프로젝트 결과 보고서

[신호등 & 표지판에 대한 사물 인식 결과 출력]

수 강 과 목 : 딥러닝 프로그래밍

담 당 교 수 : 차대현 교수님

팀소개 (7조)

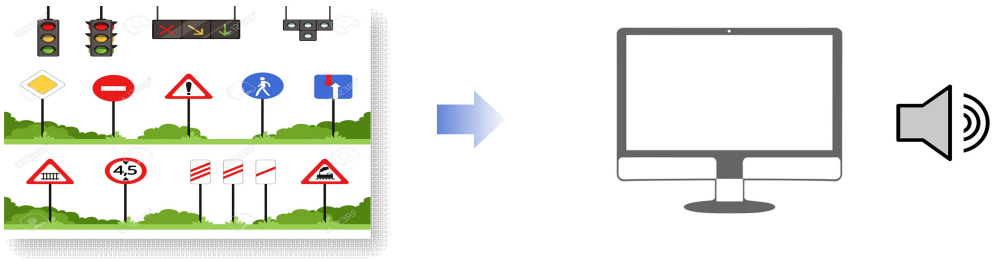
팀원(PL) : 20222749 소프트웨어학과 임혜진

팀원(PM) : 20191436 인공지능학과 김상훈

<목 차>

1. 프로젝트 개요

□ 1-1) 프로젝트 목표



① 정확한 신호등 및 표지판 인식

– 우회전/좌회전 화살표, 신호등, 속도 제한 등 안전 운전에 필수적인 신호 및 표지판에 대한 인식 정확도를 높이는 것에 집중한다.

② 실시간 인식 및 응답

– 실시간으로 보이는 영상에서 신호등 및 표지판을 빠르게 인식한다.

– 화면 및 사운드로 출력하는 것을 목표로 설정한다.

③ 인식된 신호등 및 표지판 정보를 운전자에게 명확하고 직관적인 인터페이스를 설계한다.

2. 프로젝트 구현

□ 2-1) 프로젝트 상세 내용

① 구현 순서

■ 전체 구성 요소

1. 환경 설정

- Python 및 필수 라이브러리 설치
- YOLOv5 설정 및 모델 다운로드
- OpenCV 및 기타 관련 라이브러리 설치

2. 데이터 수집 및 준비

- 표지판 데이터 수집
- 데이터 라벨링
- 데이터셋 준비 (train/val 분할)

3. 모델 학습

- 데이터셋을 이용한 모델 학습
- 학습된 모델 저장

4. 실시간 인식 시스템 구현

- 카메라 연결 및 영상 캡처
- YOLOv5 모델을 이용한 실시간 객체 감지
- 감지된 표지판에 대한 화면 출력 및 사운드 알림

② 상세 내용

1. 환경설정

1) YOLOv5 설치

- yolov5 코드를 다운로드 받은 경로에서 작업

```
git clone https://github.com/ultralytics/yolov5  # clone
cd yolov5
pip install -r requirements.txt  # install module
```

2. 데이터 수집 및 준비

1) 표지판 데이터 수집

- 실제 표지판 이미지 또는 공개된 표지판 데이터셋 수집
  (German_Traffic_Sign_Recognitioni_Benchmark)
- Raboflow의 이미지 데이터셋 사용

2) 데이터 라벨링

- 코드를 작성하여 라벨링하려 했지만 불가능하다고 판단.
- 공용 데이터셋과 라벨링 사이트(Labelimg)를 이용하여 이미지 라벨링

```
# German_Traffic_Sign_Recognition_Benchmark 각 이미지에 대한 라벨링 코드
(사용 x)
import os
import cv2
def create_label_file(image_path, label_path):
    # 이미지 파일 읽어오기
    image = cv2.imread(image_path)
    if image is None:
        print(f"Failed to read image: {image_path}")
        return  # 이미지를 읽어오지 못하면 함수 종료
    height, width, _ = image.shape
    # 중심 좌표 및 박스 크기 계산 (여기서는 이미지의 중심을 박스의 중심으로 설정)
    center_x = width /2
    center_y = height /2
    box_width = width *0.5
    box_height = height *0.5
    # 좌표 정규화
    center_x /= width
```

```
    center_y /= height
    box_width /= width
    box_height /= height
    # 클래스 ID와 박스 좌표를 YOLO 형식으로 작성하여 라벨 파일에 쓰기
    with open(label_path, 'w') as f:
        f.write(f"0 {center_x} {center_y} {box_width} {box_height}\n")
# 이미지 파일이 위치한 디렉토리 경로
image_dir ="D:/yolov5-project/yolov5-master/dataset/val/images"
# 라벨 파일을 저장할 디렉토리 경로
label_dir ="D:/yolov5-project/yolov5-master/dataset/val/labels"
# 라벨 파일을 저장할 디렉토리 생성
os.makedirs(label_dir, exist_ok=True)
# 이미지 파일별로 라벨 파일 생성
for root, _, files in os.walk(image_dir):
    for file in files:
        if file.endswith(".jpg") or file.endswith(".png"):
            image_path = os.path.join(root, file)
            label_file = os.path.splitext(file)[0] +".txt"
            label_path = os.path.join(label_dir, label_file)
            create_label_file(image_path, label_path)
print("라벨 파일 생성이 완료되었습니다.")
```

