과제 6

In [1]:
```
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.resnet50 import preprocess_input, decode_predictions
import numpy as np

model1 = ResNet50(weights='imagenet') # ImageNet으로 학습된 신경망을 호출
img_path = '안경.jpg'            # 샘플 영상을 입력. 크기는 관계없음
img = image.load_img(img_path, target_size=(224, 224)) # 영상을 resize 함
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

preds = model1.predict(x)
# decode the results into a list of tuples (class, description, probability)
# (one such list for each sample in the batch) print('Predicted:', decode_predictions(preds, top=3)[0])
decode_predictions(preds, top=3)[0]
```

Out[1]:
```
[('n03476684', 'hair_slide', 0.19860883),
 ('n04328186', 'stopwatch', 0.0922622),
 ('n04579432', 'whistle', 0.06265656)]
```

In [2]:
```
model2 = ResNet50(weights='imagenet') # ImageNet으로 학습된 신경망을 호출
img_path = '친칠라.jpg'          # 샘플 영상을 입력. 크기는 관계없음
img = image.load_img(img_path, target_size=(224, 224)) # 영상을 resize 함
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

preds = model2.predict(x)
# decode the results into a list of tuples (class, description, probability)
# (one such list for each sample in the batch) print('Predicted:', decode_predictions(preds, top=3)[0])
decode_predictions(preds, top=3)[0]
```

Out[2]:
```
[('n02325366', 'wood_rabbit', 0.7262946),
 ('n02328150', 'Angora', 0.1731174),
 ('n02326432', 'hare', 0.08775046)]
```

In [3]:
```
model3 = ResNet50(weights='imagenet') # ImageNet으로 학습된 신경망을 호출
img_path = '자전거.jpg'          # 샘플 영상을 입력. 크기는 관계없음
img = image.load_img(img_path, target_size=(224, 224)) # 영상을 resize 함
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

preds = model3.predict(x)
# decode the results into a list of tuples (class, description, probability)
# (one such list for each sample in the batch) print('Predicted:', decode_predictions(preds, top=3)[0])
decode_predictions(preds, top=3)[0]
```

Out[3]:
```
[('n02835271', 'bicycle-built-for-two', 0.6251696),
 ('n03792782', 'mountain_bike', 0.2828379),
 ('n04482393', 'tricycle', 0.005772204)]
```

In [4]:
```
model4 = ResNet50(weights='imagenet') # ImageNet으로 학습된 신경망을 호출
img_path = '자동차.jpg'          # 샘플 영상을 입력. 크기는 관계없음
img = image.load_img(img_path, target_size=(224, 224)) # 영상을 resize 함
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

preds = model4.predict(x)
# decode the results into a list of tuples (class, description, probability)
# (one such list for each sample in the batch) print('Predicted:', decode_predictions(preds, top=3)[0])
decode_predictions(preds, top=3)[0]
```

Out[4]:
```
[('n04285008', 'sports_car', 0.7899707),
 ('n04037443', 'racer', 0.06857479),
 ('n04336792', 'stretcher', 0.051590234)]
```

In [5]:
```
model5 = ResNet50(weights='imagenet') # ImageNet으로 학습된 신경망을 호출
img_path = '아령.jpg'            # 샘플 영상을 입력. 크기는 관계없음
img = image.load_img(img_path, target_size=(224, 224)) # 영상을 resize 함
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

preds = model5.predict(x)
# decode the results into a list of tuples (class, description, probability)
# (one such list for each sample in the batch) print('Predicted:', decode_predictions(preds, top=3)[0])
decode_predictions(preds, top=3)[0]
```

```
WARNING:tensorflow:5 out of the last 5 calls to <function Model.make_predict_function.<locals>.predict_function at 0x0000014014674F70> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has experimental_relax_shapes=True option that relaxes argument shapes that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for  more details.
```

Out[5]:
```
[('n03255030', 'dumbbell', 0.85074365),
 ('n02790996', 'barbell', 0.14925599),
 ('n03481172', 'hammer', 3.4103147e-07)]
```