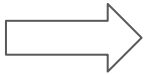
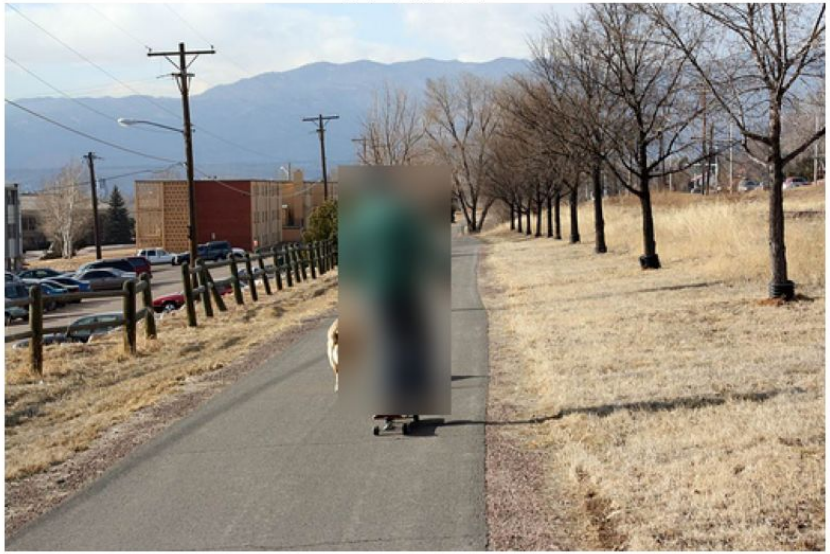
6강

Computer Vision 입문

입력 이미지

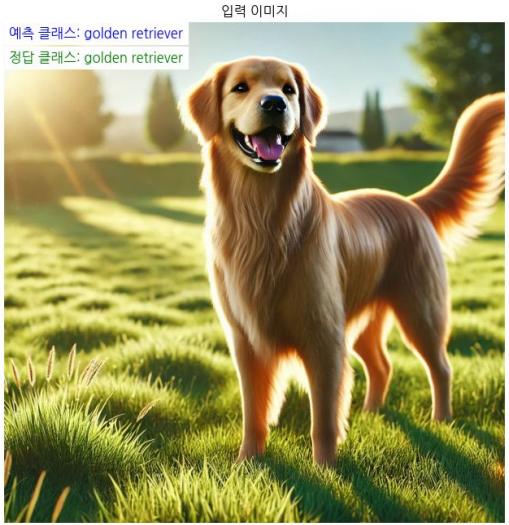


블러 처리된 이미지



개요

Image Classification



이미지 분류 (Image Classification):

- 이미지 내 객체를 식별하고, 해당 객체의 종류를 예측하는 기술 이해 및 실습

Object Detection

객체 검출 (Object Detection):

