

# 객체 탐지 알고리즘 YOLO를 활용한 보행 장애인 탐지 및 알림 모델 구현

팀명: DSplay

2019310444 김효원  
2019312564 노최유하  
2019311621 송재현  
2019313365 양지인



# Contents

---

- 01 Background
- 02 Dataset
- 03 Preprocess the Data
- 04 Model - YOLOv5
- 05 Result
- 06 Conclusion
- 07 Reference

# 01 Background

## Problem

보행 약자: 독립보행이 어려운 자 (ex. 어린이, 노인, 휠체어 사용자 등)

보행 약자는 일반인보다 낮은 시점을 가짐

⇒ 시야의 범위가 좁아지거나 시야에 왜곡이 생겨 위험 요소를 감지하기 어려움

보행 약자는 신체 활동에 제약이 있음

⇒ 반응 속도가 낮아서 보행 중 장애물과 충돌하기 전에 피하기 어려움

## Solution

보행 약자가 장애물을 인지하여 피할 수 있도록 1인칭 시점 보행 영상 데이터를 활용

YOLO를 활용하여 보행 도로 위 위험 요소 객체 탐지 모델 구축



# 01 Background

## Previous research

- 보통 시야가 제한적인 시각 장애인 대상이거나 평균적인 신장의 시야에 맞춤
- 대부분 고정적인 위치에 있는 cctv 영상 데이터를 활용함
- 탐지 대상 장애물 미리 선정

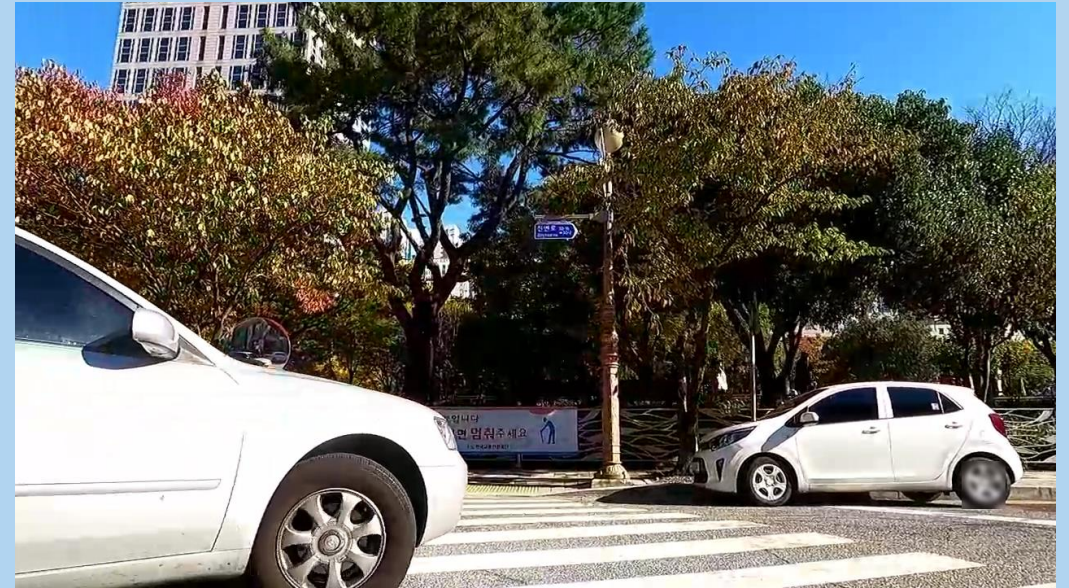
## What's new

- 노약자나 어린이 같이 평균 신장보다 **작은 사람들의 시야**에서 데이터 분석
- **1인칭 보행자 시점**에서 움직인 영상 프레임을 사용하여 실제 보행 중 마주치는 위험 요소 탐지
- 장애물 인식 대상을 사전에 한정하지 않고 **객체의 근접성을 판단**하여 위험 장애물인지 판단

# 02 Dataset



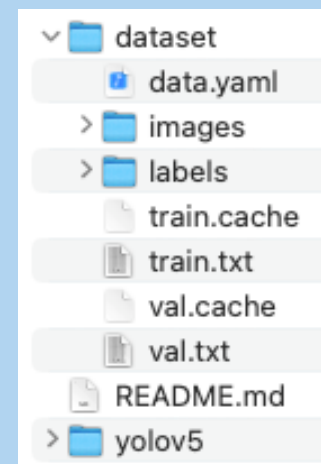
- 1인칭 시점 보행영상 데이터
- 시점: 사회적 약자 시점 80cm
- 장소: 실외
- 구성
  - 1인칭 시점 보행 영상에 대한 캡처 이미지 파일
  - 이미지와 이미지 내 방해물에 대한 JSON 파일
  - 각 5686개



