

[AI 영상인식 특강]

전동킥보드 불법 사용자 적발 .

문 효 원 | 박 유 진 | 이 호 현

[목차 페이지]

01 프로젝트 소개

02 킥보드 회사

03 헬멧 착용 여부

04 HSV 란

05 색깔 추출 코드

06 사람 인식 코드

[전동 킥보드]

01

전동 킥보드 사용자 증가

- 코로나 19 로 인해 개인 이동 수단 사용량 증가
- 이동장치 중 속도가 빠르고 편리하며 접근성이 낮음

02

불법 사용자

- 전동 킥보드의 접근성이 낮다는 점으로 인해 미성년자 사용, 헬멧 미착용, 2인 이상 탑승 등 여러 불법 사용자들 증가



[최종 프로젝트]

“전동 킥보드
불법 사용자
적발

AI 영상인식 특강



전동 킥보드 브랜드 구분

전동 킥보드의 가장 대중적인 회사 GCOO, BEAM, 썬썬의 세 가지 대표 색깔을 추출하여 해당 회사 구분



2 인 이상 탑승자 적발

전동 킥보드는 2 인 이상 탑승이 불가능하기 때문에 이를 AI 를 통하여 사람인지 인식하여 적발



헬멧 미착용자 적발

AI 를 활용하여 사용자의 헬멧 착용 여부를 적발

[HSV]