- 이미지 내 다양한 객체를 찾아 위치를 표시하고, 객체의 종류를 식별

Instance Segmentation

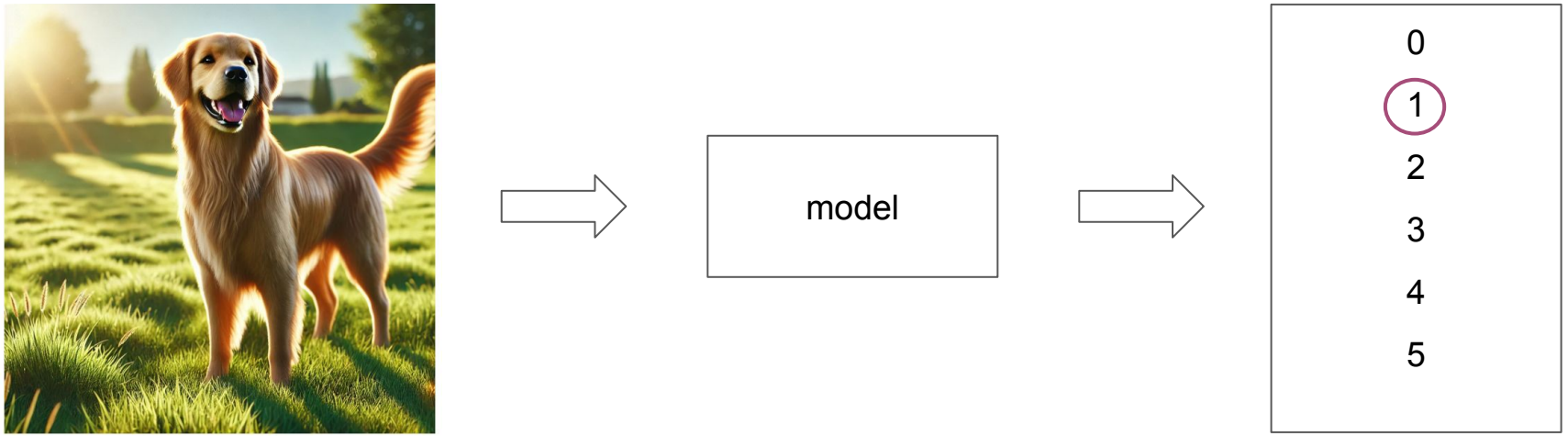
인스턴스 세그멘테이션 (Instance Segmentation):

- 객체의 경계를 더욱 정밀하게 나누어 각 객체를 마스크 처리

Image Classification

정의:

- 이미지를 입력으로 받아, 해당 이미지가 특정 클래스 (예: 개, 고양이, 자동차 등)에 속하는지 예측하는 기술.



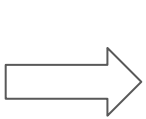
Input **Output**

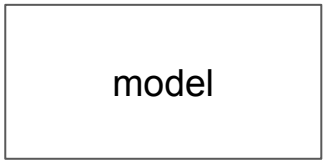
Image Classification

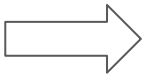
정의:

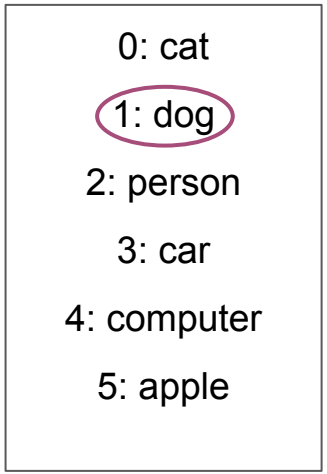
- 이미지를 입력으로 받아, 해당 이미지가 특정 클래스 (예: 개, 고양이, 자동차 등)에 속하는지 예측하는 기술.

Input









Output

Image Classification

실습

1. Image Load

```
# 필요한 라이브러리 임포트
from PIL import Image
import requests
import matplotlib.pyplot as plt
import matplotlib.patches as patches
from google.colab import files
import io

# 이미지 업로드 버튼 만들기
uploaded = files.upload()
# 첫 번째로 업로드된 파일 읽기
for filename in uploaded.keys():
    # 이미지를 열기
    image = Image.open(io.BytesIO(uploaded[filename]))
    break


plt.figure(figsize=(8, 8))
plt.imshow(image)
plt.axis('off')
plt.title('입력 이미지')
plt.show()
```

