In [1]:

```
!pip install opencv-python
```

```
Collecting opencv-python
  Downloading https://files.pythonhosted.org/packages/1f/51/e0b9cef23098bc31c77b0e06221
dd8d05119b9782d4c2b1d1482e22b5f5e/opencv_python-4.1.1.26-cp37-cp37m-win_amd64.whl (http
s://files.pythonhosted.org/packages/1f/51/e0b9cef23098bc31c77b0e06221dd8d05119b9782d4c2
b1d1482e22b5f5e/opencv_python-4.1.1.26-cp37-cp37m-win_amd64.whl) (39.0MB)
Requirement already satisfied: numpy>=1.14.5 in c:₩python₩anaconda3₩lib₩site-packages
 (from opencv-python) (1.16.1)
Installing collected packages: opencv-python
Successfully installed opencv-python-4.1.1.26
```

In [90]:

```
import cv2
import matplotlib.pyplot as plt
import os
import numpy as np
import pandas as pd
```

In [91]:

```
# plt.imshow(img)
```

In [92]:

```
# import keras
# import tensorflow as tf
# import numpy as np
```

dataset

이미지 파일들을 학습하기 위해서는 해당 이미지 파일들을 숫자 배열로 저장해야합니다. 당연히 여기서 숫자 배열을 픽셀을 의미합니다.

저 같은 경우 카테고리 별로 cnn_sample 폴더 내에 카테고리별로 폴더를 생성하고 각각 이미지들을 정리하였습니다.

소스 코드 내 categories 변수는 카테고리 명뿐만 아니라 폴더를 찾는 용도로 사용하기 때문에 해당 class의 이미지들이 들어가 있는 폴더명을 넣어주시면 됩니다.

In [93]:

```
groups_folder_path = './cnn_sample/cnn_samples/'
categories = [ "red", "green"]
num_classes = len(categories)
```

In [94]:

```
image_w = 32
image_h = 32

X = []
Y = []

for idex, categorie in enumerate(categories):
    label = [0 for i in range(num_classes)]
    label[idex] = 1
    image_dir = groups_folder_path + categorie + '/'

    for top, dir, f in os.walk(image_dir):
        for filename in f:
            print(image_dir+filename)
            img = cv2.imread(image_dir+filename)
            img = cv2.resize(img, None, fx=image_w/img.shape[0], fy=image_h/img.shape[1
            X.append(img/255)
            Y.append(label)
X = []
Y = []
```

In [95]:

```
# img = cv2.imread(image_dir+filename)
# img = cv2.resize(img, None, fx=image_w/img.shape[0], fy=image_h/img.shape[1])
# X.append(img/255)
# Y.append(label)
```

In [96]:

```
import os, re, glob
import cv2
import numpy as np
from sklearn.model_selection import train_test_split

groups_folder_path = './cnn_sample/cnn_sample/'
categories = ["red", "green"]

num_classes = len(categories)

image_w = 32
image_h = 32

X = []
Y = []

for idex, categorie in enumerate(categories):
    label = [0 for i in range(num_classes)]
    label[idex] = 1
    image_dir = groups_folder_path + categorie + '/'

    for top, dir, f in os.walk(image_dir):

        for filename in f:
            print(image_dir+filename)
            img = cv2.imread(image_dir+filename)
            img = cv2.resize(img, None, fx=image_w/img.shape[1], fy=image_h/img.shape[0
            X.append(img/255)
            Y.append(label)
X = np.array(X)
Y = np.array(Y)

# X_train, X_test, Y_train, Y_test = train_test_split(X,Y)
# xy = (X_train, X_test, Y_train, Y_test)

# np.save("./img_data.npy", xy)
```

```
./cnn_sample/cnn_sample/red/1.image.png
./cnn_sample/cnn_sample/red/1.jpg
./cnn_sample/cnn_sample/red/1.tomatoes_c_Kanawa_Studio-iStock-GettyImages-116331737
4-LEDE.jpg
./cnn_sample/cnn_sample/red/10.jpg
./cnn_sample/cnn_sample/red/10.tomatoes-406-1.jpg
./cnn_sample/cnn_sample/red/10.tomatoes.jpg
./cnn_sample/cnn_sample/red/12.redtomato-370x229.jpg
./cnn_sample/cnn_sample/red/127_100.jpg
./cnn_sample/cnn_sample/red/13.%ED%86%A0%EB%A7%88%ED%86%A0%ED%95%98%EB%A9%B4-%EB%B
9%A0%EC%A7%88-%EC%88%98-%EC%97%86%EB%8A%94-%EB%9D%BC%EC%9D%B4%EC%BD%94%ED%8E%9C.png
./cnn_sample/cnn_sample/red/13.tomatoes-close-up.jpg
./cnn_sample/cnn_sample/red/144_100.jpg
./cnn_sample/cnn_sample/red/155_100.jpg
./cnn_sample/cnn_sample/red/16.3a9fc9a51f6f4e47a9ebd36bf30a4604.jpg
./cnn_sample/cnn_sample/red/17.tomatoes_helios4eos_gettyimages-edit.jpeg
./cnn_sample/cnn_sample/red/19.thumb_d_E8FF5517608DC4A792822212F47C7052.jpg
./cnn_sample/cnn_sample/red/19.tomatoes.jpg
./cnn_sample/cnn_sample/red/195_100.jpg
./cnn_sample/cnn_sample/red/2.120_shop1_282762.jpg
```

