

In [1]:

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

In [2]:

```
import numpy as np
import tensorflow as tf
import vgg16
import utils
```

In [3]:

```
from skimage import io
import matplotlib.pyplot as plt
```

```
!pip install scikit-image
```

Image Load 및 data 준비하기

In [4]:

```
fn1 = "./test_data/tiger.jpeg"
#fn1 = "./test_data/puzzle.jpeg"
#fn1 = "./test_data/6201041_sd.jpg"
fn2 = "./test_data/bottle1.jpg"
#fn2 = "./test_data/IMG_0358s.jpg"
```

In [5]:

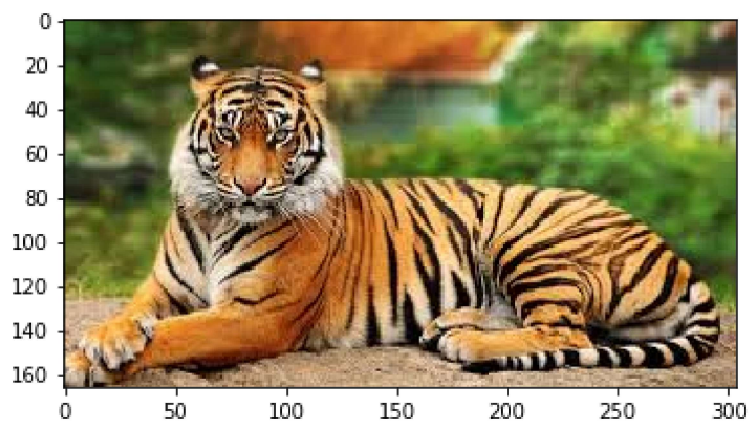
```
i1 = io.imread(fn1)
i2 = io.imread(fn2)
```

In [6]:

```
plt.imshow(i1)
```

Out[6]:

<matplotlib.image.AxesImage at 0x110698c9668>



In [7]:

```
plt.imshow(i2)
```

Out[7]:

<matplotlib.image.AxesImage at 0x1106994bef0>



```
f, axarr = plt.subplots(1,2, sharey=True, figsize=(11,5)) axarr[0].imshow(i1) axarr[1].imshow(i2)
```

In [8]:

```
img1 = utils.load_image(fn1)
img2 = utils.load_image(fn2)

# Just in case of using four-channel images
img1 = img1[:, :, :3]
img2 = img2[:, :, :3]

print(img1.shape)
print(img2.shape)
```

```
(224, 224, 3)
(224, 224, 3)
```

In [9]:

```
img1r = img1.reshape((1, 224, 224, 3))
img2r = img2.reshape((1, 224, 224, 3))

print(img1r.shape)
print(img2r.shape)
```

```
(1, 224, 224, 3)
(1, 224, 224, 3)
```

In [10]:

```
batch = np.concatenate((img1r, img2r), 0)
print(batch.shape)
```

```
(2, 224, 224, 3)
```

VGG model

In [11]:

```
# !wget https://www.dropbox.com/s/8a8rei66f72um4i/vgg16.npy
vgg = vgg16.Vgg16('vgg16.npy')
#print(vgg.data_dict)
```

npz file loaded

Tensorflow

In [12]:

```
images = tf.placeholder("float", [2, 224, 224, 3])
```

In [13]:

```
vgg.build(images)
```

```
build model started
build model finished: 18s
```

- Students : vgg16.py 를 열어서 build 함수 이해하기

In [14]:

```
# initialize
sess = tf.InteractiveSession()
```

In [15]:

```
feed_dict = {images: batch}
prob = sess.run(vgg.prob, feed_dict=feed_dict)
```