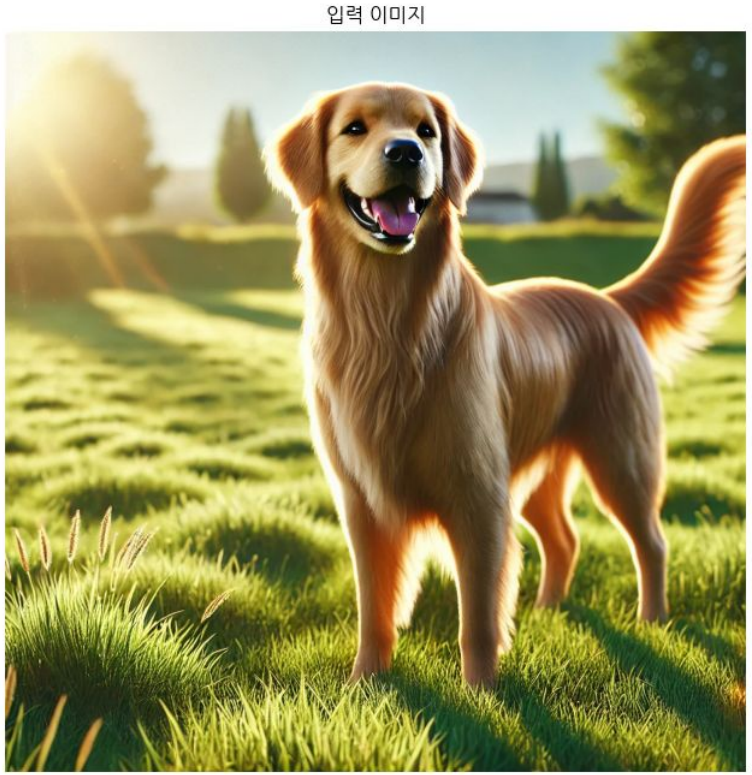


Image Classification

실습

2. Model Load

```
# ResNet 모델 로드 (사전 학습된 모델)
model = models.resnet18(pretrained=True)  # ResNet-18 모델 사용
model.eval()
```

```
ResNet(
  (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False)
  (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (relu): ReLU(inplace=True)
  (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mode=False)
  (layer1): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    )
```

3. 입력 데이터 전처리

```
# 입력 데이터 전처리
image_resized = image.resize((224, 224))  # ResNet 입력 크기와 동일하게 조정
image_array = np.array(image_resized)    # 이미지를 numpy 배열로 변환

# 배열 형태를 모델 입력 형식 (Batch, Channels, Height, Width)으로 변경
input_tensor = torch.tensor(image_array).permute(2, 0, 1).unsqueeze(0).float() / 255.
input_tensor.shape
```

4. 모델이 전처리된 입력 데이터에 대하여 추론 수행

```
# 추론 수행
with torch.no_grad():
    outputs = model(input_tensor)  # 모델에 입력
    predicted_class_idx = outputs.argmax(-1).item()  # 확률 계산
print(predicted_class_idx)
```

Image Classification

실습

5. ImageNet 학습 데이터의 클래스(=레이블) 이름 로드

```
# ImageNet 클래스 이름 로드
LABELS_URL = "https://raw.githubusercontent.com/anishathalye/imagenet-simple-labels/master/imagenet-simple-labels.json"
import requests
labels = requests.get(LABELS_URL).json()
```

labels

0 - tench
1 - goldfish
2 - great white shark
3 - tiger shark
4 - hammerhead shark
5 - electric ray
6 - stingray
…
999 - toilet paper

6. 예측된 클래스의 인덱스와 이름 매칭

```
predicted_class = labels[predicted_class_idx]
print(f"예측 클래스: {predicted_class}")
```

예측 클래스: Golden Retriever

Image Classification

이미지 분류 응용 분야 예시

일상 생활:

- 사진 검색(이미지에서 사람, 장소, 사물 검색)
- 소셜 미디어 필터링(부적절한 콘텐츠 탐지)

의료:

- 의료 영상에서 질병 진단(예: X-ray에서 폐렴 진단)

전자 상거래:

- 제품 추천 시스템

산업:

- 품질 검사(공장에서 결함 있는 제품 탐지).

Object Detection

정의:

- Object Detection은 이미지 또는 영상에서 객체를 식별하고, 해당 객체의 위치를 나타내는 경계 상자(Bounding Box)를 생성하는 기술입니다.
- 단순히 객체의 종류만 분류하는 Classification과 달리, 객체의 **위치**까지 제공.