▲ 1인칭 시점 보행 영상 이미지 예시

# 03 Preprocess the Data

- YOLOv5 모델 활용 <https://github.com/ultralytics/yolov5>
  - multi object detection이 가능하고 성능이 좋다고 알려진 객체 탐지 모델
- YOLOv5 모델 학습을 위해 필요한 데이터셋 구성
  - images: 원본 이미지(.jpg)
  - labels: bounding box(bbox)에 대한 라벨링 정보(.txt)
    - 하나의 텍스트 파일엔 하나의 이미지에서 탐지되는 객체의 bbox 정보가 각각 한 줄에 담겨 있음
    - bbox 정보: 클래스 번호, x 좌표, y 좌표, 너비, 높이
  - data.yaml: 학습 데이터/검증 데이터의 경로와 클래스 개수, 클래스 명



▲YOLOv5 모델 학습을 위해 필요한 데이터셋 구성

# 03 Preprocess the Data

## ① JSON 파일을 yaml 파일로 변환

```
# json 파일을 yaml 파일로
import yaml
import json
from glob import glob

%cd /content/drive/Othercomputers/Mymac/new/json/
json_list = glob('*.*json')
print(json_list)
for i in range(len(json_list)):
    with open(json_list[i], 'r') as file:
        config = json.load(file)
        file_name = json_list[i].split('.')[0]
        with open('/content/drive/Othercomputers/Mymac/new/yaml/' + file_name + '.yaml', 'w') as yaml_file:
            yaml.dump(config, yaml_file)
```

## ② bbox 좌상단 좌표값을 중앙 좌표값으로 변환 bbox center (x, y), w, h 값 정규화

```
# x_center, y_center 구하는 함수
def tocenter(x,y,w,h):
    x1=x+w
    y1=y+h
    cx = (x+x1)/2
    cy = (y+y1)/2

    # normalize
    nx = cx/1280
    ny = cy/720
    nw = w/1280
    nh = h/720

    return nx, ny, nw, nh
```

```
annotation:
  annotations:
  - atchFileId: null
    atchFileName: 21d0f65e669a43519b4748d674833818.png
    atchFilePath: ./JID_001600337/
    atchFileSize: 1458323
    atchOrgFileName: scene00001.png
    box:
      - box:
          h: 113.99976
          w: 220.99968
          x: 140.99967999999998
          y: 354.99996
          category_id: f_o_66
          category_name: car
        - box:
          h: 463.00032
          w: 84.99993599999999
          x: 396.99952000000005
          y: 3.999600000000023
          category_id: f_o_68
          category_name: powerpole
        boxCount: 2
        boxesJson: null
        contextId: WCID_001059753
        frame: 1
```

