7주차

Object Detection

객체 검출(Object Detection)이란?

- 이미지나 영상에서 **객체의 위치와 클래스**를 동시에 예측하는 작업
- 일반적인 출력: 바운딩 박스 + 클래스 레이블 + 신뢰도(Confidence)

구성 요소

1. **Localization (위치 추정)**
   - 객체가 이미지에서 어디에 위치하는지를 예측
   - 보통 바운딩 박스 형태로 (x, y, w, h) 좌표 출력
2. **Classification (분류)**
   - 바운딩 박스 안의 객체가 어떤 클래스인지 예측
   - 예: 고양이, 자동차, 사람 등
3. **Confidence Score**
   - 예측한 바운딩 박스가 실제 객체일 확률 (0~1)

주요 방식

방식	설명
Two-stage	Region Proposal → Classification (e.g. R-CNN)
One-stage	한 번에 위치 + 클래스 동시 예측 (e.g. YOLO, SSD)

주요 모델 비교

모델	특징
R-CNN 계열	높은 정확도, 속도 느림
YOLO 계열	매우 빠름, 실시간 처리 가능
SSD	속도와 정확도의 균형 우수
Faster R-CNN	성능 우수, 실제 서비스에 적합

학습 데이터 구성

- 각 이미지에 대해 객체 위치(box)와 클래스(label)를 포함한 어노테이션 필요
- 대표적인 데이터셋: COCO, PASCAL VOC, Open Images

평가 지표

- **Precision / Recall**
- **mAP (mean Average Precision)**
  - IoU(Intersection over Union) 기준으로 정확도 평가

OpenCV: Face Detection

Goal

- OpenCV를 사용하여 얼굴을 검출하는 방법 학습
- Haar Cascade Classifier와 DNN 기반 검출 이해

Face Detection 방법

Haar Cascade Classifier

- 머신러닝 기반의 고전적인 얼굴 검출 방법
- OpenCV에서 미리 학습된 XML 파일 제공 (`haarcascade_frontalface_default.xml`)

방법	장점	단점
Haar Cascade	빠르고 가벼움	다양한 얼굴 방향에 약함

Haar Cascade를 이용한 얼굴 검출

```python
import cv2 as cv

# Haar Cascade 로드
face_cascade = cv.CascadeClassifier(cv.data.haarcascades + 'haarcascade_frontalface_default.xml')

# 이미지 로드
img = cv.imread('face.jpg')
img = cv.resize(img, (600, 400))

gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)

# 얼굴 검출
faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5)

# 얼굴에 사각형 그리기
for (x, y, w, h) in faces:
    cv.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 2)

# 결과 출력
cv.imshow('Face Detection', img)
cv.waitKey(0)
cv.destroyAllWindows()
```

Advanced Sample

```python
import cv2 as cv

# Haar Cascade 로드
face_cascade = cv.CascadeClassifier(cv.data.haarcascades + 'haarcascade_frontalface_default.xml')

# MP4 파일 읽기
cap = cv.VideoCapture('face.mp4')

while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break

    frame = cv.resize(frame, (600, 400))

    # 이미지를 그레이스케일로 변환
    gray = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)

    # 얼굴 검출
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5)

    # 반투명 파란색 사각형 그리기
    for (x, y, w, h) in faces:
        overlay = frame.copy()
        cv.rectangle(overlay, (x, y), (x+w, y+h), (245, 165, 66), -1)

        alpha = 0.5
        frame = cv.addWeighted(overlay, alpha, frame, 1 - alpha, 0)

    # 결과 프레임을 화면에 표시
    cv.imshow('Face Detection on Video', frame)

    if cv.waitKey(33) & 0xFF == 27:
        break

cap.release()
cv.destroyAllWindows()
```

OpenCV: Object Detection

Goal

- 객체 검출(Object Detection)의 개념 이해
- OpenCV의 DNN 모듈을 활용한 사물 인식
- MobileNet + SSD 모델을 사용한 실습

Object Detection 개요

- 객체 검출은 **영상 내의 객체 위치와 종류를 식별**하는 기술
- 대표적인 딥러닝 기반 방법:
  - SSD (Single Shot Multibox Detector)
  - YOLO (You Only Look Once)
  - Faster R-CNN 등

모델	특징
SSD	빠르며 모바일 환경에 적합
YOLO	매우 빠르며 정확도도 우수
R-CNN	높은 정확도, 속도는 느린 편

OpenCV DNN을 활용한 SSD 객체 검출

- OpenCV는 사전 학습된 MobileNet SSD 모델을 사용 가능
- Caffe 모델 구조와 가중치 파일 필요:
  - `MobileNetSSD_deploy.prototxt`
  - `MobileNetSSD_deploy.caffemodel`