주요 작업:

- **Detection:** 이미지에서 객체의 존재 여부 확인.
- **Localization:** 객체의 위치(좌표)를 경계 상자로 나타냄.
- **Classification:** 각 객체의 종류를 분류.

입력 이미지

객체 검출 결과

Object Detection

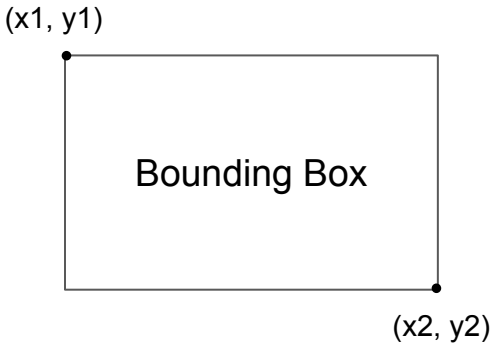
객체 검출(Object Detection)과 분류(Classification)의 차이점

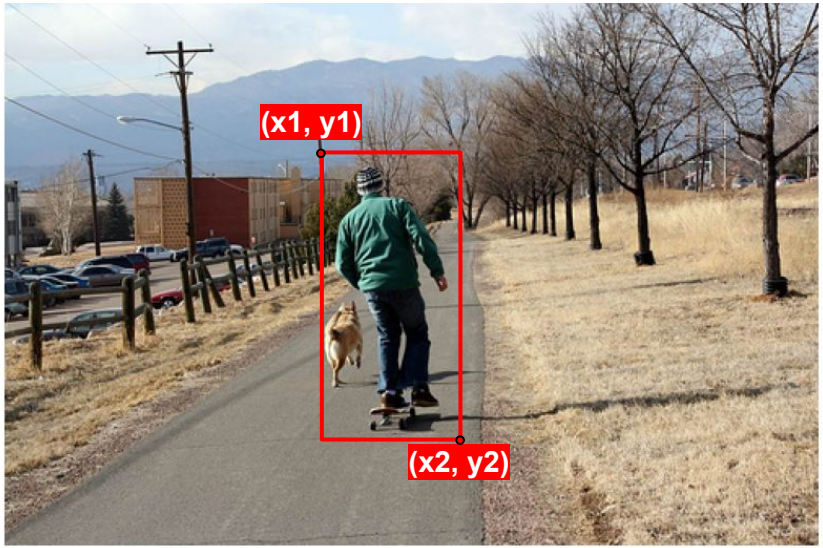
기술	주요 기능	예시
Classification	이미지 전체를 분석하여 하나의 클래스 분류	이미지: 고양이, 결과: "Cat"
Object Detection	이미지 내 객체 각각을 찾아 위치와 종류를 분류	이미지: 강아지와 고양이 포함, 결과: "Dog(위치)", "Cat(위치)"

Object Detection

Bounding Box의 개념

- **정의:**
  - Bounding Box는 객체의 위치를 나타내는 직사각형 경계 상자입니다.
  - 네 개의 좌표 값(x1, y1, x2, y2)으로 정의됩니다.
- **특징:**
  - 객체의 경계 부분을 포괄적으로 포함.
  - 중심 위치, 크기, 너비와 높이를 활용해 표현 가능.





Object Detection

실습

1. Image Load

```
# 이미지 URL
image_url = 'http://images.cocodataset.org/val2017/000000324158.jpg'
# Images from the COCO dataset, licensed under CC BY 4.0.

# 이미지 다운로드
response = requests.get(image_url)
image = Image.open(BytesIO(response.content)).convert("RGB")

# 이미지를 OpenCV 형식으로 변환 (RGB to BGR)
image_cv = cv2.cvtColor(np.array(image), cv2.COLOR_RGB2BGR)

# 이미지 시각화
plt.figure(figsize=(10, 10))
plt.imshow(image)
plt.axis('off')
plt.title('입력 이미지')
plt.show()
```

