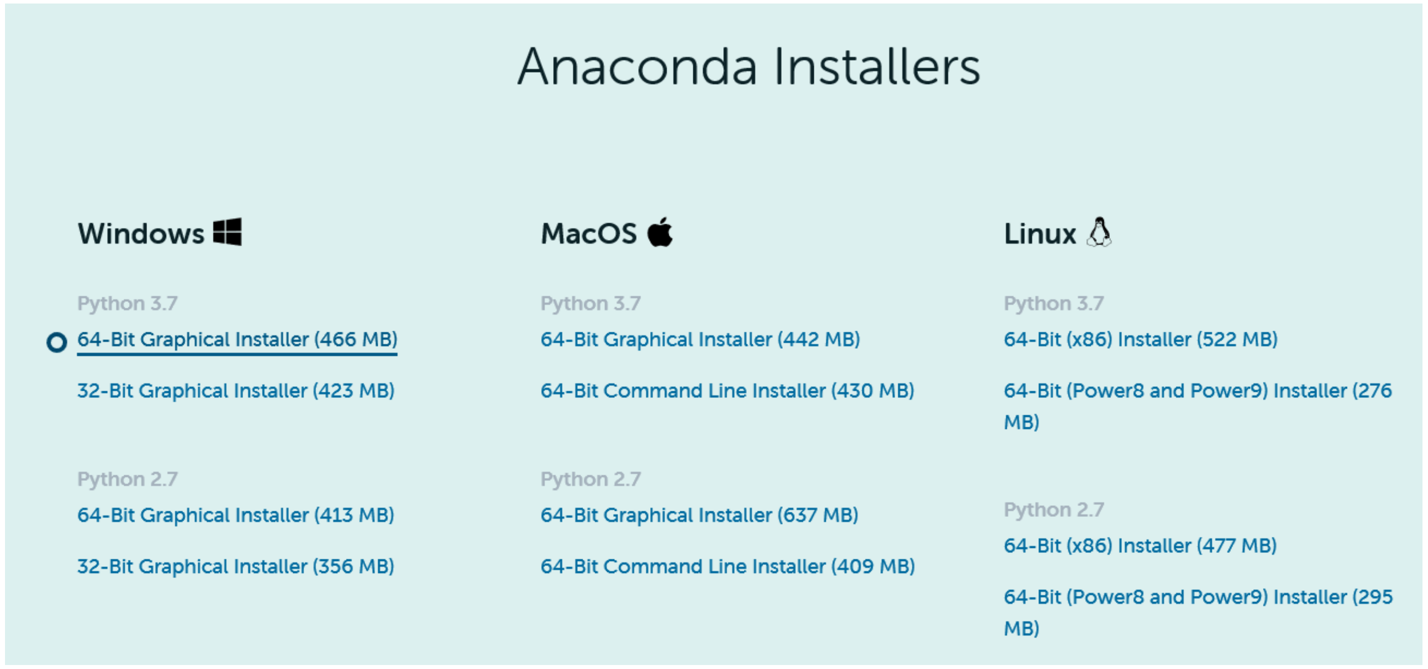
딥러닝(Deep Learning) 실습 CNN

㈜딥비전

DV(주)딥비전

Anaconda 설치

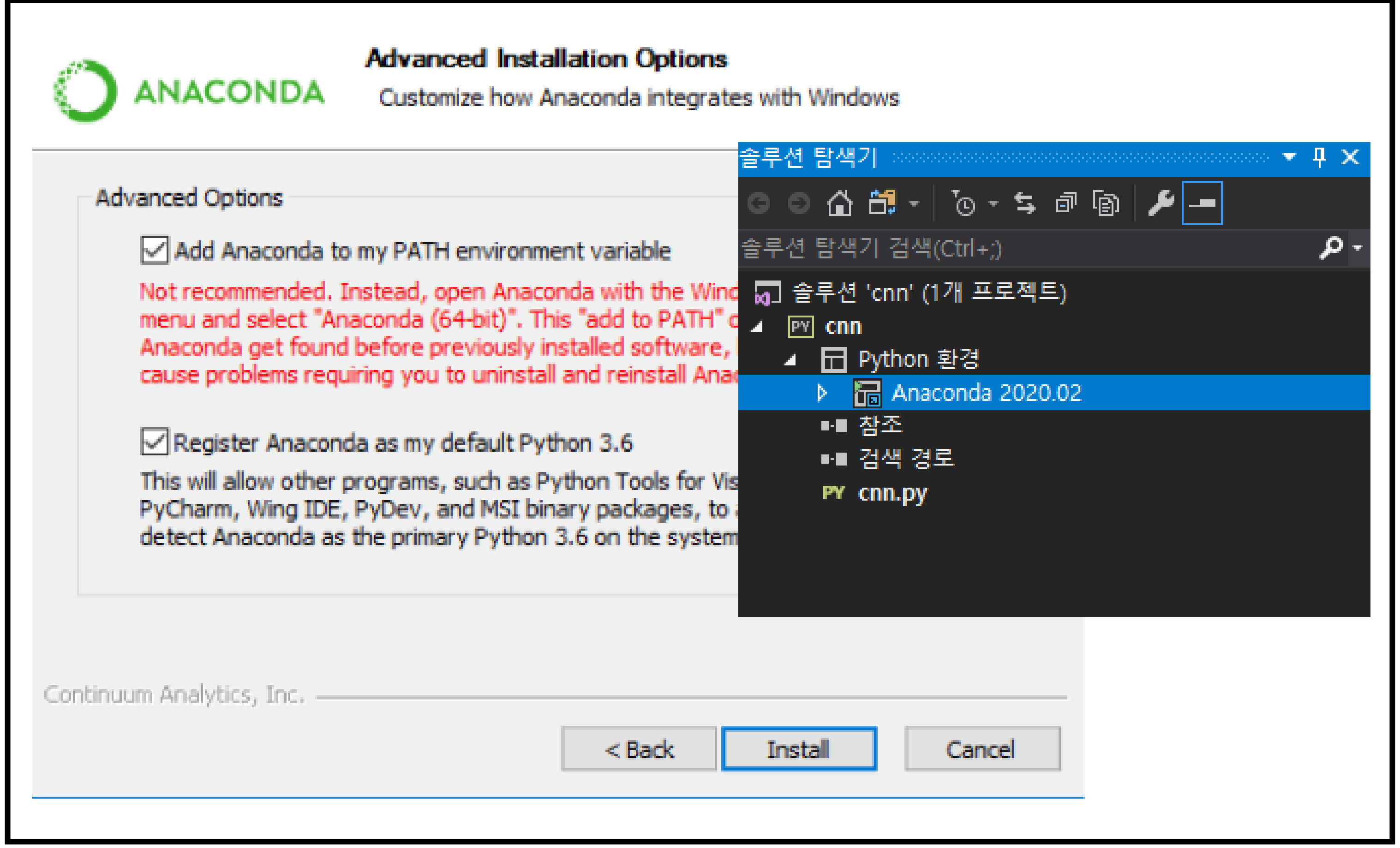
https://www.anaconda.com/products/individual



Anaconda Installers

Windows	MacOS	Linux
Python 3.7	Python 3.7	Python 3.7
64-Bit Graphical Installer (466 MB)	64-Bit Graphical Installer (442 MB)	64-Bit (x86) Installer (522 MB)
32-Bit Graphical Installer (423 MB)	64-Bit Command Line Installer (430 MB)	64-Bit (Power8 and Power9) Installer (276 MB)
Python 2.7	Python 2.7	Python 2.7
64-Bit Graphical Installer (413 MB)	64-Bit Graphical Installer (637 MB)	64-Bit (x86) Installer (477 MB)
32-Bit Graphical Installer (356 MB)	64-Bit Command Line Installer (409 MB)	64-Bit (Power8 and Power9) Installer (295 MB)

Anaconda 설치



Advanced Installation Options
Customize how Anaconda integrates with Windows

Advanced Options

[✓] Add Anaconda to my PATH environment variable

Not recommended. Instead, open Anaconda with the Wind
menu and select "Anaconda (64-bit)". This "add to PATH" o
Anaconda get found before previously installed software, l
cause problems requiring you to uninstall and reinstall Ana

[✓] Register Anaconda as my default Python 3.6

This will allow other programs, such as Python Tools for Vis
PyCharm, Wing IDE, PyDev, and MSI binary packages, to a
detect Anaconda as the primary Python 3.6 on the system

Continuum Analytics, Inc.

