Final Project

- Hallym Univ. 인공지능개론 최종 과제용 파일입니다
- 제출/발표: 2018/6/22 금 12:30 발표
- 조원 1 박상범/ 조원 2 전찬혁
- Data set 개수 : 총 3007개
- 촬영 장소 및 일시
    - 산악 협력관
    - 2018/6/21

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import tensorflow as tf
from tensorflow.python.framework import ops
from dd_nnutil_hallym1 import *
import random
import time
```

Dataset 업로드 확인

```
folder1="./dataset2"
```

```
import os
sorted(os.listdir(folder1))
```

```
['0', '1', '2', '3', '4', '5']
```

예상되는 출력:

['0', '1', '2', '3', '4', '5']

위와 다르게 출력이 된다면 애초에 상위 폴더이름이 dataset1 인지 확인해보세요.

Data 검토해보기

Data의 종류. 현재는 6가지의 data를 사용하므로 6을 사용합니다.

```
nclasses = 6
```

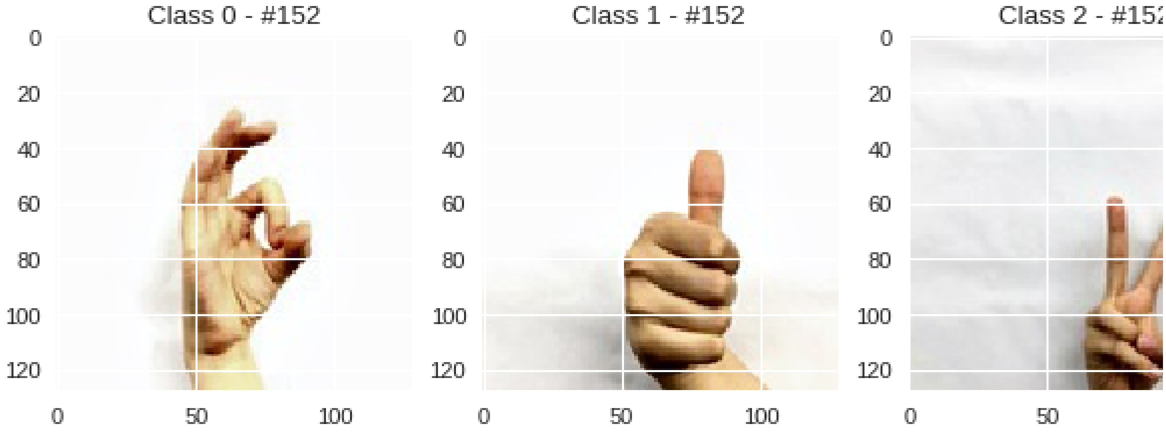
그려볼 이미지 index. 아래의 숫자를 바꾸어 가며 아래 cell에 그림이 제대로 표시가 되는지 확인해보세요. **최종적으로는 직접 생성한 이미지가 display되도록 idx를 설정하세요**

```
idx= 152
```

```
fig, ax = plt.subplots(figsize=(20, 10), dpi=80)
for i in range(nclasses):
```

```
    img1, ntot = load_image_test(folder=folder1, img_class=i, idx=idx)
    print('class', i, '--', ntot)
    img1c = centered_crop(img1, output_side_length=128)
    plt.subplot(1,nclasses,i+1)
    plt.title('Class {} - #{}'.format(i, idx))
    plt.imshow(img1c)
```

```
('class', 0, '--', 500)
('class', 1, '--', 500)
('class', 2, '--', 500)
('class', 3, '--', 500)
('class', 4, '--', 500)
('class', 5, '--', 500)
```



Hyper-parameters

```
learning_rate=0.01
num_epochs=50
minibatch_size=32
```

본격적으로 시작 - Data 로드 하기

```
X_train, Y_train_orig, X_test, Y_test_orig = ₩
    load_dataset(folder=folder1, nclasses=nclasses)
```

```
('./dataset2/0/*.JPG', '-->', (500,))
('./dataset2/1/*.JPG', '-->', (500,))
('./dataset2/2/*.JPG', '-->', (500,))
('./dataset2/3/*.JPG', '-->', (500,))
('./dataset2/4/*.JPG', '-->', (500,))
('./dataset2/5/*.JPG', '-->', (500,))
```

일반 숫자를 one-hot encoding으로 !