입력 이미지

Object Detection

실습

2. Model Load 및 객체 검출 수행

```
from ultralytics import YOLO

# YOLO 모델 로드
model = YOLO("yolov8n.pt")
# 객체 검출 수행
results = model(image_cv)

# 검출 결과 확인
results  # 콘솔에 결과 출력
```

```
0: 448x640 1 person, 5 cars, 1 dog, 1 skateboard, 9.4ms
Speed: 2.0ms preprocess, 9.4ms inference, 1.3ms postprocess per image at shape (1, 3, 448, 640)
[ultralytics.engine.results.Results object with attributes:

 boxes: ultralytics.engine.results.Boxes object
 keypoints: None
 masks: None
 names: {0: 'person', 1: 'bicycle', 2: 'car', 3: 'motorcycle', 4: 'airplane', 5: 'bus', 6: 'train', 7: 'truck', 8: 'boat', 9: 'traffic light', 10: 'fire hydrant', 11: 'stop sign', 12: 'parking
meter', 13: 'bench', 14: 'bird', 15: 'cat', 16: 'dog', 17: 'horse', 18: 'sheep', 19: 'cow', 20: 'elephant', 21: 'bear', 22: 'zebra', 23: 'giraffe', 24: 'backpack', 25: 'umbrella', 26: 'handbag',
27: 'tie', 28: 'suitcase', 29: 'frisbee', 30: 'skis', 31: 'snowboard', 32: 'sports ball', 33: 'kite', 34: 'baseball bat', 35: 'baseball glove', 36: 'skateboard', 37: 'surfboard', 38: 'tennis
racket', 39: 'bottle', 40: 'wine glass', 41: 'cup', 42: 'fork', 43: 'knife', 44: 'spoon', 45: 'bowl', 46: 'banana', 47: 'apple', 48: 'sandwich', 49: 'orange', 50: 'broccoli', 51: 'carrot', 52:
'hot dog', 53: 'pizza', 54: 'donut', 55: 'cake', 56: 'chair', 57: 'couch', 58: 'potted plant', 59: 'bed', 60: 'dining table', 61: 'toilet', 62: 'tv', 63: 'laptop', 64: 'mouse', 65: 'remote', 66:
'keyboard', 67: 'cell phone', 68: 'microwave', 69: 'oven', 70: 'toaster', 71: 'sink', 72: 'refrigerator', 73: 'book', 74: 'clock', 75: 'vase', 76: 'scissors', 77: 'teddy bear', 78: 'hair drier',
79: 'toothbrush'}
 obb: None
```

Object Detection

실습

3. Image Load

```
# 검출된 결과를 이미지에 그리기
annotated_image = results[0].plot()

# OpenCV 형식에서 RGB로 변환
annotated_image_rgb = cv2.cvtColor(annotated_image, cv2.COLOR_BGR2RGB)

# 시각화
plt.figure(figsize=(10, 10))
plt.imshow(annotated_image_rgb)
plt.axis('off')
plt.title('객체 검출 결과')
plt.show()
```