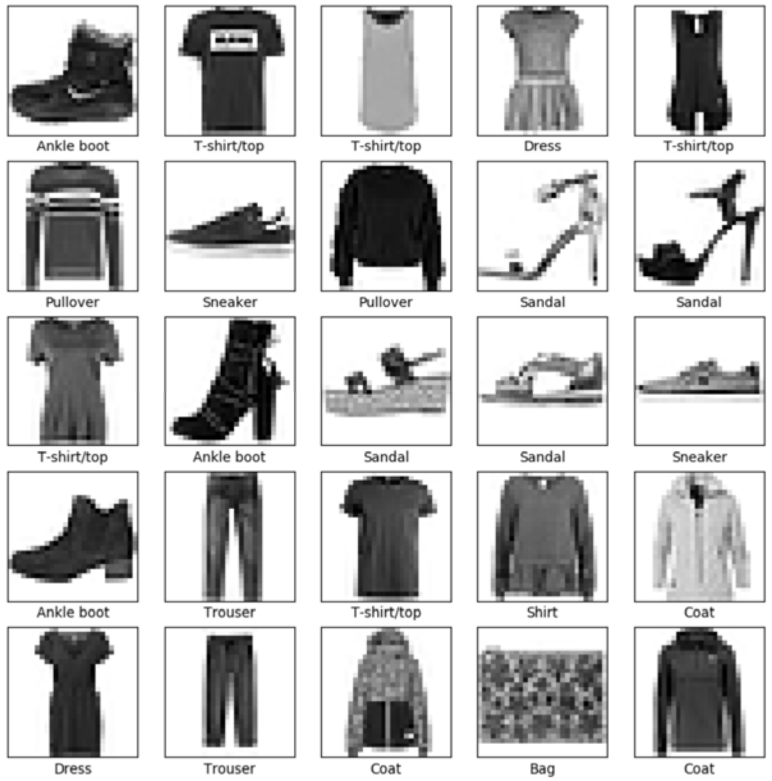
솔루션 탐색기
솔루션 탐색기 검색(Ctrl+;)
솔루션 'cnn' (1개 프로젝트)
cnn
Python 환경
Anaconda 2020.02
참조
검색 경로
cnn.py

1. 딥러닝(Deep Learning) 실습 - Basic Classification

basic classification

Fashion MNIST dataset

training set of 60,000 examples
test set of 10,000 examples
images size 28*28 pixel
labels 10



```
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
```

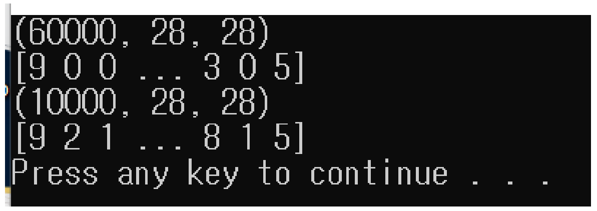
https://www.tensorflow.org/tutorials/

https://www.tensorflow.org/tutorials/keras/basic_classification

1. 딥러닝(Deep Learning) 실습 - Basic Classification

Fashion MNIST database Set

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

fashion_mnist = keras.datasets.fashion_mnist

(train_images, train_labels), (test_images, test_labels) =
fashion_mnist.load_data()

class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']

print(train_images.shape)
print(train_labels)

print(test_images.shape)
print(test_labels)
```



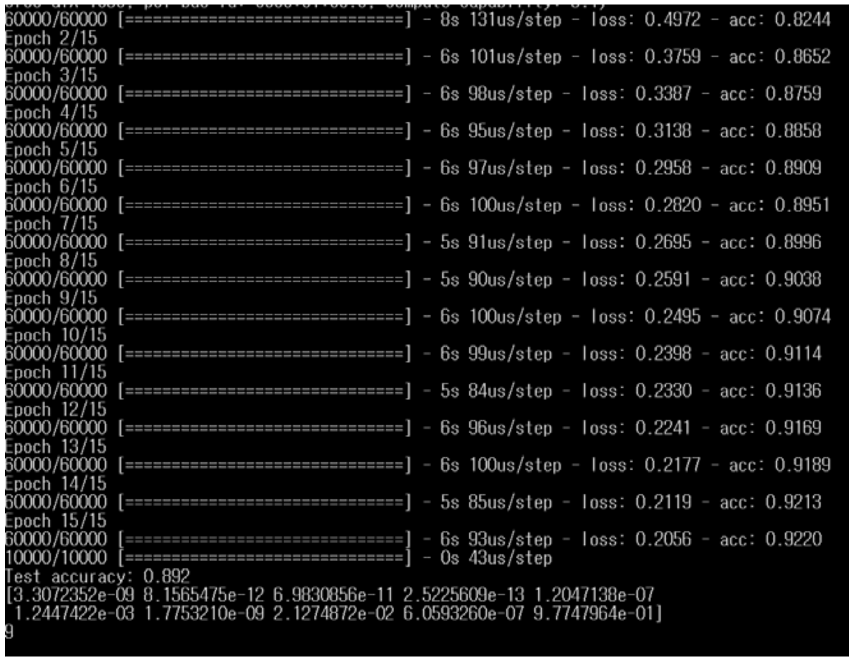
```
(60000, 28, 28)
[9 0 0 ... 3 0 5]
(10000, 28, 28)
[9 2 1 ... 8 1 5]
Press any key to continue . . .
```

(주)딥비전

1. 딥러닝(Deep Learning) 실습 - Basic Classification

weights init, model init, train, test