3) 데이터셋 준비

- yolov5 형식에 맞게 데이터셋 준비(train, val로 분할)
- 실제 표지판 이미지와 공개된 표지판 데이터셋 수집

```
# 디렉토리 구조 예시
dataset/
├── train/
│   ├── images/
│   │   └── 0001.png
│   │       .
│   │       .
│   └── labels/
│       └── 0001.txt
│           .
│           .
```

```
└── val/
    ├── images/
    │   └── 00001.png
    │       .
    │       .
    └── labels/
        └── 00001.txt
            .
            .
```

3. 모델 학습

1) Dataset 경로 설정 및 학습

- *D:\yolov5-project\yolov5-master\dataset\data.yaml*

  파일 생성 및 경로 설정

```
train: D:\yolov5-master\yolov5-master\dataset\train\images
val: D:\yolov5-master\yolov5-master\dataset\val\images
nc: 9
names: ['Speed limit: 30km/h', 'Speed limit: 60km/h', 'Speed limit: 100km/h',
'Speed limit: 120km/h', 'Stop!', "Children Protecting Area",
        'Turn left', 'Turn right', 'Go Straight']
```

- yolov5s.yaml (yolov5에서 주어진 yaml 파일)

  => data.yaml 파일과 nc값 맞추기

```
# YOLOv5 🚀 by Ultralytics, AGPL-3.0 license
# Parameters
nc: 9 # number of classes
depth_multiple: 0.33 # model depth multiple
width_multiple: 0.50 # layer channel multiple
anchors:
  - [10, 13, 16, 30, 33, 23] # P3/8
  - [30, 61, 62, 45, 59, 119] # P4/16
  - [116, 90, 156, 198, 373, 326] # P5/32
# YOLOv5 v6.0 backbone
```

```yaml
backbone:
  # [from, number, module, args]
  [
    [-1, 1, Conv, [64, 6, 2, 2]], # 0-P1/2
    [-1, 1, Conv, [128, 3, 2]], # 1-P2/4
    [-1, 3, C3, [128]],
    [-1, 1, Conv, [256, 3, 2]], # 3-P3/8
    [-1, 6, C3, [256]],
    [-1, 1, Conv, [512, 3, 2]], # 5-P4/16
    [-1, 9, C3, [512]],
    [-1, 1, Conv, [1024, 3, 2]], # 7-P5/32
    [-1, 3, C3, [1024]],
    [-1, 1, SPPF, [1024, 5]], # 9
  ]
# YOLOv5 v6.0 head
head: [
    [-1, 1, Conv, [512, 1, 1]],
    [-1, 1, nn.Upsample, [None, 2, "nearest"]],
    [[-1, 6], 1, Concat, [1]], # cat backbone P4
    [-1, 3, C3, [512, False]], # 13
    [-1, 1, Conv, [256, 1, 1]],
    [-1, 1, nn.Upsample, [None, 2, "nearest"]],
    [[-1, 4], 1, Concat, [1]], # cat backbone P3
    [-1, 3, C3, [256, False]], # 17 (P3/8-small)
    [-1, 1, Conv, [256, 3, 2]],
    [[-1, 14], 1, Concat, [1]], # cat head P4
    [-1, 3, C3, [512, False]], # 20 (P4/16-medium)
    [-1, 1, Conv, [512, 3, 2]],
    [[-1, 10], 1, Concat, [1]], # cat head P5
    [-1, 3, C3, [1024, False]], # 23 (P5/32-large)
    [[17, 20, 23], 1, Detect, [nc, anchors]], # Detect(P3, P4, P5)
  ]
```