▲yaml 형태로 변환한 라벨링 파일

# 03 Preprocess the Data

## ③ txt 파일 구성

```
# yaml 파일에서 필요한 데이터만 추출 = x, y, w, h, 클래스넘버, 클래스이름
# txt 파일 구성: class number, x_center, y_center, width, height (normalized)
import yaml
import os

%cd /content/drive/0thercomputers/Mymac/new/yaml/
yaml_list = glob('*.*.yaml')
class_name = []
index_list = []
for yaml_name in yaml_list:
    with open(yaml_name, 'r') as file:
        yaml_data = yaml.safe_load(file)
        annotation = yaml_data['annotation']
        ans = annotation['annotations'] # 각 annotations가 저장되어 있는 리스트
        front = yaml_name.split('.')[0]

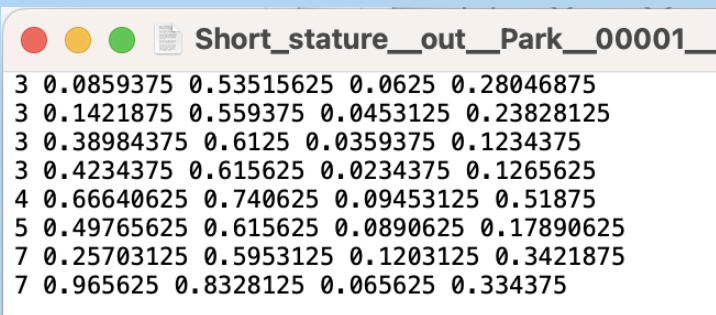
        #print(len(ans))
        for a in range(len(ans)):
            boxs = ans[a].get('box') # 하나의 annotation에서 box 정보만 추출 (리스트 저장) -> 즉 이미지 하나에 있는 box들
            box_info = []
            scene_name = ans[a].get('attachOrgFileName').split('.')[0]
            if '__' in scene_name:
                txt_name = scene_name
            else:
                txt_name = front + '__' + scene_name
```

```
for b in range(len(boxs)): # 한 개 box의 정보
    box = boxs[b]
    #print(box)
    # 클래스 이름 리스트
    cname = box.get('category_name')
    if cname not in class_name:
        class_name.append(cname)

    # x, y, w, h 각각 리스트로
    box_data = box.get('box')
    #print(box_data)
    x, y, w, h = box_data.get('x'), box_data.get('y'), box_data.get('w'), box_data.get('h')
    # x_center, y_center 함수 불러오기 -> 리스트 저장
    nx, ny, nw, nh = tocenter(x,y,w,h)
    box_coordinate = [class_name.index(cname), nx, ny, nw, nh]
    box_info.append(box_coordinate)

if len(box_info) == 0:
    index_list.append(txt_name)

# txt 파일에 저장
f = open('/content/drive/0thercomputers/Mymac/new/labels/' + txt_name + '.txt', 'a')
for i in box_info:
    value = ' '.join(list(map(str, i)))
    f.write(value + '\n')
f.close()
```



```
Short_stature__out__Park__00001__
3 0.0859375 0.53515625 0.0625 0.28046875
3 0.1421875 0.559375 0.0453125 0.23828125
3 0.38984375 0.6125 0.0359375 0.1234375
3 0.4234375 0.615625 0.0234375 0.1265625
4 0.66640625 0.740625 0.09453125 0.51875
5 0.49765625 0.615625 0.0890625 0.17890625
7 0.25703125 0.5953125 0.1203125 0.3421875
7 0.965625 0.8328125 0.065625 0.334375
```

▲ 한 이미지에서 탐지된 방해물에 대한 bbox 정보 txt 파일



# 03 Preprocess the Data

## ④ data.yaml 파일 구성

```
# yaml 파일에 필요한 정보: 학습데이터 경로, 클래스 개수, 클래스 이름
import yaml
%cd /content/dataset/
with open('data.yaml', 'r') as f:
    data = yaml.safe_load(f)

print(data)

data['train'] = '/content/dataset/train.txt'
data['val'] = '/content/dataset/val.txt'
data['nc'] = len(class_name)
data['names'] = class_name

with open('/content/dataset/data.yaml', 'w') as f:
    yaml.dump(data, f)

print(data)
```

```
1 names:
2   - person
3   - powerpole
4   - motorcycle
5   - car
6   - chair
7   - signboard
8   - barrigate
9   - bench
10  - pot
11  - colorcone
12  - bicycle
13  - stop
14  - bollard
15  - tree
16  - trash
17  - sculpture
18  - collectionbox
19  - table
20  - burlapbag
21  - stand
22  - trashcan
23  - box
24  - cart
25  - kickboard
26  nc: 24
27  train: /content/drive/MyDrive/dsplay/dataset/train.txt
28  val: /content/drive/MyDrive/dsplay/dataset/val.txt
```