```python
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(128, activation=tf.nn.relu),
    keras.layers.Dense(10, activation=tf.nn.softmax)
])

model.compile(optimizer=tf.train.AdamOptimizer(),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

train_images = train_images / 255.0
test_images = test_images / 255.0

model.fit(train_images, train_labels, epochs=15)

test_loss, test_acc = model.evaluate(test_images, test_labels)

print('Test accuracy:', test_acc)

predictions = model.predict(test_images)

print(predictions[0])
print(np.argmax(predictions[0]))
```



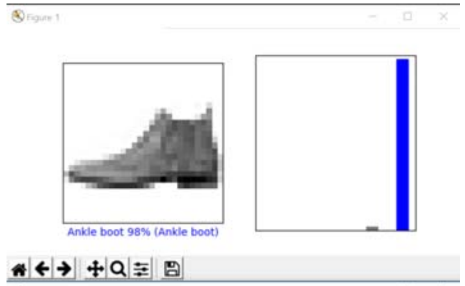
```
[illegible]
60000/60000 [==============================] - 8s 131us/step - loss: 0.4972 - acc: 0.8244
Epoch 2/15
60000/60000 [==============================] - 6s 101us/step - loss: 0.3759 - acc: 0.8652
Epoch 3/15
60000/60000 [==============================] - 6s 98us/step - loss: 0.3387 - acc: 0.8759
Epoch 4/15
60000/60000 [==============================] - 6s 95us/step - loss: 0.3138 - acc: 0.8858
Epoch 5/15
60000/60000 [==============================] - 6s 97us/step - loss: 0.2958 - acc: 0.8909
Epoch 6/15
60000/60000 [==============================] - 6s 100us/step - loss: 0.2820 - acc: 0.8951
Epoch 7/15
60000/60000 [==============================] - 5s 91us/step - loss: 0.2695 - acc: 0.8996
Epoch 8/15
60000/60000 [==============================] - 5s 90us/step - loss: 0.2591 - acc: 0.9038
Epoch 9/15
60000/60000 [==============================] - 6s 100us/step - loss: 0.2495 - acc: 0.9074
Epoch 10/15
60000/60000 [==============================] - 6s 99us/step - loss: 0.2398 - acc: 0.9114
Epoch 11/15
60000/60000 [==============================] - 5s 84us/step - loss: 0.2330 - acc: 0.9136
Epoch 12/15
60000/60000 [==============================] - 6s 96us/step - loss: 0.2241 - acc: 0.9169
Epoch 13/15
60000/60000 [==============================] - 6s 100us/step - loss: 0.2177 - acc: 0.9189
Epoch 14/15
60000/60000 [==============================] - 5s 85us/step - loss: 0.2119 - acc: 0.9213
Epoch 15/15
60000/60000 [==============================] - 6s 93us/step - loss: 0.2056 - acc: 0.9220
10000/10000 [==============================] - 0s 43us/step
Test accuracy: 0.892
[3.3072352e-09 8.1565475e-12 6.9830856e-11 2.5225609e-13 1.2047138e-07
 1.2447422e-03 1.7753210e-09 2.1274872e-02 6.0593260e-07 9.7747964e-01]
9
```

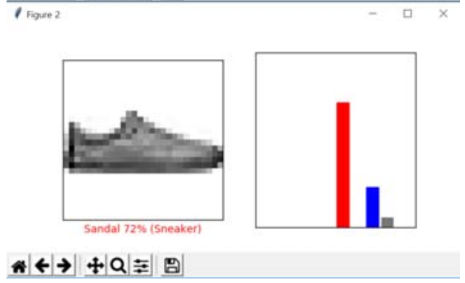
1. 딥러닝(Deep Learning) 실습 - Basic Classification

plot