- 모델 학습 실행

```
yolovt-procjet\yolov5-master python train.py --img 416 --batch 16
--epochs 50 --data ./dataset/data.yaml --cfg ./models/yolov5s.yaml
--weights yolov5s.pt --name Traffic_sign_yolov5s_results
```

--img 416

=〉 입력 이미지 크기 지정, 416*416 픽셀로 리사이즈

--batch 16

=〉 배치 크기 지정, 한번의 학습 스텝에서 사용할 이미지의 수 16으로 설정

--epochs 50

=〉 전체 데이터셋을 반복하여 학습하는 횟수, 에폭(epoch)은 전체 데이터셋을 한 번씩 모델에 통과시키는 것을 의미

./dataset/data.yaml

=〉 모델 구성 파일 경로, YOLOv5s 모델의 아키텍처와 하이퍼파라미터 지정

--weights yolov5s.pt

=〉 사전 학습된 가중치 파일 경로 지정, yolov5s.pt 파일 사용해 YOLOv5s 모델의 사전 학습된 가중치를 로드

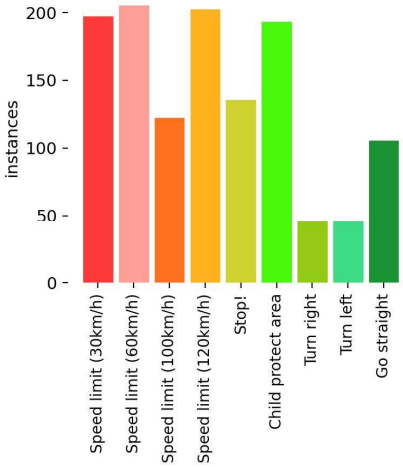
2) 학습된 모델 저장

- *D:\yolov5-project\yolov5-master\runs\train* 해당 경로에 저장

3) 학습 모델에 대한 설명

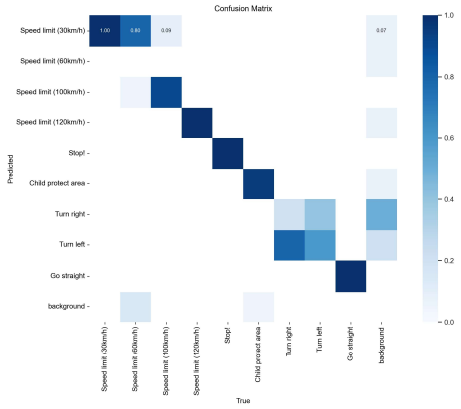
① labels.jpg:

- 데이터셋에 포함된 라벨(클래스)의 분포를 보여주는 이미지



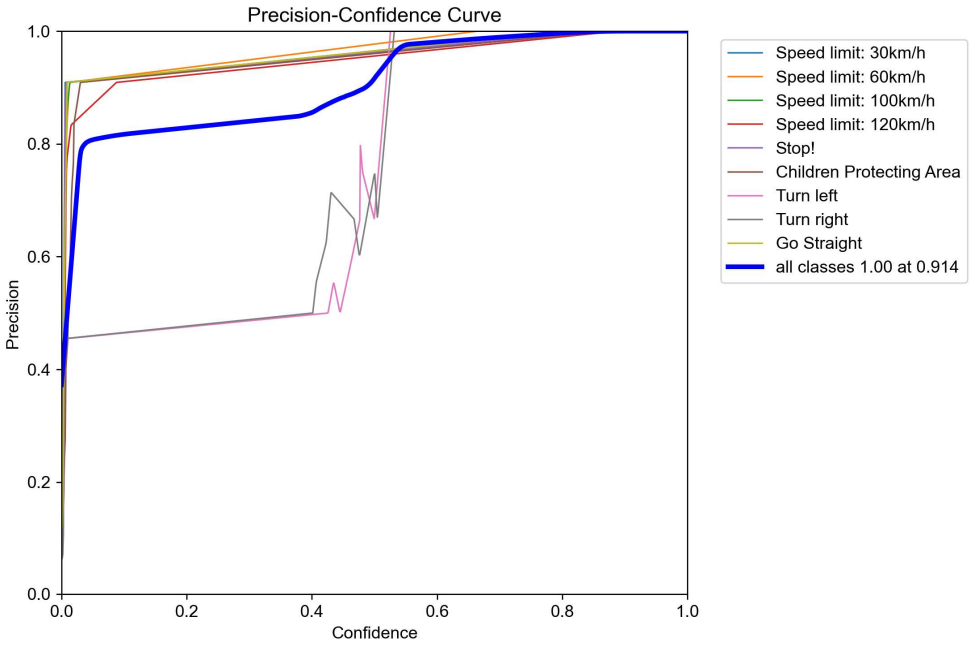
② confusion_matrix.jpg:

- 혼동 행렬은 분류 모델의 예측 결과를 실제 레이블과 비교하여 시각화한 이미지입니다.



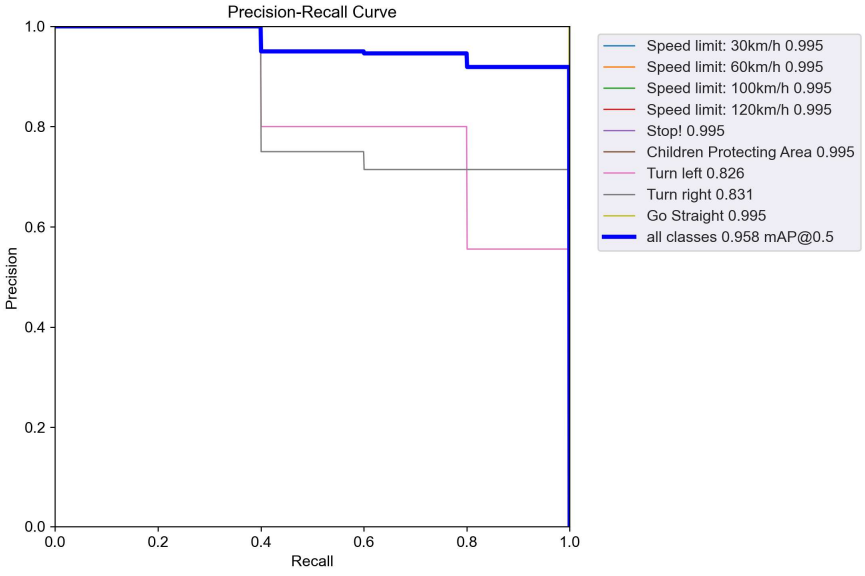
③ P_curve.jpg(Precision Curve):

- 에포크(학습 단계)별로 모델의 정밀도(Precision)의 변화를 나타내는 그래프 (정밀도: 모델이 예측한 긍정 샘플 중 실제로 긍정인 샘플의 비율)



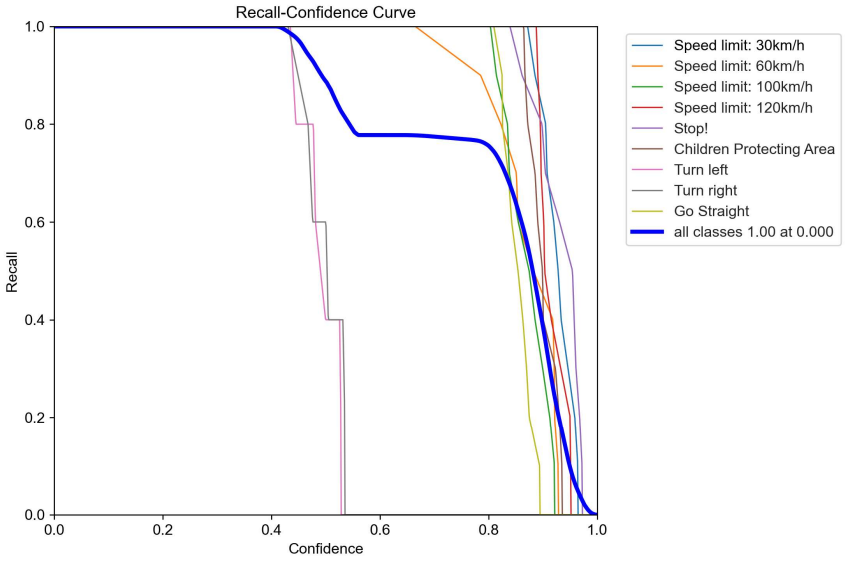
④ PR_curve.jpg(Precision–Recall Curve):