▲보행 방해물 전체 클래스

# 03 Preprocess the Data

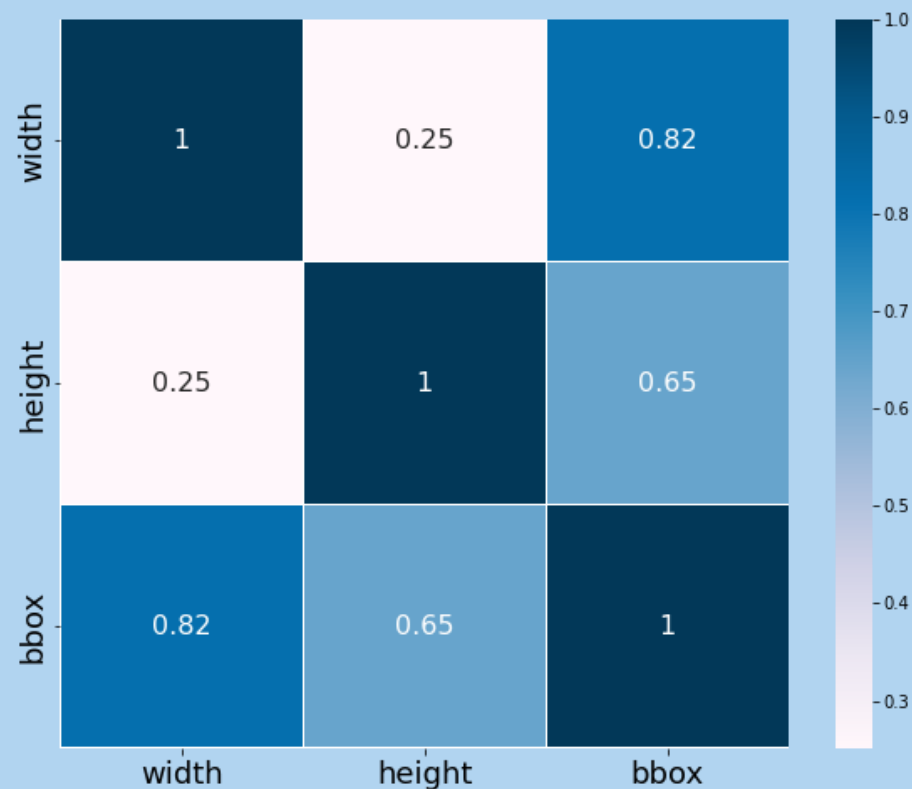
## ⑤ bbox의 면적, 높이, 너비 간의 상관관계 분석

일반적으로 영상 분석에서 위험 요소를 탐지할 때, 객체의 면적을 활용

⇒ 상관관계 분석 결과 **면적과 너비 간 상관계수**가 높게 나타남

⇒ 객체의 너비가 커지면 **위험 상황으로 인식**하여 주의를 줄 수 있는 방안 필요

Correlation of Features



# 04 Model - YOLOv5

## Training

```
%cd /content/drive/MyDrive/dsplay/yolov5/
```

```
!python train.py --img 736 --batch 32 --epochs 300 --data /content/drive/MyDrive/dsplay/dataset/data.yaml --cfg ./models/yolov5s.yaml --weights yolov5s.pt --name result_80cm_736pro
```

학습 결과 요약 ▼

- Hyperparameter setting
  - img: 736
  - batch: 32
  - epoch: 300
  - model: yolov5s
- GPU 사용
- 실행시간: 10.109 hours

```
300 epochs completed in 10.109 hours.
Optimizer stripped from runs/train/result_80cm_736pro/weights/last.pt, 14.5MB
Optimizer stripped from runs/train/result_80cm_736pro/weights/best.pt, 14.5MB

Validating runs/train/result_80cm_736pro/weights/best.pt...
Fusing layers...
YOLOv5s summary: 157 layers, 7074853 parameters, 0 gradients, 16.0 GFLOPs
```

