이미지 인식 샘플 코드

학습 내용

- resnet50의 모델을 활용하여 이미지 분류를 수행해 봅니다.

In [5]:
```
# 필요한 라이브러리를 가져옵니다.
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.preprocessing import image
import numpy as np
from tensorflow.keras.applications.resnet50 import preprocess_input, decode_predict
from keras.applications.imagenet_utils import preprocess_input
```

In [2]:
```
# 이미지를 로드하고 전처리하는 함수
def preprocess_image(image_path):
    img = image.load_img(image_path, target_size=(224, 224))
    img = image.img_to_array(img)
    img = np.expand_dims(img, axis=0)
    img = preprocess_input(img)
    return img
```

In [3]:
```
# 이미지 객체 인식을 수행하는 함수
def predict_objects(image_path):
    # 사전에 훈련된 ResNet50 모델을 불러옵니다.
    model = keras.applications.resnet50.ResNet50(weights='imagenet')

    # 이미지를 전처리합니다.
    img = preprocess_image(image_path)

    # 이미지를 모델에 입력하여 예측을 수행합니다.
    preds = model.predict(img)
    # 예측 결과를 해석합니다.
    decoded_preds = decode_predictions(preds, top=3)[0]

    # 예측 결과를 출력합니다.
    for _, label, confidence in decoded_preds:
        print(f"{label}: {confidence * 100}%")
```

In [9]:
```
from IPython.display import Image
```

In [11]:
```
# 이미지 파일 경로를 지정합니다.
image_path = 'Dog_rawPixel01.jpg'

# 이미지 표시
Image(image_path)
```

Out[11]:

In [12]:
```
# 이미지 객체 인식을 수행합니다.
predict_objects(image_path)
```

```
1/1 [==============================] - 2s 2s/step
standard_poodle: 93.59152317047119%
miniature_poodle: 4.353092610836029%
golden_retriever: 0.6918197963386774%
```

In [13]:
```
# 이미지 파일 경로를 지정합니다.
image_path = 'life_unsplash.jpeg'

# 이미지 표시
Image(image_path)
```

Out[13]:

In [14]:
```
# 이미지 객체 인식을 수행합니다.
predict_objects(image_path)
```

```
WARNING:tensorflow:5 out of the last 5 calls to <function Model.make_predict_functio
n.<locals>.predict_function at 0x000002AA0377C550> triggered tf.function retracing.
Tracing is expensive and the excessive number of tracings could be due to (1) creati
ng @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 reduce_retracing=True option that can
avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/gui
de/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/
function for  more details.
1/1 [==============================] - 2s 2s/step
mosquito_net: 26.91299319267273%
quilt: 19.8626309633255%
crib: 19.388511776924133%
```

In [15]:
```
# 이미지 파일 경로를 지정합니다.
image_path = 'Deer_un.jpeg'

# 이미지 표시
Image(image_path)
```

Out[15]:

In [16]:

```
# 이미지 객체 인식을 수행합니다.
predict_objects(image_path)
```

```
WARNING:tensorflow:6 out of the last 6 calls to <function Model.make_predict_functio
n.<locals>.predict_function at 0x000002AA005BE550> triggered tf.function retracing.
Tracing is expensive and the excessive number of tracings could be due to (1) creati
ng @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 reduce_retracing=True option that can
avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/gui
de/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/
function for  more details.
1/1 [==============================] - 2s 2s/step
ibex: 55.18874526023865%
impala: 14.141763746738434%
gazelle: 11.144579201936722%
```

이벡스(Ibex): 이벡스는 산악 지형에 서식하는 동물로, 주로 유럽, 아시아, 아프리카 지역에서 발견됩니다. 큰 뿔과 특이한 외모를 가지고 있으며, 주로 암벽이나 절벽에서 생활합니다.

임팔라(Impala): 임팔라는 아프리카 대륙에서 서식하는 큰 영양 동물로, 주로 사바나와 숲 지역에서 발견됩니다. 얇은 몸과 날씬한 다리를 가지고 있으며, 높은 점프력과 빠른 속도로 알려져 있습니다.

가젤(Gazelle): 가젤은 아프리카와 아시아 지역에서 발견되는 작은 동물로, 주로 사막과 초원에서 서식합니다. 가늘고 우아한 몸과 긴 다리를 가지고 있으며, 빠른 속도와 재빠른 움직임으로 유명합니다.