- 모델의 정밀도(Precision)와 재현율(Recall)의 관계를 시각화한 그래프 (임계값(threshold)에 따라 Precision과 Recall이 어떻게 변하는지를 보여줌)



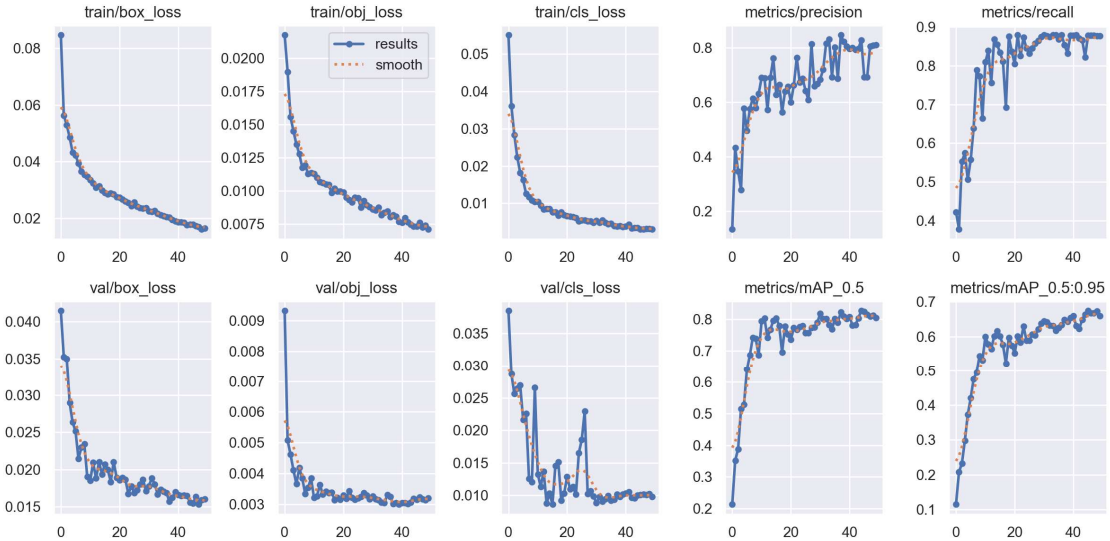
⑤ R_curve.jpg(Recall Curve):

- 에포크별로 모델의 재현율(Recall)의 변화를 나타내는 그래프
  (재현율: 실제 긍정 샘플 중 모델이 올바르게 예측한 비율)



⑥ results.jpg:

- 학습 과정 동안의 주요 성능 지표들을 종합적으로 시각화한 그래프
- 손실(Loss), mAP(mean Average Precision), Precision, Recall 등을 포함
- 해당 그래프를 통해 학습의 진행 상황과 모델의 성능 변화를 확인 가능



4. 실시간 인식 시스템 구현

1) 학습된 모델을 이용해 실시간 객체 탐지

- 필요 라이브러리 설치

```
pip install gtts
pip install openCV-python
```

(1) 카메라를 통한 표지판 인식 코드(yolov5_opencv)

```
import torch
import cv2
# 모델 경로 확인
model_path='D:/yolov5-project/yolov5-master/runs/train/Project_Result_9_class/weights/best.pt'
# 모델 로드
model = torch.hub.load('ultralytics/yolov5', 'custom', path=model_path, force_reload=True)
# 카메라 설정
cap = cv2.VideoCapture(0)  # 0번 카메라를 사용하여 영상을 가져옵니다.
if not cap.isOpened():
    print("Error: Could not open camera.")
    exit()
while True:
    # 프레임 읽기
    ret, frame = cap.read()
    if not ret:
        print("Error: Could not read frame.")
        break
    # YOLOv5 모델에 프레임 전달
    results = model(frame)
    # 지된 객체 그리기
    results.render()  # results.render()는 이미지를 직접 수정합니다
    # 수정된 프레임을 표시
    cv2.imshow('YOLOv5 Detection', frame)
    # 'q' 키를 누르면 종료
    if cv2.waitKey(1) &0xFF == ord('q'):
        break
# 카메라 종료
cap.release()
cv2.destroyAllWindows()
```