|  | Class         | Images | Instances | P     | R     | mAP50 | mAP50-95: 100% |
|--|---------------|--------|-----------|-------|-------|-------|----------------|
|  | all           | 1138   | 4016      | 0.982 | 0.893 | 0.995 | 0.88           |
|  | person        | 1138   | 1002      | 1     | 0.981 | 0.995 | 0.924          |
|  | powerpole     | 1138   | 659       | 0.992 | 0.978 | 0.994 | 0.906          |
|  | motorcycle    | 1138   | 265       | 0.988 | 0.943 | 0.994 | 0.909          |
|  | car           | 1138   | 1194      | 0.993 | 0.969 | 0.994 | 0.932          |
|  | chair         | 1138   | 185       | 0.998 | 0.984 | 0.995 | 0.923          |
|  | signboard     | 1138   | 227       | 0.995 | 0.907 | 0.993 | 0.807          |
|  | barrigate     | 1138   | 118       | 0.975 | 0.981 | 0.994 | 0.897          |
|  | bench         | 1138   | 36        | 0.995 | 1     | 0.995 | 0.88           |
|  | pot           | 1138   | 14        | 0.991 | 1     | 0.995 | 0.921          |
|  | colorcone     | 1138   | 30        | 1     | 0.985 | 0.995 | 0.923          |
|  | bicycle       | 1138   | 12        | 0.987 | 1     | 0.995 | 0.911          |
|  | stop          | 1138   | 74        | 0.995 | 1     | 0.995 | 0.923          |
|  | bollard       | 1138   | 101       | 0.979 | 0.937 | 0.993 | 0.905          |
|  | tree          | 1138   | 34        | 1     | 0.989 | 0.995 | 0.917          |
|  | trash         | 1138   | 42        | 1     | 0.982 | 0.995 | 0.91           |
|  | collectionbox | 1138   | 6         | 0.996 | 1     | 0.995 | 0.847          |
|  | table         | 1138   | 1         | 0.92  | 1     | 0.995 | 0.796          |
|  | burlapbag     | 1138   | 10        | 0.976 | 1     | 0.995 | 0.946          |
|  | stand         | 1138   | 1         | 1     | 0     | 0.995 | 0.796          |
|  | trashcan      | 1138   | 1         | 0.871 | 1     | 0.995 | 0.796          |
|  | box           | 1138   | 2         | 0.962 | 1     | 0.995 | 0.846          |
|  | kickboard     | 1138   | 2         | 1     | 0     | 0.995 | 0.746          |

```
Results saved to runs/train/result_80cm_736pro