```python
def plot_image(i, predictions_array, true_label, img):
  predictions_array, true_label, img = predictions_array[i],
true_label[i], img[i]
  plt.grid(False)
  plt.xticks([])
  plt.yticks([])
  plt.imshow(img, cmap=plt.cm.binary)
  predicted_label = np.argmax(predictions_array)
  if predicted_label == true_label:
    color = 'blue'
  else:
    color = 'red'
  plt.xlabel("{} {:2.0f}% ({})".format(class_names[predicted_label],
                                100*np.max(predictions_array),
                                class_names[true_label]),
                                color=color)

def plot_value_array(i, predictions_array, true_label):
  predictions_array, true_label = predictions_array[i], true_label[i]
  plt.grid(False)
  plt.xticks([])
  plt.yticks([])
  thisplot = plt.bar(range(10), predictions_array, color="#777777")
  plt.ylim([0, 1])
  predicted_label = np.argmax(predictions_array)
  thisplot[predicted_label].set_color('red')
  thisplot[true_label].set_color('blue')
```

```python
i = 0
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
plot_image(i, predictions, test_labels, test_images)
plt.subplot(1,2,2)
plot_value_array(i, predictions,  test_labels)

i = 12
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
plot_image(i, predictions, test_labels, test_images)
plt.subplot(1,2,2)
plot_value_array(i, predictions,  test_labels)
plt.show()
```





DV (주)딥비전

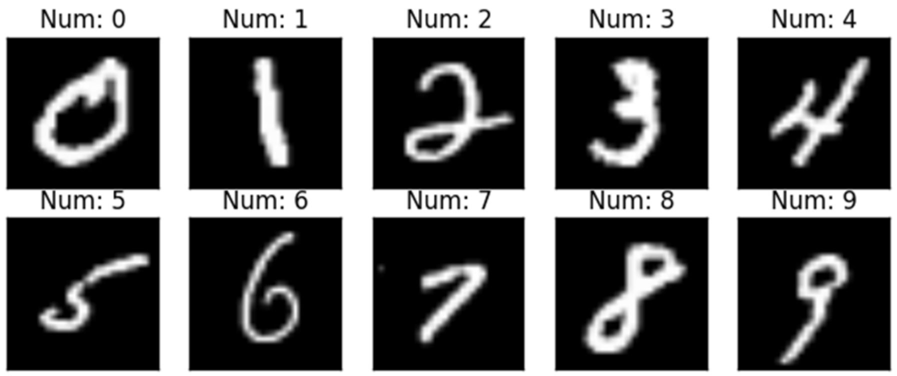
2. 딥러닝(Deep Learning) 실습 - Convolutional Neural Network

Convolutional Neural Network

https://github.com/nlintz/TensorFlow-Tutorials

https://github.com/nlintz/TensorFlow-Tutorials/blob/master/05_convolutional_net.py

MNIST database

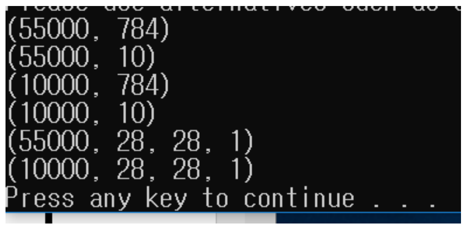


train-images-idx3-ubyte.gz: training set images (9912422 bytes)
train-labels-idx1-ubyte.gz: training set labels (28881 bytes)
t10k-images-idx3-ubyte.gz: test set images (1648877 bytes)
t10k-labels-idx1-ubyte.gz: test set labels (4542 bytes)

2. 딥러닝(Deep Learning) 실습 - Convolutional Neural Network

MNIST database Set

training set of 55,000 examples
test set of 10,000 examples
images size 28*28 pixel
labels 10



```
(55000, 784)
(55000, 10)
(10000, 784)
(10000, 10)
(55000, 28, 28, 1)
(10000, 28, 28, 1)
Press any key to continue . . .
```

