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
```

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/water1.jpg"
#fn2 = "./test_data/IMG_0358s.jpg"
```

In [5]:

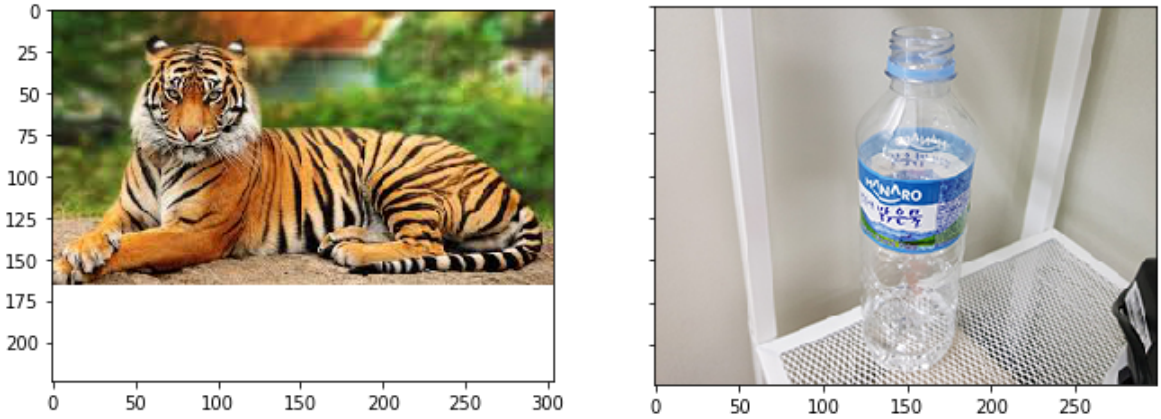
```
i1 = io.imread(fn1)
i2 = io.imread(fn2)
```

In [6]:

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

Out[6]:

```
<matplotlib.image.AxesImage at 0x7f927f48fc50>
```



In [7]:

```
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 [8]:

```
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 [9]:

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

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

VGG model

In [10]:

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

```
npy file loaded
```

Tensorflow

In [11]:

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

In [12]:

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

```
build model started
build model finished: 0s
```

- Students : vgg16.py 를 열어서 build 함수 이해하기

In [13]:

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

In [14]:

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

예측 결과 출력하기

In [15]:

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

```
 1.34655387e-09 3.93415245e-10 4.99996355e-09 5.53041113e-10
 2.73681366e-09 1.62876557e-08 7.26736005e-09 2.38210287e-08
 1.34322690e-07 1.42641454e-07 3.35240351e-08 3.60810787e-07
 1.64736069e-09 2.99022787e-07 1.64784453e-09 2.74404357e-08
 6.17680485e-09 2.86641438e-07 1.59507945e-08 4.63069760e-09
 3.55706256e-08 1.25475879e-07 2.36197444e-08 6.43075850e-08
 1.89985094e-09 2.11092810e-09 7.76221043e-09 1.08953491e-09
 4.23251194e-08 3.02716430e-09 3.41064110e-09 1.74157311e-09
 4.16769552e-08 4.97485289e-08 2.42212877e-07 3.45940565e-09
 1.68887304e-09 2.52265586e-09 3.70298814e-08 9.99667105e-09
 3.05509502e-08 6.52017107e-10 1.39747893e-08 1.21629185e-09
 3.55578713e-08 1.15770922e-08 2.29155397e-08 1.17242371e-09
 2.61728630e-08 1.27341462e-08 1.74181236e-09 1.39801486e-08
 2.87972757e-09 1.89755833e-09 1.44619918e-07 4.40924097e-08
 2.22063932e-08 9.78265913e-09 1.77386283e-07 1.71648945e-10
 8.18653163e-08 2.15145857e-09 6.49100240e-09 3.77893183e-09
 4.33837144e-09 3.51123859e-08 3.40012379e-10 3.28161520e-09
 1.03618691e-09 2.86512822e-08 2.02806305e-09 7.07099801e-09
 2.64505085e-10 1.02093012e-09 6.29163122e-09 2.38308329e-09
 2.63221498e-08 2.13384173e-08 8.39223002e-09 2.93032576e-09
```

In [16]:

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

```
(1000,)
```

In [17]:

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

```
292
```

In [18]:

```
# 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 [19]:

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

```
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 cyane
a
n01558993 robin, American robin, Turdus migratorius
n01560419 bulbul
n01580077 jay
n01582220 magpie
n01592084 chickadee
n01601694 water ouzel, dipper
n01608432 kite
n01614925 bald eagle, American eagle, Haliaeetus leucocephalus
n01616318 vulture
n01622779 great grey owl, great gray owl, Strix nebulosa
```

In [20]:

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

```
('Top1: ', 'n02129604 tiger, Panthera tigris', 0.82691544)
('Top5: ', [('n02129604 tiger, Panthera tigris', 0.82691544), ('n02123
159 tiger cat', 0.17136905), ('n02128925 jaguar, panther, Panthera onc
a, Felis onca', 0.0012765428), ('n02127052 lynx, catamount', 0.0001735
1768), ('n02128385 leopard, Panthera pardus', 0.00015544178)])
```

In [21]:

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

```
Top1: n02129604 tiger, Panthera tigris
```

In [22]:

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

```
('Top1: ', 'n04557648 water bottle', 0.9483804)
('Top5: ', [('n04557648 water bottle', 0.9483804), ('n03983396 pop bot
tle, soda bottle', 0.017549021), ('n04560804 water jug', 0.012552273),
('n03825788 nipple', 0.011409558), ('n02815834 beaker', 0.00464157)])
```

Out[22]:

```
'n04557648 water bottle'
```