```

# 04 Model - YOLOv5

## Inference

```
val_img_path = val_img_list[25]
```

```
!python detect.py --weights /content/drive/MyDrive/dsplay/yolov5/runs/train/result_80cm_736pro/weights/best.pt --img 736 --conf 0.5 --save-txt --source "{val_img_path}"
```

- best 모델 사용
- confidence score가 0.5 이상인 객체만 표시
- 위험 요소 감지
  - bbox 평균 폭보다 크면 **attention**
  - bbox 평균 면적보다 크면 **warning**
  - 객체명 리스트, 해당 클래스 평균 bbox 면적, 클래스 평균 bbbox 폭 크기가 담긴 클래스 **ClassAvg**를 정의하여 detect.py 파일 수정

detect.py 파일 중 위험 요소 감지 및 알림을 위해 수정한 부분 ▼

```
# Write results
for *xyxy, conf, cls in reversed(det):
    xywh = (xyxy2xywh(torch.tensor(xyxy).view(1, 4)) / gn).view(-1).tolist() # normalized xywh
    bbox = xywh[2]*xywh[3]
    c = int(cls) # integer class

    if save_txt: # Write to file
        with open(f'{txt_path}.txt', 'a') as f:
            line = (cls, *xywh, conf) if save_conf else (cls, *xywh) # label format
            # 클래스의 거리가 가까우면 화면에 경고
            if CLASSAVG.bboxSize[c]<bbox:
                # line = (cls, *xywh, conf, "위험") if save_conf else (cls, *xywh, "위험") # label format
                f.write((' %g ' * len(line)).rstrip() % line + ' 위험\n')

            elif CLASSAVG.widthAvg[c]<xywh[2]:
                # line = (cls, *xywh, conf, "경로 확인 요망") if save_conf else (cls, *xywh, "경로 확인 요망") # label format
                f.write((' %g ' * len(line)).rstrip() % line + ' 경로 확인 요망\n')

            else:
                f.write((' %g ' * len(line)).rstrip() % line + '\n')

    if save_img or save_crop or view_img: # Add bbox to image
        label = "" if hide_labels else (f'{names[c]}' if hide_conf else f'{names[c]} {conf:.2f}')

        # 클래스의 거리가 가까우면 화면에 경고
        if CLASSAVG.bboxSize[c]<bbox:
            label = f'{label} warning'
            annotator.box_label(xyxy, label, color=colors(c, True))

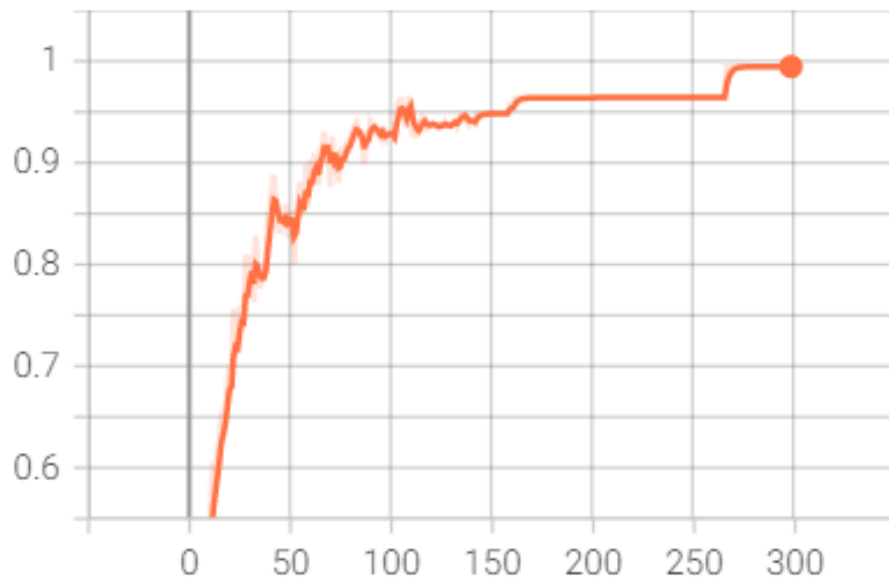
        elif CLASSAVG.widthAvg[c]<xywh[2]:
            label = f'{label} attention'
            annotator.box_label(xyxy, label, color=colors(c, True))

        else:
            annotator.box_label(xyxy, label, color=colors(c, True))
```