예측 결과 출력하기

In [16]:

```
print(prob[0])
```

```
[3.17088733e-09 5.07244025e-09 1.05872644e-09 2.29469843e-09
 5.10026778e-11 6.15994722e-09 1.24792243e-09 1.06688454e-07
 8.25466202e-08 5.73307126e-08 2.28812428e-06 6.02177597e-09
 5.84958970e-09 7.22982563e-09 4.58966154e-09 7.93626569e-08
 5.83563364e-10 5.54858914e-09 3.53818141e-10 1.28610966e-09
 2.47085108e-09 2.70781655e-08 1.90534588e-09 4.97443464e-09
 2.11313793e-08 9.73005587e-09 5.21899235e-09 1.21839139e-09
 1.65776615e-09 5.07614395e-10 5.99975269e-09 7.07930936e-10
 3.33412720e-09 1.76633037e-08 8.31221492e-09 2.57834714e-08
 1.42284051e-07 1.48224146e-07 4.07635845e-08 4.03743940e-07
 2.14506324e-09 3.30699834e-07 2.08427986e-09 3.46129632e-08
 7.68477104e-09 3.44609447e-07 1.87470555e-08 5.59177638e-09
 4.10665315e-08 1.42383968e-07 2.81769825e-08 7.21696125e-08
 2.64849120e-09 2.73193801e-09 1.03320392e-08 1.43537759e-09
 5.62095046e-08 3.80479648e-09 4.39980585e-09 2.34869013e-09
 5.17448058e-08 6.60588952e-08 2.84789223e-07 4.82069762e-09
 2.20147189e-09 3.13233772e-09 4.79283813e-08 1.35849660e-08
 4.07164187e-08 8.80545414e-10 1.69365215e-08 1.57482216e-09
 4.32372076e-08 1.41449483e-08 2.65772790e-08 1.57803937e-09]
```

In [17]:

```
print(prob[0].shape)
```

```
(1000,)
```

In [18]:

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

292

In [19]:

```
# returns the top1 string
def print_prob(prob, file_path):
    synset = [l.strip() for l in open(file_path).readlines()]

    # print prob
    pred = np.argsort(prob)[::-1]

    # Get top1 label
    top1 = synset[pred[0]]
    print(("Top1: ", top1, prob[pred[0]]))
    # Get top5 label
    top5 = [(synset[pred[i]], prob[pred[i]]) for i in range(5)]
    print(("Top5: ", top5))
    return top1
```

In [20]:

```
#!/cat synset.txt # Linux or Mac users
!type synset.txt # Windows users
```

```
n01440764 tench, Tinca tinca
n01443537 goldfish, Carassius auratus
n01484850 great white shark, white shark, man-eater, man-eating shark, Carcharodon
n carcharias
n01491361 tiger shark, Galeocerdo cuvieri
n01494475 hammerhead, hammerhead shark
n01496331 electric ray, crampfish, numbfish, torpedo
n01498041 stingray
n01514668 cock
n01514859 hen
n01518878 ostrich, Struthio camelus
n01530575 brambling, Fringilla montifringilla
n01531178 goldfinch, Carduelis carduelis
n01532829 house finch, linnet, Carpodacus mexicanus
n01534433 junco, snowbird
n01537544 indigo bunting, indigo finch, indigo bird, Passerina cyanea
n01558993 robin, American robin, Turdus migratorius
n01560419 bulbul
n01580077 jay
```

In [21]:

```
top1 = print_prob(prob[0], 'synset.txt')
```

```
('Top1: ', 'n02129604 tiger, Panthera tigris', 0.8209941)
('Top5: ', [('n02129604 tiger, Panthera tigris', 0.8209941), ('n02123159 tiger cat',
0.17724687), ('n02128925 jaguar, panther, Panthera onca, Felis onca', 0.0012660765),
('n02127052 lynx, catamount', 0.00020411507), ('n02128385 leopard, Panthera pardus',
0.00016284417)])
```

In [22]:



```
print("Top1: {}".format(top1))
```

Top1: n02129604 tiger, Panthera tigris

In [23]:



```
print_prob(prob[1], 'synset.txt')
```

```
('Top1: ', 'n03887697 paper towel', 0.1499218)
('Top5: ', [('n03887697 paper towel', 0.1499218), ('n04254120 soap dispenser', 0.148
15806), ('n15075141 toilet tissue, toilet paper, bathroom tissue', 0.09420873), ('n0
3666591 lighter, light, igniter, ignitor', 0.07085726), ('n02747177 ashcan, trash ca
n, garbage can, wastebin, ash bin, ash-bin, ashbin, dustbin, trash barrel, trash bi
n', 0.029463036)])
```

Out[23]:

```
'n03887697 paper towel'
```