In [40]:

```
Y
```

Out[40]:

```
array([[1, 0],
       [1, 0],
       [1, 0],
       [1, 0],
       [1, 0],
       [1, 0],
       [1, 0],
       [1, 0],
       [1, 0],
       [1, 0],
       [1, 0],
       [1, 0],
       [1, 0],
       [1, 0],
       [1, 0],
       [1, 0],
       [1, 0],
       [1, 0],
       [1, 0],
       [1, 0]
```

In [97]:

```
import keras
import tensorflow as tf
```

In [99]:

```
num_classes = 2
```

In [100]:

```
label_names = ["red", "green"]
```

In [101]:

```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y)
```

In [102]:

```
print(X_train.shape)
print(X_test.shape)
print(Y_train.shape)
print(Y_test.shape)
```

```
(285, 32, 32, 3)
(96, 32, 32, 3)
(285, 2)
(96, 2)
```

In [104]:

```
X_train = X_train.reshape(285, 32,32,3).astype('float32')
X_test = X_test.reshape(96, 32,32,3).astype('float32')
```

In [105]:

```
Y_train
```

Out[105]:

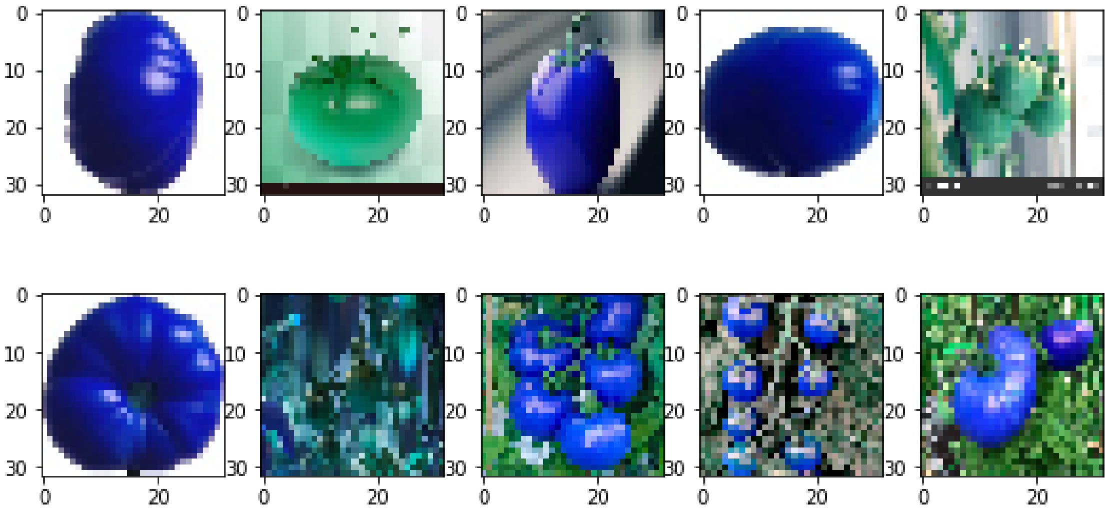
```
array([[1, 0],
       [0, 1],
       [1, 0],
       [1, 0],
       [0, 1],
       [1, 0],
       [0, 1],
       [1, 0],
       [1, 0],
       [1, 0],
       [0, 1],
       [0, 1],
       [1, 0],
       [0, 1],
       [0, 1],
       [0, 1],
       [1, 0],
       [0, 1],
       [0, 1],
```

In [106]:

```
print('label={}'.format(Y_train[:10,0]))
fig, ax = plt.subplots(2,5,figsize=(10,5))

for i in range(5):
    ax[0][i].imshow(np.reshape(X_train[i], (32,32,3)))
    ax[1][i].imshow(np.reshape(X_train[i+5], (32,32,3)))
```

```
label=[1 0 1 1 0 1 0 1 1 1]
```



In [107]:

```
X = tf.placeholder(tf.float32, [None, 32, 32, 3])
Y = tf.placeholder(tf.float32, [None,2])
keep_prob = tf.placeholder(tf.float32)
```