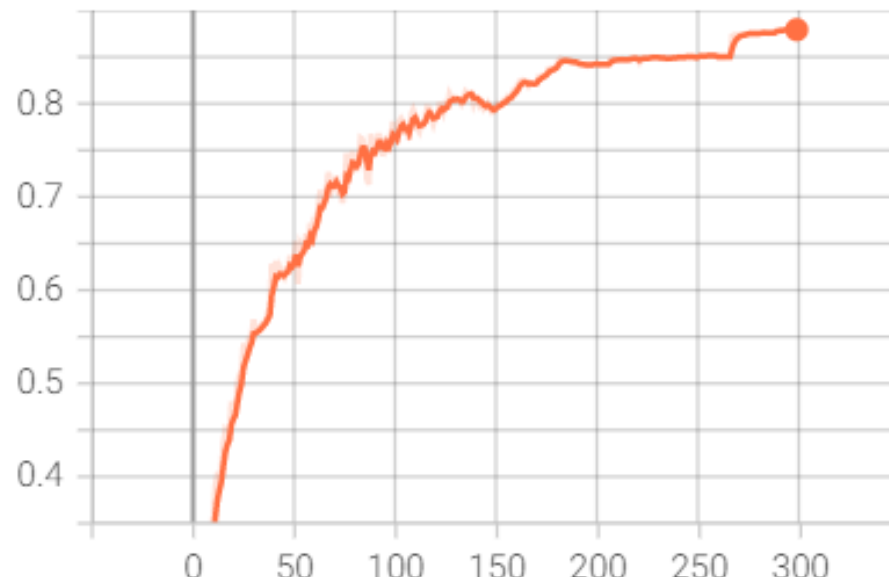
# 05 Result

## ① 모델 성능 평가

metrics/mAP\_0.5  
tag: metrics/mAP\_0.5

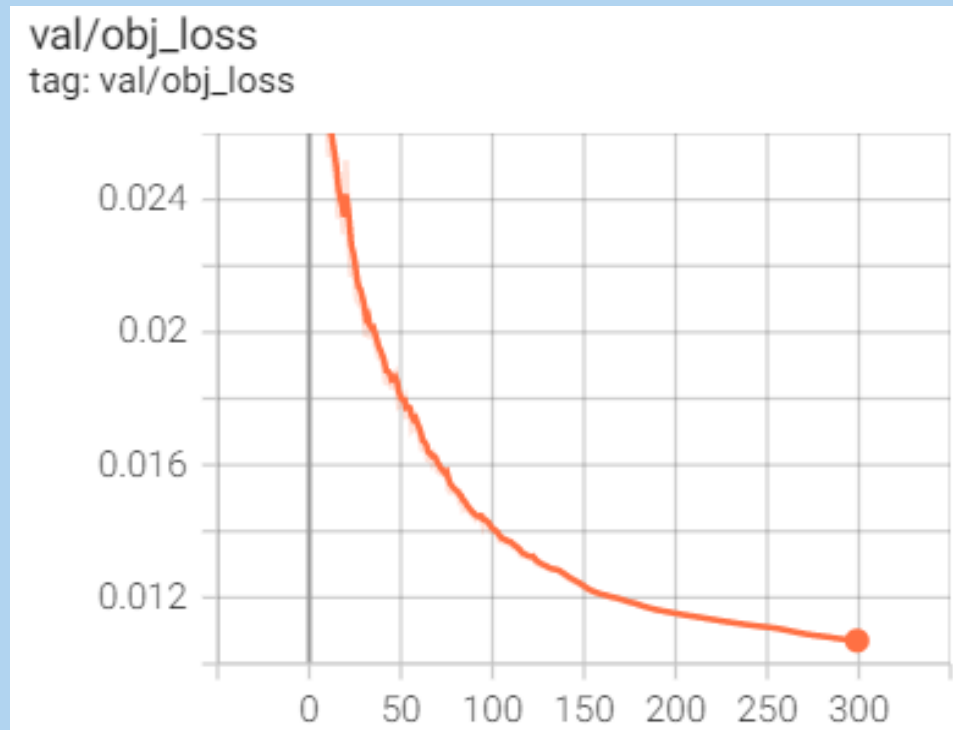
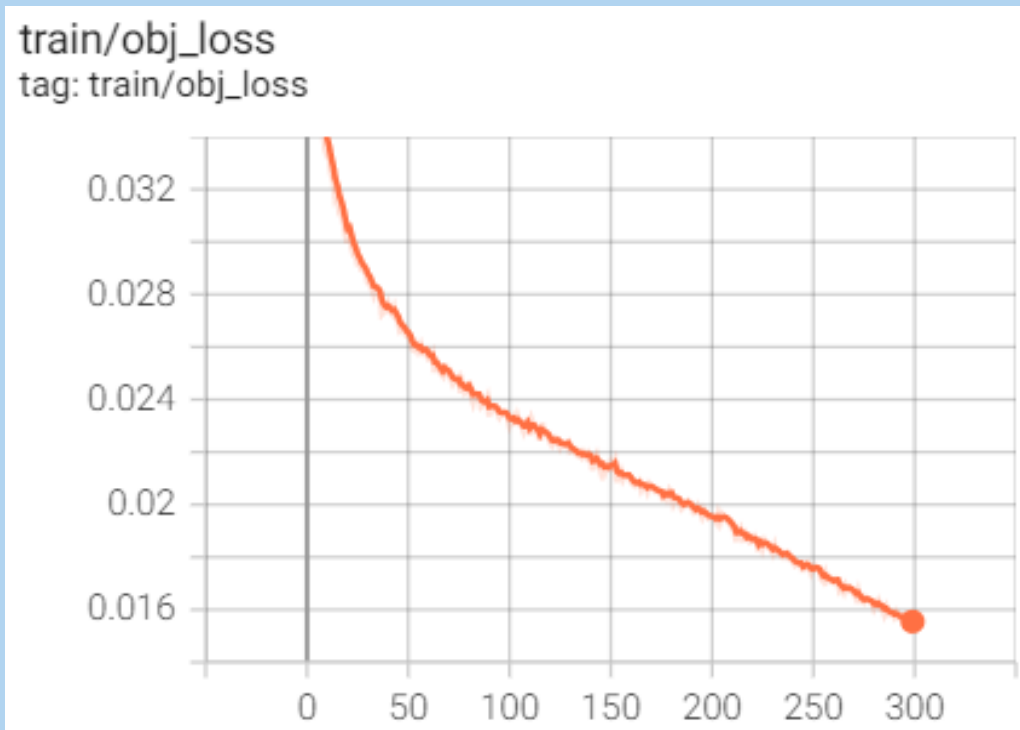