객체 검출 결과

Object Detection

실습

4. 검출된 객체 크롭 및 블러 처리

- 검출된 객체 영역을 크롭하고 선택적으로 블러 처리.
- 특정 클래스(예: "person")만 블러 처리하여 개인정보를 보호하는 데 사용.

```python
def blur_objects(image, boxes, classes, class_names, target_classes, blur_ratio=51):
    """
    이미지에서 지정된 클래스의 바운딩 박스 영역을 블러 처리합니다.

    :param image: 원본 이미지 (OpenCV 형식)
    :param boxes: 바운딩 박스 좌표 리스트 (x1, y1, x2, y2)
    :param classes: 객체 클래스 인덱스 리스트
    :param class_names: 모델의 클래스 이름 리스트
    :param target_classes: 블러 처리할 클래스 이름 리스트
    :param blur_ratio: 블러 강도 (양수, 클수록 더 흐림)
    :return: 블러 처리된 이미지
    """
    for box, cls_idx in zip(boxes, classes):
        # 현재 객체의 클래스 이름
        class_name = class_names[cls_idx]
        # 대상 클래스인지 확인
        if class_name in target_classes:
            x1, y1, x2, y2 = map(int, box)
            # 객체 영역 추출
            obj = image[y1:y2, x1:x2]
            # 블러 처리
            blur_obj = cv2.GaussianBlur(obj, (blur_ratio, blur_ratio), 0)
            # 블러 처리된 객체를 원본 이미지에 다시 삽입
            image[y1:y2, x1:x2] = blur_obj
    return image
```

```python
# 블러 처리 수행
target_classes = ['person']  # 블러 처리할 클래스 이름
blurred_image = blur_objects(image_cv.copy(), boxes, classes, class_names, target_classes, blur_ratio=51)

# OpenCV 형식에서 RGB로 변환
blurred_image_rgb = cv2.cvtColor(blurred_image, cv2.COLOR_BGR2RGB)

# 시각화
plt.figure(figsize=(10, 10))
plt.imshow(blurred_image_rgb)
plt.axis('off')
plt.title('블러 처리된 이미지')
plt.show()
```

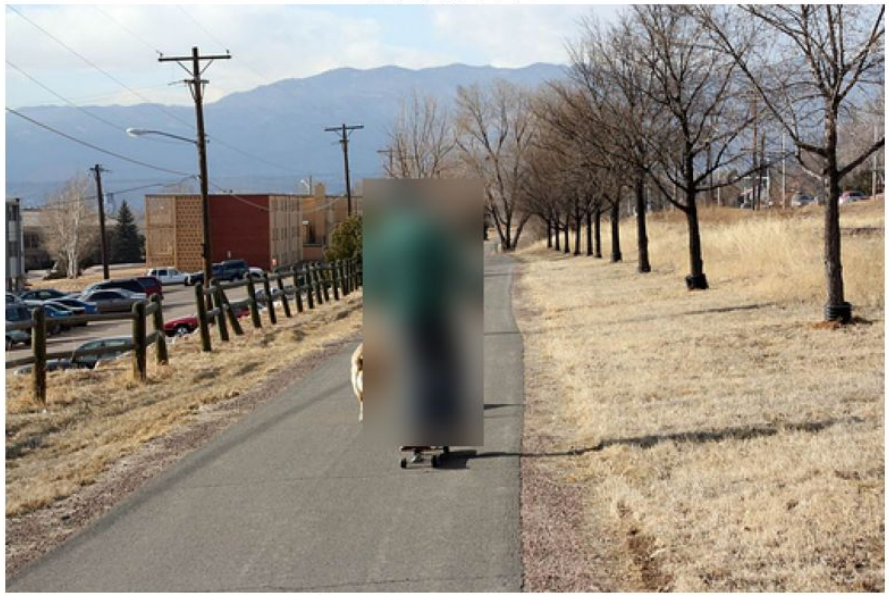
Object Detection

실습

4. 검출된 객체 크롭 및 블러 처리

- 검출된 객체 영역을 크롭하고 선택적으로 블러 처리.
- 특정 클래스(예: "person")만 블러 처리하여 개인정보를 보호하는 데 사용.

블러 처리된 이미지



Object Detection

1. 자율 주행 차량

- 도로 위 보행자, 차량, 신호등 등을 탐지해 안전한 주행 지원.

2. 스마트 시티

- CCTV 영상을 분석해 교통 흐름 개선 및 시민 안전 강화.

3. 의료 분야

- 의료 영상(CX, CT)에서 병변이나 종양을 탐지해 진단 지원.

4. 전자상거래

- 상품 이미지 분석으로 자동 태깅 및 검색 결과 개선.

5. 스포츠 분석

- 경기에서 선수와 공의 움직임을 추적해 전략 수립 지원.

실습