In [108]:

```
W1 = tf.Variable(tf.random_normal([3,3,3,32], stddev=0.01))
L1 = tf.nn.conv2d(X, W1, strides=[1,1,1,1], padding='SAME')
L1 = tf.nn.relu(L1)
L1 = tf.nn.max_pool(L1, ksize=[1,2,2,1], strides=[1,2,2,1], padding='SAME')
L1 = tf.nn.dropout(L1, keep_prob)
print(L1)
```

```
Tensor("dropout_3/mul_1:0", shape=(?, 16, 16, 32), dtype=float32)
```

In [109]:
```
W2 = tf.Variable(tf.random_normal([3,3,32,64], stddev=0.01))
L2 = tf.nn.conv2d(L1, W2, strides=[1,1,1,1], padding='SAME')
L2 = tf.nn.relu(L2)
L2 = tf.nn.max_pool(L2, ksize=[1,2,2,1], strides=[1,2,2,1], padding='SAME')
L2 = tf.nn.dropout(L2, keep_prob)

print(L2)
```

```
Tensor("dropout_4/mul_1:0", shape=(?, 8, 8, 64), dtype=float32)
```

In [110]:
```
W3 = tf.Variable(tf.random_normal([3,3,64, 128], stddev=0.01))
L3 = tf.nn.conv2d(L2, W3, strides=[1,1,1,1], padding='SAME')
L3 = tf.nn.relu(L3)
L3 = tf.nn.max_pool(L3, ksize=[1,2,2,1], strides=[1,2,2,1], padding='SAME')
L3 = tf.nn.dropout(L3, keep_prob)

print(L3)
```

```
Tensor("dropout_5/mul_1:0", shape=(?, 4, 4, 128), dtype=float32)
```

In [111]:
```
W4 = tf.Variable(tf.random_normal([4 * 4 * 128, 256], stddev=0.01))
L4 = tf.reshape(L3, [-1, 4 *4 * 128])
L4 = tf.matmul(L4, W4)
L4 = tf.nn.relu(L4)
print(L4)
```

```
Tensor("Relu_7:0", shape=(?, 256), dtype=float32)
```

In [112]:
```
W5 = tf.Variable(tf.random_normal([256,2], stddev=0.01))
model = tf.matmul(L4, W5)
model
```

Out[112]:
```
<tf.Tensor 'MatMul_3:0' shape=(?, 2) dtype=float32>
```

In [113]:
```
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits_v2(logits=model, labels=Y)
optimizer = tf.train.AdamOptimizer(0.001).minimize(cost)
```

In [114]:
```
init = tf.global_variables_initializer()
sess = tf.Session()
sess.run(init)
```

In [115]:
```
batch_size = 100
total_batch = int(X_train.shape[0]/ batch_size)
total_batch
epochs = 20
```

In [116]:
```
def next_batch(start, num, data, labels):
    data_X = data[start:start+num]
    data_y = labels[start:start+num]
    return np.asarray(data_X), np.asarray(data_y)
```

In [117]:

```
print(X_train.shape, Y_train.shape)
batch_X, batch_y = next_batch(0,10, X_train, Y_train)
print(batch_X.shape, batch_y.shape)
```

```
(285, 32, 32, 3) (285, 2)
(10, 32, 32, 3) (10, 2)
```

