```
TENSORFLOW EXAMPLE: CONVOLUTIONAL NEURAL NETWORK
```

In [1]:

```
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
```

In [2]:

```
# Read in data
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets('MNIST_data', one_hot=True)
```

```
Extracting MNIST_data\train-images-idx3-ubyte.gz
Extracting MNIST_data\train-labels-idx1-ubyte.gz
Extracting MNIST_data\t10k-images-idx3-ubyte.gz
Extracting MNIST_data\t10k-labels-idx1-ubyte.gz
```

In [3]:

```
def TRAIN_SIZE(num):
    print ('Total Training Images in Dataset = ' + str(mnist.train.images.shape))
    print ('--------------------------------------------------')
    x_train = mnist.train.images[:num,:]
    print ('x_train Examples Loaded = ' + str(x_train.shape))
    y_train = mnist.train.labels[:num,:]
    print ('y_train Examples Loaded = ' + str(y_train.shape))
    print('')
    return x_train, y_train

def TEST_SIZE(num):
    print ('Total Test Examples in Dataset = ' + str(mnist.test.images.shape))
    print ('--------------------------------------------------')
    x_test = mnist.test.images[:num,:]
    print ('x_test Examples Loaded = ' + str(x_test.shape))
    y_test = mnist.test.labels[:num,:]
    print ('y_test Examples Loaded = ' + str(y_test.shape))
    return x_test, y_test

def display_train_digit(num):
    print(Y_train[num])
    label = Y_train[num].argmax(axis=0)
    image = X_train[num].reshape([28,28])
    plt.title('TRAINING Example: %d  Label: %d' % (num, label))
    plt.imshow(image, cmap=plt.get_cmap('gray_r'))
    plt.show()

def display_test_digit(num):
    print(Y_test[num])
    label = Y_test[num].argmax(axis=0)
    image = X_test[num].reshape([28,28])
    plt.title('TESTING Example: %d  Label: %d' % (num, label))
    plt.imshow(image, cmap=plt.get_cmap('gray_r'))
    plt.show()
```

In [4]:

```
# Establish the training dataset and the testing dataset
X_train, Y_train = TRAIN_SIZE(5500)
X_test, Y_test = TEST_SIZE(1000)
```

```
Total Training Images in Dataset = (55000, 784)
--------------------------------------------------
x_train Examples Loaded = (5500, 784)
y_train Examples Loaded = (5500, 10)

Total Test Examples in Dataset = (10000, 784)
--------------------------------------------------
x_test Examples Loaded = (1000, 784)
y_test Examples Loaded = (1000, 10)
```

In [5]:

```
# input X and target ouput y
X = tf.placeholder(tf.float32, [None, 784])
y = tf.placeholder(tf.float32, [None, 10])

# the model

# Input layer : note that MNIST data has grayscale images, i.e. single channel
input_layer = tf.reshape(X/255, [-1, 28, 28, 1])

# Convolutional layer #1 : will produce [batch size, 28, 28, 4] because of "same"
conv1 = tf.layers.conv2d(
    inputs=input_layer,
    filters=4,
    kernel_size=[5,5],
    padding="same",
    activation=tf.nn.relu)

# Pooling layer #1 : will produce [batch size, 14, 14, 4]
pool1 = tf.layers.max_pooling2d(
    inputs=conv1,
    pool_size=[2,2],
    strides=2)

# Convolutional layer #2 : will produce [batch size, 14, 14, 8] because of "same"
conv2 = tf.layers.conv2d(
    inputs=pool1,
    filters=8,
    kernel_size=[5,5],
    padding="same",
    activation=tf.nn.relu)

# Pooling layer #2 : will produce [batch size, 7, 7, 8]
pool2 = tf.layers.max_pooling2d(
    inputs=conv2,
    pool_size=[2,2],
    strides=2)

# Dense layer : [batch size, 10]
pool2_flat = tf.reshape(pool2, [-1, 7*7*8])
probabilities = tf.layers.dense(inputs=pool2_flat, activation=tf.nn.softmax, units=10)

# for training: loss and trainer
loss = tf.losses.softmax_cross_entropy(y, probabilities)
train_step = tf.train.GradientDescentOptimizer(0.01).minimize(loss)

# for testing: accuracy
predictions = tf.argmax(probabilities, axis=1)
correct_prediction = tf.equal(predictions, tf.argmax(y, axis=1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
```

In [6]:

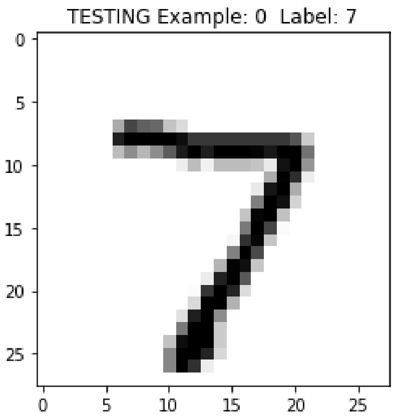
```
# Launch the graph
sess = tf.Session()
sess.run(tf.global_variables_initializer())

# train the model and check the accuracy on the test data at intervals
for i in range(1000):
    sess.run(train_step, feed_dict={X: X_train, y: Y_train})
    if ((i+1)%100 == 0):
        print('After training step : ', i+1)
        print('Accuracy            : ', sess.run(accuracy, feed_dict={X: X_test, y: Y_test}))
```

```
After training step :  100
Accuracy            :  0.086
After training step :  200
Accuracy            :  0.107
After training step :  300
Accuracy            :  0.107
After training step :  400
Accuracy            :  0.107
After training step :  500
Accuracy            :  0.107
After training step :  600
Accuracy            :  0.107
After training step :  700
Accuracy            :  0.107
After training step :  800
Accuracy            :  0.107
After training step :  900
Accuracy            :  0.107
After training step :  1000
Accuracy            :  0.107
```

In [7]:

```
Prediction = sess.run(predictions, feed_dict={X: X_test, y: Y_test})
for i in range(10):
    display_test_digit(i)
    print('Prediction: ', Prediction[i])
    print('=================')
```

```
[ 0.  0.  0.  0.  0.  0.  0.  1.  0.  0.]
```



```
Prediction:  3
-----------------
```

In [8]:

```
sess.close()
```