```
Y_train = convert_to_one_hot(Y_train_orig, nclasses).T
Y_test = convert_to_one_hot(Y_test_orig, nclasses).T
```

Data shape 살펴보기. Dimension을 살펴보세요

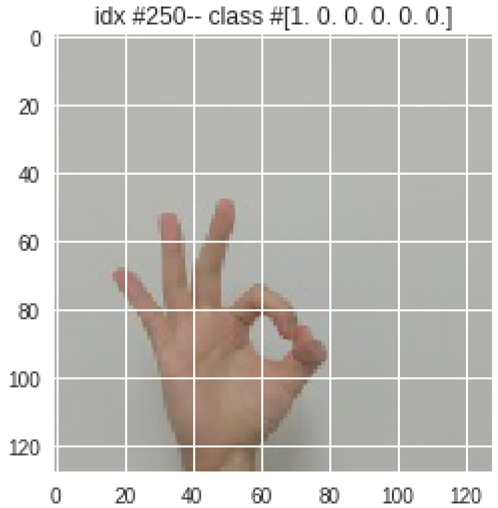
```
print ("number of training examples = " + str(X_train.shape[0]))
print ("number of test examples = " + str(X_test.shape[0]))
print ("X_train shape: " + str(X_train.shape))
```

```
print ("Y_train shape: " + str(Y_train.shape))
print ("X_test shape: " + str(X_test.shape))
print ("Y_test shape: " + str(Y_test.shape))
```

```
number of training examples = 2100
number of test examples = 900
X_train shape: (2100, 128, 128, 3)
Y_train shape: (2100, 6)
X_test shape: (900, 128, 128, 3)
Y_test shape: (900, 6)
```

```
# display images
idx1 = 250
x1 = X_train[idx1]
y1 = Y_train[idx1]

plt.figure()
plt.imshow(x1)
plt.title('idx #{}-- class #{}'.format(idx1, y1))
```

```
Text(0.5,1,u'idx #250-- class #[1. 0. 0. 0. 0. 0.]')
```



위의 idx1 값을 바꾸어보며 테스트해보세요. **마찬가지로 본인들의 사진이 나오도록 최종 idx1 값을 설정하세요**

Placeholders 만들기

```
def create_placeholders(n_H0, n_W0, n_C0, n_y):
    X = tf.placeholder(tf.float32, [None, n_H0, n_W0, n_C0])
    Y = tf.placeholder(tf.float32, [None, n_y])

    return X, Y
```

Parameter 초기화 하기

- W1, W2의 크기, 갯수를 원하는 대로 변경하세요.
- W3, W4.. 등이 필요한 경우 자유롭게 넣어보세요

```
def initialize_parameters():
    W1 = tf.get_variable("W1", [4, 4, 3, 4], initializer=tf.contrib.layers.xavier_initializ
    W2 = tf.get_variable("W2", [2, 2, 4, 8], initializer=tf.contrib.layers.xavier_initializ
    #W3 = ...  #필요하면 추가해보세요
    parameters = {"W1": W1,
                  "W2": W2}
```

```
    return parameters
```

Forward propagation

- 아래는 다음과 같은 ConvNet을 구현한 것입니다. 자유롭게 stride, pooling의 ksize 변경해 보세요.

    CONV2D -> RELU -> MAXPOOL -> CONV2D -> RELU -> MAXPOOL -> FLATTEN -> FC

- CONV2D -> RELU -> MAXPOOL 을 하나의 덩어리로 생각하면 좋습니다

```
def forward_propagation(X, parameters):
    W1 = parameters['W1']
    W2 = parameters['W2']

    # CONV2D: stride of 1, padding 'SAME'
    Z1 = tf.nn.conv2d(X, W1, strides=[1, 1, 1, 1], padding='SAME')
    # RELU
    A1 = tf.nn.relu(Z1)
    # MAXPOOL: window 8x8, sride 8, padding 'SAME'
    P1 = tf.nn.max_pool(A1, ksize=[1, 8, 8, 1], strides=[1, 8, 8, 1], padding='SAME')

    # CONV2D: filters W2, stride 1, padding 'SAME'
    Z2 = tf.nn.conv2d(P1, W2, strides=[1, 1, 1, 1], padding='SAME')
    # RELU
    A2 = tf.nn.relu(Z2)
    # MAXPOOL: window 4x4, stride 4, padding 'SAME'
    P2 = tf.nn.max_pool(A2, ksize=[1, 4, 4, 1], strides=[1, 4, 4, 1], padding='SAME')

    # FLATTEN
    P2 = tf.contrib.layers.flatten(P2)
    # FULLY-CONNECTED without non-linear activation function (not not call softmax).
    # 6 neurons in output layer. Hint: one of the arguments should be "activation_fn=None"
    Z3 = tf.contrib.layers.fully_connected(P2, 6, activation_fn=None)
    Y_hat = Z3

    return Y_hat
```

Cost 계산하기

```
def compute_cost(Y_hat, Y):
     # cost = tf.nn.softmax_cross_entropy_with_logits(logits = Y_hat, labels = Y)
    cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits_v2(logits=Y_hat, labels=Y
    return cost
```

Model

지금까지 함수로 만들었던 기능들을 불러 만들어 네트워크를 구성합니다

```
(m, n_H0, n_W0, n_C0) = X_train.shape
n_y = Y_train.shape[1]
```

```
ops.reset_default_graph()
X, Y = create_placeholders(n_H0, n_W0, n_C0, n_y)
parameters = initialize_parameters()
Y_hat = forward_propagation(X, parameters)
cost = compute_cost(Y_hat, Y)
optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate).minimize(cost)
```

Start the session to compute the tensorflow graph

```
print_cost = True


vcosts = []  # cost를 저장할 빈 list
vtime = [] # 연산시간을 기록할 빈 list


# Reset the graph
#tf.reset_default_graph()

# Start interactive session
sess = tf.InteractiveSession()


seed = 0
sess.run(tf.global_variables_initializer())
# Do the training loop
for epoch in range(num_epochs):

    minibatch_cost = 0.
    num_minibatches = int(m / minibatch_size)  # number of minibatches of size minibatch_si
    seed = seed + 1
    minibatches = random_mini_batches(X_train, Y_train, minibatch_size, seed=seed)

    for minibatch in minibatches:
        # Select a minibatch
        (minibatch_X, minibatch_Y) = minibatch
        # IMPORTANT: The line that runs the graph on a minibatch.
        # Run the session to execute the optimizer and the cost, the feedict should contain

        #_, temp_cost = sess.run([optimizer, cost], feed_dict={X: minibatch_X, Y: minibatch
        t0 = time.time()
        sess.run(optimizer, feed_dict={X: minibatch_X, Y: minibatch_Y})
        t_elapsed = time.time() - t0
        vtime.append(t_elapsed) # 시간을 측정하고 이를 list에 저장함 (append)

        temp_cost = sess.run(cost, feed_dict={X: minibatch_X, Y: minibatch_Y})

        minibatch_cost += temp_cost / num_minibatches

    # Print the cost every epoch
    if print_cost == True and epoch % 5 == 0:
        print("Cost after epoch %i: %f" % (epoch, minibatch_cost))

    vcosts.append(minibatch_cost)
```

```
Cost after epoch 0: 1.827218
Cost after epoch 5: 0.173040
Cost after epoch 10: 0.066017
Cost after epoch 15: 0.063294
Cost after epoch 20: 0.044596
Cost after epoch 25: 0.022372
Cost after epoch 30: 0.021404
Cost after epoch 35: 0.023178
Cost after epoch 40: 0.016705
Cost after epoch 45: 0.003797
```

```
correct_prediction = tf.equal(tf.argmax(Y_hat, 1), tf.argmax(Y, 1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, "float"))

train_accuracy = accuracy.eval({X: X_train, Y: Y_train})
test_accuracy = accuracy.eval({X: X_test, Y: Y_test})
print("Train Accuracy:", train_accuracy)
```

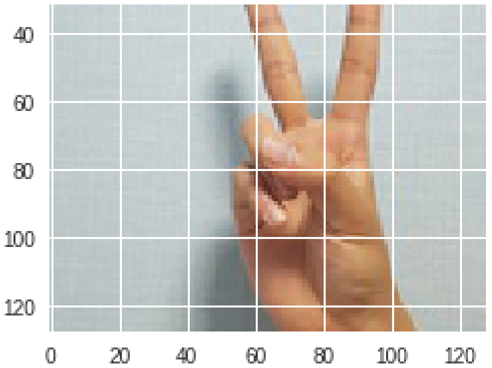
```
print("Test Accuracy:", test_accuracy)

print("Mean time to train for each batch: {:.3f} sec / batch size : {}".format(np.mean(vtim
```

...

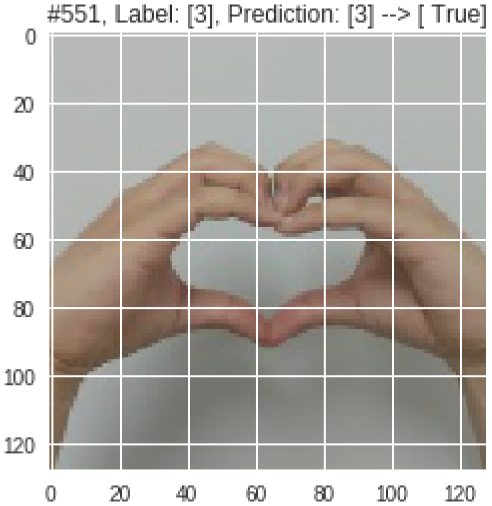
학습된 모델로 예측해보기 (correct predictions)

- 10개 테스트 해보기 --> 자유롭게 원하는 대로 변경하여 테스트 해보세요

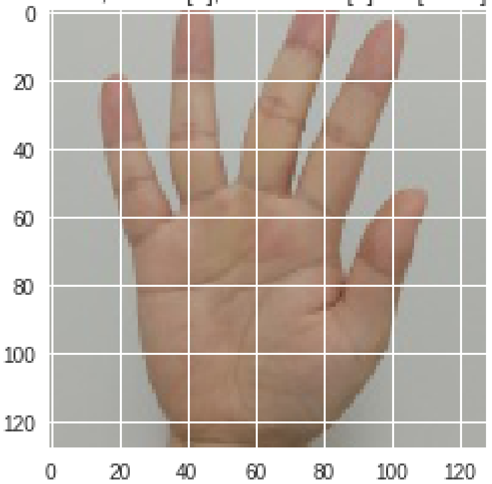
```
ntest = 10


for j in range(0,ntest):
    # Get one and predict
    r = random.randint(0, X_test.shape[0] - 1)
    #print('Picked {} / {}'.format(r, X_test.shape[0]))

    v1 = sess.run(tf.argmax(Y_test[r:r+1], 1))
    t0 = time.time()
    v2 = sess.run(tf.argmax(Y_hat, 1), feed_dict={X: X_test[r:r+1]})
    t_elapsed = time.time() - t0
    bok = v1 == v2
    str1 = '#{}, Label: {}, Prediction: {}, {}, Time : {:.3f} sec'.format(r, v1, v2,bok, t_
    print(str1)
    str2 = '#{}, Label: {}, Prediction: {} --> {}'.format(r, v1, v2, bok)
    plt.figure()
    plt.imshow(X_test[r])
    plt.title(str2)