In [118]:

```
total_cost = 0
epoch = 0
print("전체 입력 데이터 : {}".format(X_train.shape))
print("전체 출력 데이터 : {}".format(Y_train.shape))

for epoch in range(20):
    total_cost = 0
    for i in range(total_batch):
        batch_xs, batch_ys = next_batch(batch_size*i, batch_size, X_train, Y_train)
        batch_xs = batch_xs.reshape(-1, 32, 32, 3)
        _, cost_val = sess.run([optimizer, cost], feed_dict={X: batch_xs, Y: batch_ys, k
        total_cost += cost_val
        if (i==0 or i==total_batch-1):
            print('data_step = {}, Avg. cost = {:.3f}'.format(i, cost_val))
    print('epoch: {} total.cost = {:.3f}'.format(epoch, total_cost))
```

```
전체 입력 데이터 : (285, 32, 32, 3)
전체 출력 데이터 : (285, 2)
data_step = 0, Avg. cost = 0.693
data_step = 1, Avg. cost = 0.693
epoch: 0 total.cost = 1.386
data_step = 0, Avg. cost = 0.693
data_step = 1, Avg. cost = 0.691
epoch: 1 total.cost = 1.384
data_step = 0, Avg. cost = 0.687
data_step = 1, Avg. cost = 0.676
epoch: 2 total.cost = 1.363
data_step = 0, Avg. cost = 0.658
data_step = 1, Avg. cost = 0.616
epoch: 3 total.cost = 1.273
data_step = 0, Avg. cost = 0.569
data_step = 1, Avg. cost = 0.474
epoch: 4 total.cost = 1.043
data_step = 0, Avg. cost = 0.425
data_step = 1, Avg. cost = 0.402
epoch: 5 total.cost = 0.826
data_step = 0, Avg. cost = 0.340
data_step = 1, Avg. cost = 0.227
epoch: 6 total.cost = 0.567
data_step = 0, Avg. cost = 0.217
data_step = 1, Avg. cost = 0.208
epoch: 7 total.cost = 0.425
data_step = 0, Avg. cost = 0.302
data_step = 1, Avg. cost = 0.199
epoch: 8 total.cost = 0.501
data_step = 0, Avg. cost = 0.156
data_step = 1, Avg. cost = 0.162
epoch: 9 total.cost = 0.318
data_step = 0, Avg. cost = 0.272
data_step = 1, Avg. cost = 0.174
epoch: 10 total.cost = 0.446
data_step = 0, Avg. cost = 0.134
data_step = 1, Avg. cost = 0.202
epoch: 11 total.cost = 0.336
data_step = 0, Avg. cost = 0.175
data_step = 1, Avg. cost = 0.161
epoch: 12 total.cost = 0.336
data_step = 0, Avg. cost = 0.160
data_step = 1, Avg. cost = 0.175
epoch: 13 total.cost = 0.334
data_step = 0, Avg. cost = 0.114
data_step = 1, Avg. cost = 0.177
epoch: 14 total.cost = 0.291
```

```
data_step = 0, Avg. cost = 0.125
data_step = 1, Avg. cost = 0.131
epoch: 15 total.cost = 0.256
data_step = 0, Avg. cost = 0.161
data_step = 1, Avg. cost = 0.116
epoch: 16 total.cost = 0.277
data_step = 0, Avg. cost = 0.123
data_step = 1, Avg. cost = 0.129
epoch: 17 total.cost = 0.252
data_step = 0, Avg. cost = 0.121
data_step = 1, Avg. cost = 0.134
epoch: 18 total.cost = 0.255
data_step = 0, Avg. cost = 0.125
data_step = 1, Avg. cost = 0.111
epoch: 19 total.cost = 0.236
```

In [123]:

```
print(X_test.shape, Y_test.shape)
```

```
(96, 32, 32, 3) (96, 2)
```

In [124]:

```
is_correct = tf.equal(tf.argmax(model,1), tf.argmax(Y,1))
accuracy = tf.reduce_mean(tf.cast(is_correct, tf.float32))
print('정확도 : ', sess.run(accuracy,
                        feed_dict={X:X_train.reshape(-1,32,32,3),
                                   Y:Y_train,
                                   keep_prob:0.8}))
```

```
정확도 :  0.95789474
```

In [141]:

```
image_w = 32
image_h = 32

print(groups_folder_path)
img = cv2.imread(groups_folder_path + 'test/green.jpg')
# img = cv2.imread('00.jpg')

img.shape
```

```
./cnn_sample/cnn_sample/
```

Out[141]:

```
(336, 448, 3)
```

In [135]:

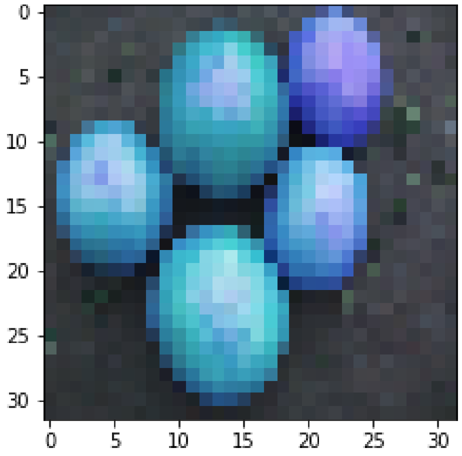
```
img = cv2.resize(img, None, fx=image_w/img.shape[1], fy=image_h/img.shape[0])
```

In [136]:

```
plt.imshow(img)
img.shape
```

Out[136]:

```
(32, 32, 3)
```



In [137]:

```
img_re = img.reshape(1, 32*32*3)
img_re
```

Out[137]:

```
array([[57, 58, 62, ..., 43, 47, 47]], dtype=uint8)
```

In [138]:

```
result = tf.argmax(model, 1)
res_idx = sess.run(result, feed_dict={X: img_re.reshape(-1, 32, 32, 3), keep_prob:1})
print('예측 레이블 :', label_names[res_idx[0]])
```

```
예측 레이블 : green
```