(2) 저장된 mp4 파일에서 표지판 객체 탐지(yolov5_opencv_video)

```
import torch
import cv2
# 모델 경로 확인
model_path='D:/yolov5-master/yolov5-master/runs/train/Project_Result/weights/best.pt'
# 모델 로드
model = torch.hub.load('ultralytics/yolov5', 'custom', path=model_path, force_reload=True)
# 동영상 경로
video_path ='C:/Users/shkim/OneDrive/바탕 화면/100, 30.mp4'  # 동영상 경로를 적절히 수정하세요
#video_path = 'C:/Users/shkim/OneDrive/바탕 화면/30, left.mp4'  # 동영상 경로를 적절히 수정하세요
#video_path = 'C:/Users/shkim/OneDrive/바탕 화면/protect.mp4'  # 동영상 경로를 적절히 수정하세요
#video_path = 'C:/Users/shkim/OneDrive/바탕 화면/left, right, straight.mp4'  # 동영상 경로를 적절히 수정하세요
# 동영상 읽기
cap = cv2.VideoCapture(video_path)
if not cap.isOpened():
    print("Error: Could not open video.")
    exit()
# 프레임 건너뛰기 간격
frame_skip_interval =2  # 예를 들어, 매 2번째 프레임마다 처리
frame_count =0
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break
    # YOLOv5 모델에 프레임 전달
    results = model(frame)
    # 감지된 객체 그리기
    results.render()  # results.render()는 프레임을 직접 수정합니다
    # 수정된 프레임 표시
    cv2.imshow('YOLOv5 Detection', frame)
    # 키 입력 대기 및 ESC 키 누르면 종료
    if cv2.waitKey(1) &0xFF ==27:
        break
# 자원 해제
```

```
cap.release()
cv2.destroyAllWindows()
```

(3) 카메라를 통해 표지판 탐지 후, 사운드로 출력 (yolov5_sound)

```
import torch
import cv2
from gtts import gTTS
import os
# 모델 경로 확인
model_path='D:/yolov5-project/yolov5-master/runs/train/Project_Result_9_class/weights/best.pt'
# 모델 로드
model = torch.hub.load('ultralytics/yolov5', 'custom', path=model_path, force_reload=True)
# 표지판을 인식한 내용을 저장할 변수를 초기화
last_sign_label =None
# 텍스트를 음성으로 변환하여 재생하는 함수
def play_sound(text):
    # gTTS를 사용하여 텍스트를 음성으로 변환(ko => 한글)
    tts = gTTS(text=text, lang="ko")
    tts.save("temp.mp3")
    # 저장된 음성 파일을 재생합니다.
    os.system("start temp.mp3")
# 메인 함수를 정의합니다.
def main():
    global last_sign_label
    # 카메라를 초기화합니다.
    cap = cv2.VideoCapture(0)
    # 카메라 프레임을 읽어오고 표지판을 인식하여 해당하는 텍스트 추출, 음성 출력
    while True:
        ret, frame = cap.read()
        if not ret:
            print("카메라에서 프레임을 읽을 수 없습니다.")
            break
        results = model(frame)
        # 탐지된 객체를 순회하면서 표지판 찾기
        for detection in results.xyxy[0]:
            label = model.names[int(detection[5])]  # 객체 라벨
            if "Speed limit"in label or "Protecting Area"in label:
```

```
                # 표지판의 내용을 추출합니다.
                text = label.replace("(30km/h)", "30km/h").replace("(60km/h)", "60km/h").replace("(100km/h)", "100km/h").replace("(120km/h)", "120km/h").replace("Protecting Area", "어린이 보호구역").replace("Turn left", "좌회전").replace("Turn right", "우회전").replace("Go straight", "직진")
                # 이전에 인식한 표지판과 다르다면 해당 내용을 음성으로 출력
                if last_sign_label is None or text != last_sign_label:
                    play_sound(text)
                    last_sign_label = text
                # 객체의 바운딩 박스를 화면에 표시
                x1, y1, x2, y2 =int(detection[0]), int(detection[1]), int(detection[2]), int(detection[3])
                cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)  # 바운딩 박스 그리기
                cv2.putText(frame, text, (x1, y1 -10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 255, 0), 2)  # 텍스트 그리기
        # 프레임 출력
        cv2.imshow('YOLOv5 Detection', frame)
        # 'q' 키를 누르면 프로그램을 종료
        if cv2.waitKey(1) &0xFF == ord('q'):
            break
    cap.release()
    cv2.destroyAllWindows()
if __name__ =="__main__":
    main()
```