```
import tensorflow as tf
import numpy as np
from tensorflow.examples.tutorials.mnist import input_data

mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
trX = mnist.train.images
trY = mnist.train.labels
teX = mnist.test.images
teY = mnist.test.labels

print(trX.shape)
print(trY.shape)
print(teX.shape)
print(teY.shape)

trX = trX.reshape(-1, 28, 28, 1)  # 28x28x1 input img
teX = teX.reshape(-1, 28, 28, 1)  # 28x28x1 input img

print(trX.shape)
print(teX.shape)
```

(주)딥비전

2. 딥러닝(Deep Learning) 실습 - Convolutional Neural Network

weights, model init

```
## init_weights
w = tf.Variable(tf.random_normal([3, 3, 1, 32], stddev=0.01))      # 3x3x1 conv, 32 outputs
w2 = tf.Variable(tf.random_normal([3, 3, 32, 64], stddev=0.01))     # 3x3x32 conv, 64 outputs
w3 = tf.Variable(tf.random_normal([3, 3, 64, 128], stddev=0.01))    # 3x3x32 conv, 128 outputs
w4 = tf.Variable(tf.random_normal([128 * 4 * 4, 625], stddev=0.01)) # FC 128 * 4 * 4 = 2048 inputs, 625 outputs
w_o = tf.Variable(tf.random_normal([625, 10], stddev=0.01))         # FC 625 inputs, 10 outputs (labels)

## init_model
X = tf.placeholder("float", [None, 28, 28, 1])
Y = tf.placeholder("float", [None, 10])
p_keep_conv = tf.placeholder("float")
p_keep_hidden = tf.placeholder("float")

l1a = tf.nn.relu(tf.nn.conv2d(X, w,strides=[1, 1, 1, 1], padding='SAME'))          # l1a shape=(?, 28, 28, 32)
l1 = tf.nn.max_pool(l1a, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')  # l1 shape=(?, 14, 14, 32)
l1 = tf.nn.dropout(l1, p_keep_conv)

l2a = tf.nn.relu(tf.nn.conv2d(l1, w2,  strides=[1, 1, 1, 1], padding='SAME'))       # l2a shape=(?, 14, 14, 64)
l2 = tf.nn.max_pool(l2a, ksize=[1, 2, 2, 1],  strides=[1, 2, 2, 1], padding='SAME') # l2 shape=(?, 7, 7, 64)
l2 = tf.nn.dropout(l2, p_keep_conv)

l3a = tf.nn.relu(tf.nn.conv2d(l2, w3,  strides=[1, 1, 1, 1], padding='SAME'))       # l3a shape=(?, 7, 7, 128)
l3 = tf.nn.max_pool(l3a, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')  # l3 shape=(?, 4, 4, 128)
l3 = tf.reshape(l3, [-1, w4.get_shape().as_list()[0]])                              # reshape to (?, 2048)
l3 = tf.nn.dropout(l3, p_keep_conv)

l4 = tf.nn.relu(tf.matmul(l3, w4))
l4 = tf.nn.dropout(l4, p_keep_hidden)

py_x = tf.matmul(l4, w_o)
```

train, test set & Launch the graph in a session

```
## train, test set
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits=py_x, labels=Y))
train_op = tf.train.RMSPropOptimizer(0.001, 0.9).minimize(cost)
predict_op = tf.argmax(py_x, 1)

batch_size = 8
test_size = 256

## Launch the graph in a session
with tf.Session() as sess:
    # you need to initialize all variables
    tf.global_variables_initializer().run()

    for i in range(15):
        training_batch = zip(range(0, len(trX), batch_size), range(batch_size, len(trX)+1, batch_size))
        for start, end in training_batch:
            sess.run(train_op, feed_dict={X: trX[start:end], Y: trY[start:end],
                                          p_keep_conv: 0.8, p_keep_hidden: 0.5})

        test_indices = np.arange(len(teX)) # Get A Test Batch
        np.random.shuffle(test_indices)
        test_indices = test_indices[0:test_size]

        print(i, np.mean(np.argmax(teY[test_indices], axis=1) ==
                         sess.run(predict_op, feed_dict={X: teX[test_indices],
                                                         p_keep_conv: 1.0,
                                                         p_keep_hidden: 1.0})))
```

감사합니다.

Thank you.