metrics/mAP\_0.5:0.95  
tag: metrics/mAP\_0.5:0.95



# 05 Result

## ① 모델 성능 평가



# 05 Result

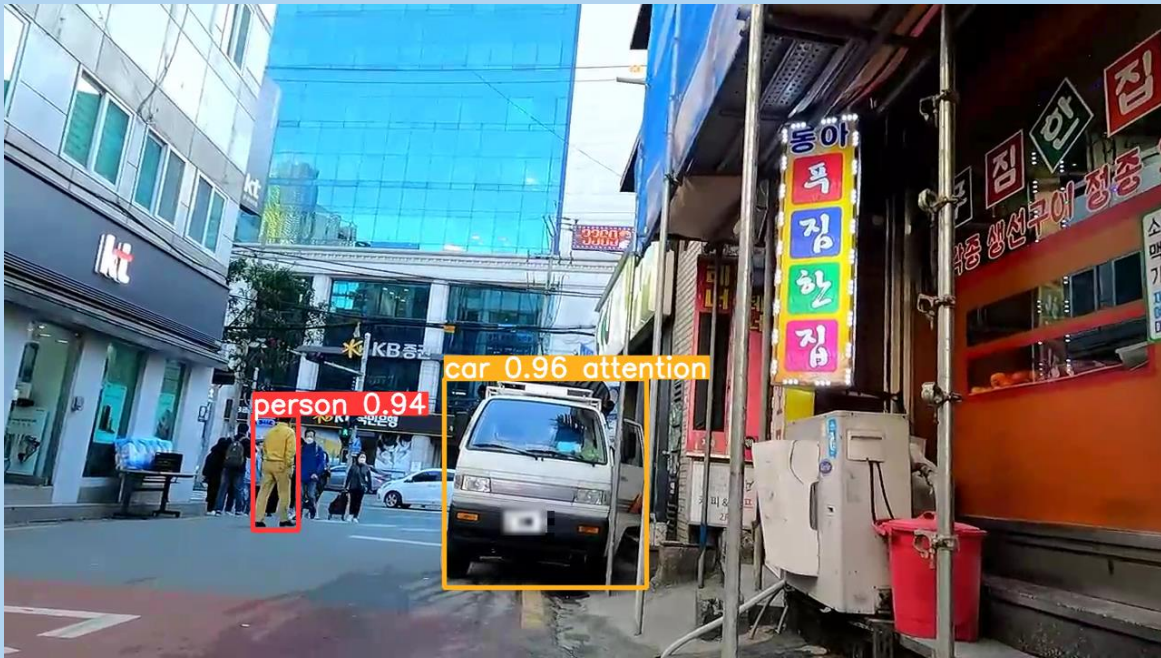
② validation 이미지에 적용





# 05 Result

## ② validation 이미지에 적용





## 05 Result

③ 다른 1인칭 인도 보행 영상 데이터에 학습된 모델을 적용하여 향후 모델의 활용 가능성 검증



# 06 Conclusion

---

## 연구 결과 및 성과

- 보행 약자의 입장에서 마주하는 장애물을 분석하고, 그 결과를 바탕으로 위험 장애물을 탐지해주는 모델을 제시
- 클래스 별로 bbox 면적의 평균값을 구하여, 인식된 특정 객체의 bbox 면적이 해당 클래스의 평균값보다 더 큰 경우 '경고(Warning)', 너비가 더 큰 경우 '도로 주의 요망(Attention)'이라는 문구가 나오는 모델을 구현

## 의의 및 활용 가능성

- 실제 보행자의 1인칭 시점 데이터 사용
- 너비와 거리(bbox 넓이)에 큰 상관관계 발견
- 상용화 용이: 카메라렌즈가 있는 웨어러블 디바이스에 적용 가능

# 07 Reference

---

- 1) Jianrong Cao, Yuan Zhuang, Ming Wang, Xinying Wu, and Fatong Han. 2022. "Pedestrian Detection Algorithm Based on ViBe and YOLO", 2021 The 5th International Conference on Video and Image Processing (ICVIP 2021). Association for Computing Machinery, New York, NY, USA, 92-97. <https://doi-org-ssl.ca.skku.edu/10.1145/3511176.3511191>
- 2) 조수형, 김호진, 박상순, 최유준, 이수원. (2021). 시각장애인을 위한 모바일 기반 장애물 탐지 연구. 한국정보처리학회 학술대회논문집, 28(1): 433-436
- 3) Yolov5 문서 <https://github.com/ultralytics/yolov5>
- 4) Kim, Sang Gu. (2016). Walking Accident Characteristics and Walking Factors for Road Crossing of the Transportation Vulnerable in the Case of Yeosu. *Journal of Digital Convergence*, 14(6), 439-448. <https://doi.org/10.14400/JDC.2016.14.6.439>
- 5) 이소민 and 이명훈. (2021). 생애주기별 사회적 약자의 보행환경 만족도 영향요인 연구 - 근린생활권에서의 보행친화도와 보행네트워크를 중심으로. 도시설계, 22(4), 17-34.

감 사 합 니 다