40)
Generation # 195. Train Loss: 0.30. Train Acc (Test Acc): 93.00 (90.00)
Generation # 200. Train Loss: 0.17. Train Acc (Test Acc): 93.00 (94.00)
Generation # 205. Train Loss: 0.21. Train Acc (Test Acc): 92.00 (94.80)
Generation # 210. Train Loss: 0.20. Train Acc (Test Acc): 95.00 (93.60)
Generation # 215. Train Loss: 0.26. Train Acc (Test Acc): 93.00 (92.80)
```

```
Generation # 220. Train Loss: 0.19. Train Acc (Test Acc): 95.00 (94.20)
Generation # 225. Train Loss: 0.34. Train Acc (Test Acc): 89.00 (94.80)
Generation # 230. Train Loss: 0.18. Train Acc (Test Acc): 94.00 (93.60)
Generation # 235. Train Loss: 0.10. Train Acc (Test Acc): 96.00 (94.20)
Generation # 240. Train Loss: 0.12. Train Acc (Test Acc): 97.00 (93.60)
Generation # 245. Train Loss: 0.28. Train Acc (Test Acc): 91.00 (93.00)
Generation # 250. Train Loss: 0.23. Train Acc (Test Acc): 92.00 (93.20)
Generation # 255. Train Loss: 0.29. Train Acc (Test Acc): 92.00 (91.00)
Generation # 260. Train Loss: 0.19. Train Acc (Test Acc): 90.00 (93.80)
Generation # 265. Train Loss: 0.16. Train Acc (Test Acc): 97.00 (95.00)
Generation # 270. Train Loss: 0.13. Train Acc (Test Acc): 99.00 (95.60)
Generation # 275. Train Loss: 0.14. Train Acc (Test Acc): 95.00 (96.40)
Generation # 280. Train Loss: 0.13. Train Acc (Test Acc): 95.00 (94.60)
Generation # 285. Train Loss: 0.17. Train Acc (Test Acc): 95.00 (96.20)
Generation # 290. Train Loss: 0.18. Train Acc (Test Acc): 95.00 (95.00)
Generation # 295. Train Loss: 0.10. Train Acc (Test Acc): 95.00 (96.60)
Generation # 300. Train Loss: 0.06. Train Acc (Test Acc): 99.00 (95.80)
Generation # 305. Train Loss: 0.23. Train Acc (Test Acc): 93.00 (93.40)
Generation # 310. Train Loss: 0.21. Train Acc (Test Acc): 94.00 (95.60)
Generation # 315. Train Loss: 0.12. Train Acc (Test Acc): 97.00 (96.80)
Generation # 320. Train Loss: 0.12. Train Acc (Test Acc): 98.00 (95.20)
Generation # 325. Train Loss: 0.16. Train Acc (Test Acc): 96.00 (95.60)
Generation # 330. Train Loss: 0.21. Train Acc (Test Acc): 96.00 (94.60)
Generation # 335. Train Loss: 0.15. Train Acc (Test Acc): 94.00 (94.20)
Generation # 340. Train Loss: 0.20. Train Acc (Test Acc): 93.00 (94.80)
Generation # 345. Train Loss: 0.11. Train Acc (Test Acc): 96.00 (94.80)
Generation # 350. Train Loss: 0.16. Train Acc (Test Acc): 98.00 (96.00)
Generation # 355. Train Loss: 0.27. Train Acc (Test Acc): 94.00 (95.20)
Generation # 360. Train Loss: 0.18. Train Acc (Test Acc): 94.00 (96.80)
Generation # 365. Train Loss: 0.20. Train Acc (Test Acc): 93.00 (95.60)
Generation # 370. Train Loss: 0.12. Train Acc (Test Acc): 97.00 (94.40)
Generation # 375. Train Loss: 0.20. Train Acc (Test Acc): 94.00 (94.60)
Generation # 380. Train Loss: 0.30. Train Acc (Test Acc): 95.00 (95.60)
Generation # 385. Train Loss: 0.12. Train Acc (Test Acc): 96.00 (97.00)
Generation # 390. Train Loss: 0.11. Train Acc (Test Acc): 96.00 (96.20)
Generation # 395. Train Loss: 0.12. Train Acc (Test Acc): 97.00 (95.60)
Generation # 400. Train Loss: 0.13. Train Acc (Test Acc): 98.00 (94.60)
Generation # 405. Train Loss: 0.10. Train Acc (Test Acc): 97.00 (97.00)
Generation # 410. Train Loss: 0.20. Train Acc (Test Acc): 94.00 (97.20)
Generation # 415. Train Loss: 0.18. Train Acc (Test Acc): 95.00 (96.40)
Generation # 420. Train Loss: 0.13. Train Acc (Test Acc): 94.00 (95.40)
Generation # 425. Train Loss: 0.07. Train Acc (Test Acc): 98.00 (96.40)
Generation # 430. Train Loss: 0.15. Train Acc (Test Acc): 97.00 (94.00)
Generation # 435. Train Loss: 0.14. Train Acc (Test Acc): 96.00 (97.40)
Generation # 440. Train Loss: 0.09. Train Acc (Test Acc): 98.00 (94.80)
Generation # 445. Train Loss: 0.26. Train Acc (Test Acc): 92.00 (96.40)
Generation # 450. Train Loss: 0.10. Train Acc (Test Acc): 96.00 (95.80)
Generation # 455. Train Loss: 0.21. Train Acc (Test Acc): 92.00 (95.80)
Generation # 460. Train Loss: 0.16. Train Acc (Test Acc): 95.00 (96.00)
Generation # 465. Train Loss: 0.21. Train Acc (Test Acc): 93.00 (96.00)
Generation # 470. Train Loss: 0.09. Train Acc (Test Acc): 98.00 (95.80)
Generation # 475. Train Loss: 0.08. Train Acc (Test Acc): 98.00 (98.00)
Generation # 480. Train Loss: 0.10. Train Acc (Test Acc): 98.00 (95.60)
Generation # 485. Train Loss: 0.15. Train Acc (Test Acc): 95.00 (97.40)
Generation # 490. Train Loss: 0.17. Train Acc (Test Acc): 97.00 (98.60)
Generation # 495. Train Loss: 0.04. Train Acc (Test Acc): 99.00 (96.20)
```

```
Generation # 500. Train Loss: 0.08. Train Acc (Test Acc): 98.00 (95.20)
Done
```

In [ ]: