Hand pose classifier ¶

- 한림대학교 딥러닝이해및응용(7210231) 과목 자료입니다
- Hallym Univ.
- Deeplearning
- 2018/11/13

In [1]:

```
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import tensorflow as tf
from tensorflow.python.framework import ops
from dd_nnutil_hallym3 import *
import random
import time
```

In [3]:

```
!pip install pillow
```

```
Requirement already satisfied: pillow in /home/seung/.venv/py3Keras/li
b/python3.5/site-packages (5.2.0)
You are using pip version 18.0, however version 18.1 is available.
You should consider upgrading via the 'pip install --upgrade pip' comm
and.
```

결과 저장을 위한 폴더 생성

In [4]:

```
import os
directory_out = 'out'
if not os.path.exists(directory_out):
    os.makedirs(directory_out)
```

Dataset 업로드 확인

!ls !pwd

In [5]:

```
!dir
```

```
7210231_2018Nov_handpose_v3.ipynb  handpose_small   __pycache__
dd_nnutil_hallym3.py               out
```

In [6]:

```
folder1 ="./data"
```

In [7]:

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

Out[7]:

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

예상되는 출력:

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

or

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

위와 다르게 출력이 된다면 애초에 상위 폴더이름이 handpose_small 인지 확인해보세요.

Data 검토해보기

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

In [8]:

```
nclasses = 6
```

그려볼 이미지 index. 아래의 숫자를 바꾸어 가며 아래 cell에 그림이 제대로 표시가 되는지 확인해보세요. 최종적으로는 직접 생성한 이미지가 **display**되도록 **idx**를 설정하세요

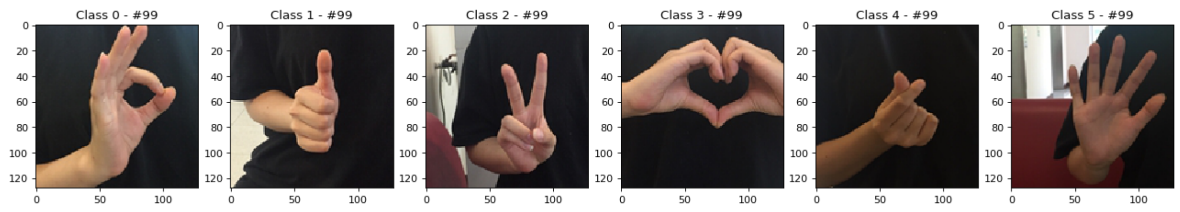
In [9]:

```
idx= 99
```

In [10]:

```
fig, ax = plt.subplots(figsize=(20, 10), dpi=80)
for i in range(nclasses):
    img1, ntot = load_image_test(folder=folder1, img_class=i, suffix='_small', 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 -- 100
class 1 -- 100
class 2 -- 100
class 3 -- 100
class 4 -- 100
class 5 -- 100
```



Hyper-parameters

In [11]:

```
learning_rate=0.005
num_epochs=50
minibatch_size=128
```

본격적으로 시작 - Data 로드 하기

In [12]:

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

```
./handpose_small/0_small/*.JPG --> (100,)
./handpose_small/1_small/*.JPG --> (100,)
./handpose_small/2_small/*.JPG --> (100,)
./handpose_small/3_small/*.JPG --> (100,)
./handpose_small/4_small/*.JPG --> (100,)
./handpose_small/5_small/*.JPG --> (100,)
```

In [13]:

```
print(X_train.shape)
print(Y_train_orig.shape)
```

```
(420, 128, 128, 3)
(420,)
```

일반 숫자를 one-hot encoding으로 !

In [14]:

```
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

In [15]:

```
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 = 420
number of test examples = 180
X_train shape: (420, 128, 128, 3)
Y_train shape: (420, 6)
X_test shape: (180, 128, 128, 3)
Y_test shape: (180, 6)
```

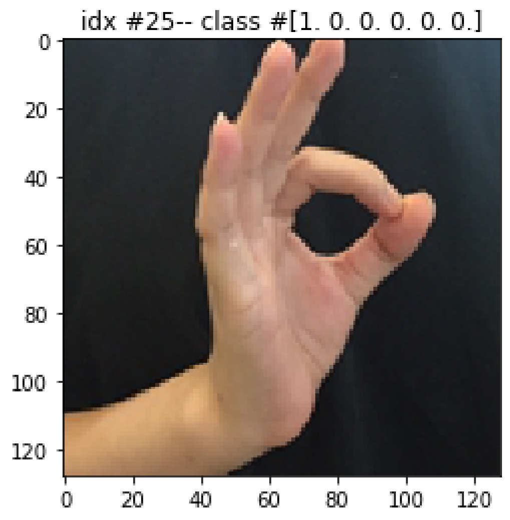
In [16]:

```
# display images
idx1 = 25
x1 = X_train[idx1]
y1 = Y_train[idx1]

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

Out[16]:

```
Text(0.5,1,'idx #25-- class #[1. 0. 0. 0. 0. 0.]')
```



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

Placeholders 만들기

In [17]:

```
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

In [18]:

```
def initialize_parameters():
    W1 = tf.get_variable("W1", [3, 3, 3, 16], initializer=tf.contrib.layers.xavier_
    W2 = tf.get_variable("W2", [2, 2, 16, 32], initializer=tf.contrib.layers.xavier
    #W3 = ...  #필요하면 추가해보세요
    parameters = {"W1": W1,
                  "W2": W2}

    return parameters
```

Forward propagation

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

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

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

In [19]:

```
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_f
    Z3 = tf.contrib.layers.fully_connected(P2, 6, activation_fn=None)
    Y_hat = Z3

    return Y_hat
```

Cost 계산하기

In [20]:

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

Model

지금까지 함수로 만들었던 기능들을 불러 만들어 네트워크를 구성합니다

In [21]:

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

In [22]:

```
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

In [23]:

```
print_cost = True
```

In [24]:

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

In [25]:

```
# Reset the graph
#tf.reset_default_graph()

# Start interactive session
sess = tf.InteractiveSession()
```

In [26]:

```
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 mini
    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

        #_, temp_cost = sess.run([optimizer, cost], feed_dict={X: minibatch_X, Y: m
        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 {}\t{}".format(epoch, minibatch_cost))

    vcosts.append(minibatch_cost)
```

```
Cost after epoch 0      2.078949451446533
Cost after epoch 5      0.0302955973893404
Cost after epoch 10     0.0005268128006719053
Cost after epoch 15     0.00012399681266591264
Cost after epoch 20     7.010720946709625e-05
Cost after epoch 25     5.143407239908508e-05
Cost after epoch 30     5.071608211437706e-05
Cost after epoch 35     5.0524071411928155e-05
Cost after epoch 40     5.080128842867756e-05
Cost after epoch 45     4.1333974271158994e-05
```

In [27]:

```
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[:500], Y: Y_train[:500]})
test_accuracy = accuracy.eval({X: X_test[:500], Y: Y_test[:500]})
print("Train Accuracy:", train_accuracy)
print("Test Accuracy:", test_accuracy)

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

```
Train Accuracy: 1.0
Test Accuracy: 1.0
Mean time to train for each batch: 0.025 sec / batch size : 128
```

학습된 모델로 예측해보기 (correct predictions)

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

In [28]:

```
ntest = -1
if ntest==-1:
    ntest = X_test.shape[0]
print('number of test images: {}'.format(ntest))
```

```
number of test images: 180
```

In [29]:

```python
for j in range(0,ntest):
    # Get one and predict
    if ntest == X_test.shape[0]:
        r = j
    else:
        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]})
    v2raw = sess.run(Y_hat, feed_dict={X: X_test[r:r+1]})
    v2 = np.argmax(v2raw)
    #print(v2raw, '----', v2)
    t_elapsed = time.time() - t0
    bok = (v1 == v2)
    str1 = '#{}, Label: {}, Pred: {}, {}, Time : {:.3f} sec'.format(r, v1, v2,bok,
    if ntest < X_test.shape[0]:
        print(str1)
    else:
        if j % 100 ==0:
            print(str1)
    if not bok:
        str2 = '#{}, Label: {}, Pred: {} --> {}\n{}'.format(r, v1, v2, bok, v2raw)
    else:
        str2 = '#{}, Label: {}, Pred: {} --> {}'.format(r, v1, v2, bok)
    #print(str2)
    plt.figure()
    plt.imshow(X_test[r])

    plt.title(str2)


    if not bok:
        filename_img = '{}/image_test_{}_fail.png'.format(directory_out, j)
    else:
        filename_img = '{}/image_test_{}.png'.format(directory_out, j)
    plt.savefig(filename_img)
    plt.close()
```

```
#0, Label: [0], Pred: 0, [ True], Time : 0.008 sec
#100, Label: [3], Pred: 3, [ True], Time : 0.001 sec
```

분류가 잘된 예, 잘못된 예 각각 10가지 이상을 찾아서, 보고서에 제출하세요

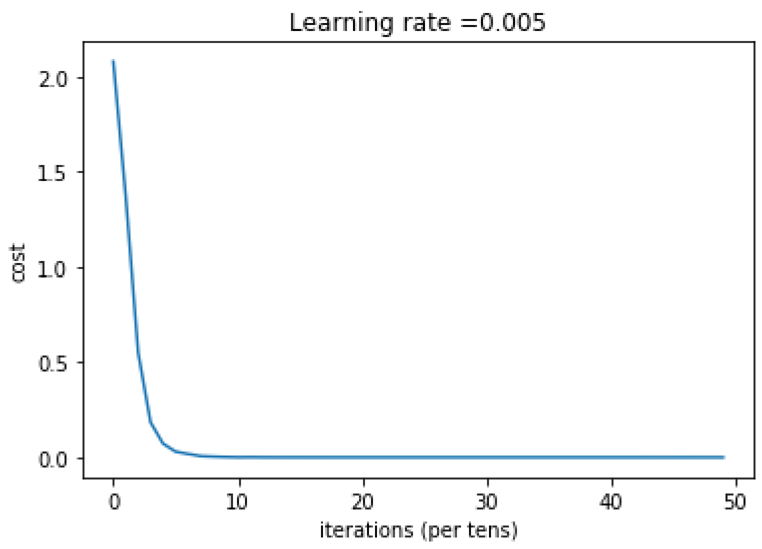
Cost 그려보기

In [30]:

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

Out[30]:

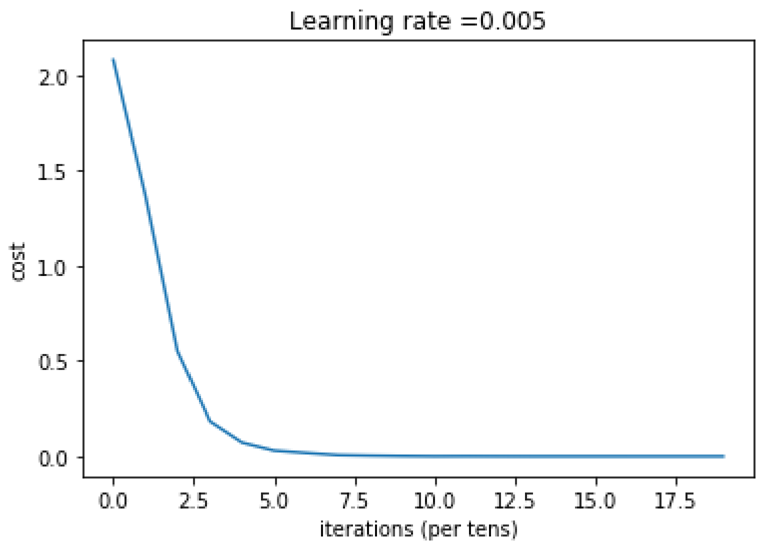
```
Text(0.5,1,'Learning rate =0.005')
```



Cost 의 초반부 확대해서 그려보기

In [31]:

```
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()
```



In [32]:

```
#sess.close()
```