MobileNet-SSD 모델 클래스 리스트

인덱스(Index)	클래스(Class Name)	인덱스(Index)	클래스(Class Name)
0	background	11	diningtable
1	aeroplane	12	dog
2	bicycle	13	horse
3	bird	14	motorbike
4	boat	15	person
5	bottle	16	pottedplant
6	bus	17	sheep
7	car	18	sofa
8	cat	19	train
9	chair	20	tvmonitor
10	cow	-	-

```python
import cv2 as cv

# 클래스 이름
class_names = ["background", "aeroplane", "bicycle", "bird", "boat",
               "bottle", "bus", "car", "cat", "chair", "cow", "diningtable",
               "dog", "horse", "motorbike", "person", "pottedplant",
               "sheep", "sofa", "train", "tvmonitor"]

# DNN 모델 로드
net = cv.dnn.readNetFromCaffe('MobileNetSSD_deploy.prototxt',
                              'MobileNetSSD_deploy.caffemodel')

# 이미지 로드
img = cv.imread('city.jpg')
# img = cv.imread('racetrack.jpg')
img = cv.resize(img, (1280, 720))

(h, w) = img.shape[:2]
blob = cv.dnn.blobFromImage(img, 0.007843, (300, 300), 127.5)

# 추론
net.setInput(blob)
detections = net.forward()

# 검출 결과 처리
for i in range(detections.shape[2]):
    confidence = detections[0, 0, i, 2]
    if confidence > 0.5:
        idx = int(detections[0, 0, i, 1])
        box = detections[0, 0, i, 3:7] * [w, h, w, h]
        (startX, startY, endX, endY) = box.astype("int")

        label = f"{class_names[idx]}: {confidence:.2f}"
        cv.rectangle(img, (startX, startY), (endX, endY), (23, 230, 210), 2)
        cv.putText(img, label, (startX, startY - 10),
                   cv.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0), 2)

# 결과 출력
cv.imshow("Object Detection", img)
cv.waitKey(0)
cv.destroyAllWindows()
```

참고

- 모델 다운로드 링크:
  - https://github.com/chuanqi305/MobileNet-SSD
- OpenCV DNN 모듈은 Caffe, TensorFlow, ONNX 등의 다양한 모델 지원

YOLO (You Only Look Once)

Goal

- YOLO 객체 검출 알고리즘의 개념 이해
- 실시간 객체 검출 예제 실습

YOLO란?

- "You Only Look Once"의 약자
- 입력 이미지를 한 번만 통과시켜 객체의 **클래스와 위치를 동시에 예측**하는 딥러닝 기반 알고리즘
- 대표적인 실시간 객체 검출 모델

YOLO 특징

항목	설명
속도	매우 빠름 (실시간 처리 가능)
정확도	SSD보다 높은 정확도 (버전에 따라 다름)
사용 구조	단일 CNN으로 위치 + 클래스 동시 예측 > `1-stage segmentation`

YOLO 처리 흐름

1. 이미지를 여러 그리드로 분할
2. 각 그리드에서 박스 좌표와 클래스 확률 예측
3. Non-Maximum Suppression(NMS)으로 중복 제거
4. 최종 객체 위치와 클래스 결정

YOLOv3 모델을 OpenCV로 사용하기

- 필요한 파일:
  - `yolov3.cfg` (네트워크 구성)
  - `yolov3.weights` (사전 학습된 가중치)
  - `coco.names` (클래스 이름)

```python
from ultralytics import YOLO
from PIL import Image

# 모델 로드 (YOLOv5 또는 YOLOv8 사용 가능)
model = YOLO('yolov5s.pt')  # 또는 'yolov8n.pt'

# 이미지 객체 검출
results = model('city.jpg')  # 경로 또는 numpy 배열 입력 가능
results[0].show()  # PIL 이미지 윈도우 출력

for r in results:
    for box in r.boxes:
        cls_id = int(box.cls[0])
        conf = float(box.conf[0])
        xyxy = box.xyxy[0].tolist()  # [x1, y1, x2, y2]
        print(f"Class: {model.names[cls_id]}, Conf: {conf:.2f}, Box: {xyxy}")
```

Labeling (레이블링)

Goal

- 객체 탐지 및 분류 모델을 위한 레이블링(Labeling)의 개념과 방법 이해

레이블링(Labeling)이란?

- 이미지나 비디오에서 **객체의 위치와 클래스 정보를 수작업으로 부여하는 작업**
- 지도 학습(Supervised Learning)을 위한 **학습 데이터 생성의 핵심 단계**
- 대표적인 레이블링 형태:
  - 이미지 분류: 이미지 하나당 클래스 하나 지정
  - 객체 검출: 바운딩 박스 + 클래스
  - 세분화(Segmentation): 픽셀 단위 클래스 할당

레이블링 도구

도구 이름	특징
LabelImg	바운딩 박스 기반, YOLO/Pascal VOC 지원
CVAT	웹 기반 협업 레이블링 도구
Label Studio	다양한 포맷과 태스크 지원

좋은 레이블링의 조건

- **정확성**: 객체의 정확한 위치와 클래스 지정
- **일관성**: 동일 클래스에 대해 일관된 기준 적용
- **형식 통일**: 모델 입력 형식에 맞게 저장
- **검토 과정**: 교차 검토 등을 통해 오류 최소화

labelimg

다운로드드 링크

- **labelimg** : https://github.com/HumanSignal/labelImg

사용법 시연

1. **predefined_classes.txt** 수정
2. labeling 진행