sess.close()
```



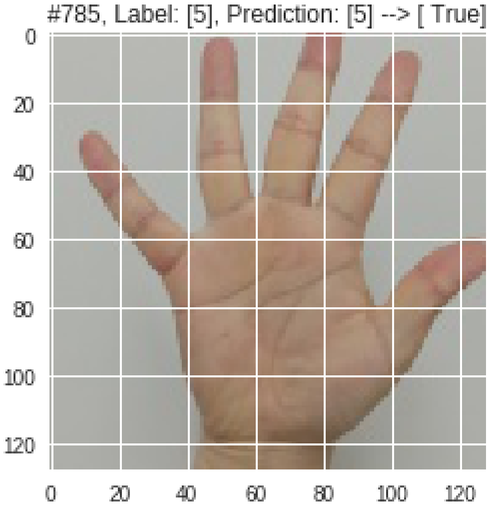
#551, Label: [3], Prediction: [3] --> [True]



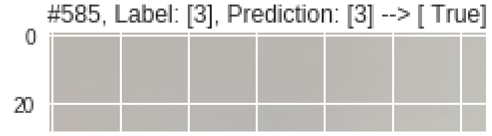
#893, Label: [5], Prediction: [5] --> [True]

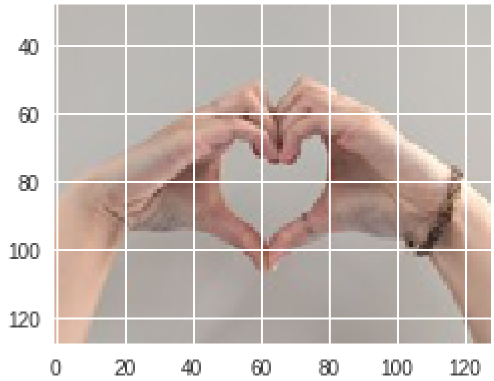


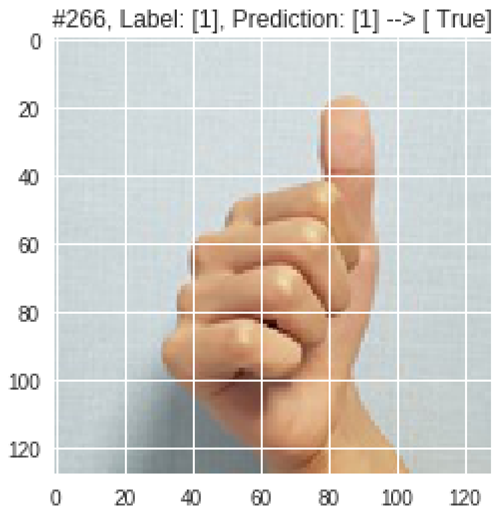
#785, Label: [5], Prediction: [5] --> [True]

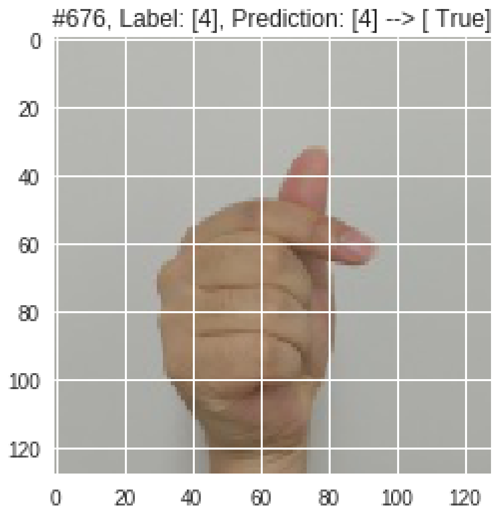


#585, Label: [3], Prediction: [3] --> [True]





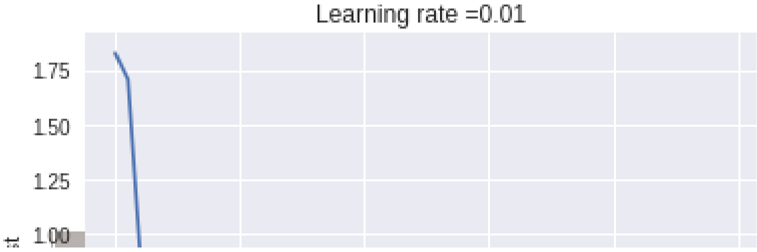




Cost 그려보기

```
plt.figure()
plt.plot(np.squeeze(vcosts))
plt.ylabel('cost')
plt.xlabel('iterations (per tens)')
plt.title("Learning rate =" + str(learning_rate))
```

```
Text(0.5,1,u'Learning rate =0.01')
```



Cost 의 초반부 확대해서 그려보기

```
plt.figure()
plt.plot(np.squeeze(vcosts[:20]))
plt.ylabel('cost')
plt.xlabel('iterations (per tens)')
plt.title("Learning rate =" + str(learning_rate))
plt.show()
```