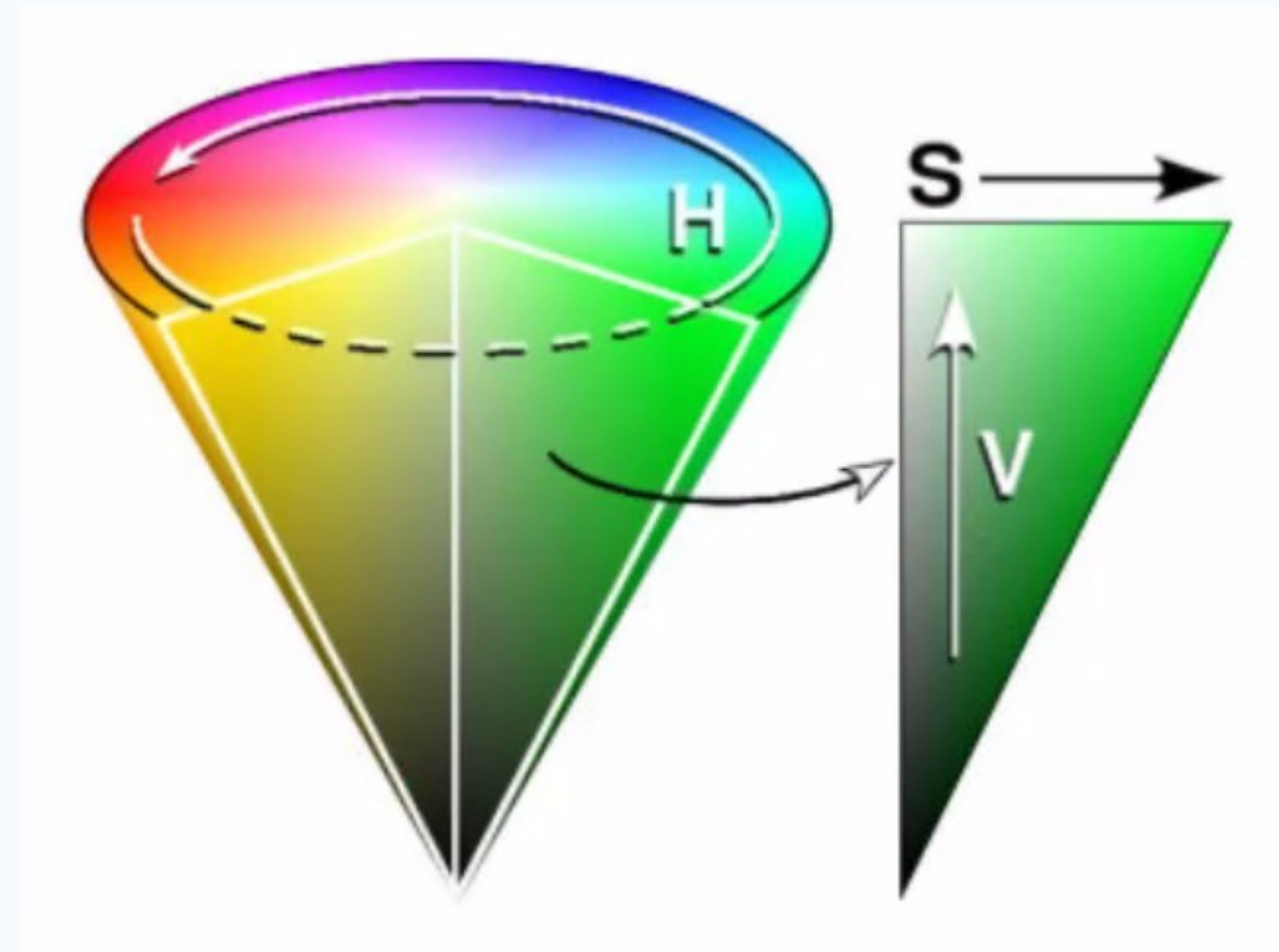
01 색도 & 명도 & 채도

- Hue : 색상의 종류
- Saturation : 색상의 순도
- Value (또는 Brightness) : 색상의 밝기

02 사용 이유

- 명도와 채도를 쉽게 조작할 수 있어 이미지의 특정 색상만을 선택하거나 조정하는 데 유리

[HSV]



■ 계열01 ■ 계열02

[색깔 추출 코드]

```
# 6C00 (초록색)
lower_green = np.array([30, 40, 40])
upper_green = np.array([90, 255, 255])
# Beam (보라색)
lower_purple = np.array([120, 50, 50])
upper_purple = np.array([170, 255, 255])
# 씽씽 (노란색)
lower_yellow = np.array([10, 100, 100])
upper_yellow = np.array([40, 255, 255])

# 각각의 이미지를 HSV로 변환
balloon_hsv1 = cv2.cvtColor(balloon1, cv2.COLOR_BGR2HSV)
balloon_hsv2 = cv2.cvtColor(balloon2, cv2.COLOR_BGR2HSV)
balloon_hsv3 = cv2.cvtColor(balloon3, cv2.COLOR_BGR2HSV)

# 각 이미지에 대해 색상별 마스크 생성 및 적용
mask_green = cv2.inRange(balloon_hsv1, lower_green, upper_green)
result1 = cv2.bitwise_and(balloon1, balloon1, mask=mask_green)

mask_purple = cv2.inRange(balloon_hsv3, lower_purple, upper_purple)
result2 = cv2.bitwise_and(balloon3, balloon3, mask=mask_purple)

mask_yellow = cv2.inRange(balloon_hsv2, lower_yellow, upper_yellow)
result3 = cv2.bitwise_and(balloon2, balloon2, mask=mask_yellow)
```



HSV

세 가지 회사에 대한 대표 색 설정

- HSV 로 세 가지 색 지정
- 각 이미지에 대해 색상별 마스크 생성 및 추출

[사람 인식]

```
# 총 'person' 객체의 수를 계산합니다
person_count = len(persons)

# 텍스트를 추가할 공간을 포함한 최종 이미지를 준비합니다
image_height, image_width = image.shape[:2]
space_height = 50 # 텍스트를 추가할 공간의 높이
final_image = np.ones((image_height + space_height, image_width, 3), dtype=np.uint8) * 255
final_image[:image_height, :] = image

# 2명 이상의 'person' 객체가 있을 때 메시지를 추가합니다
if person_count > 1:
    message = "More than 2 people detected"
    cv2.putText(final_image, message, (10, image_height + space_height - 10),
                cv2.FONT_HERSHEY_SIMPLEX, 1.0, (0, 0, 0), 2, cv2.LINE_AA)

return final_image
```

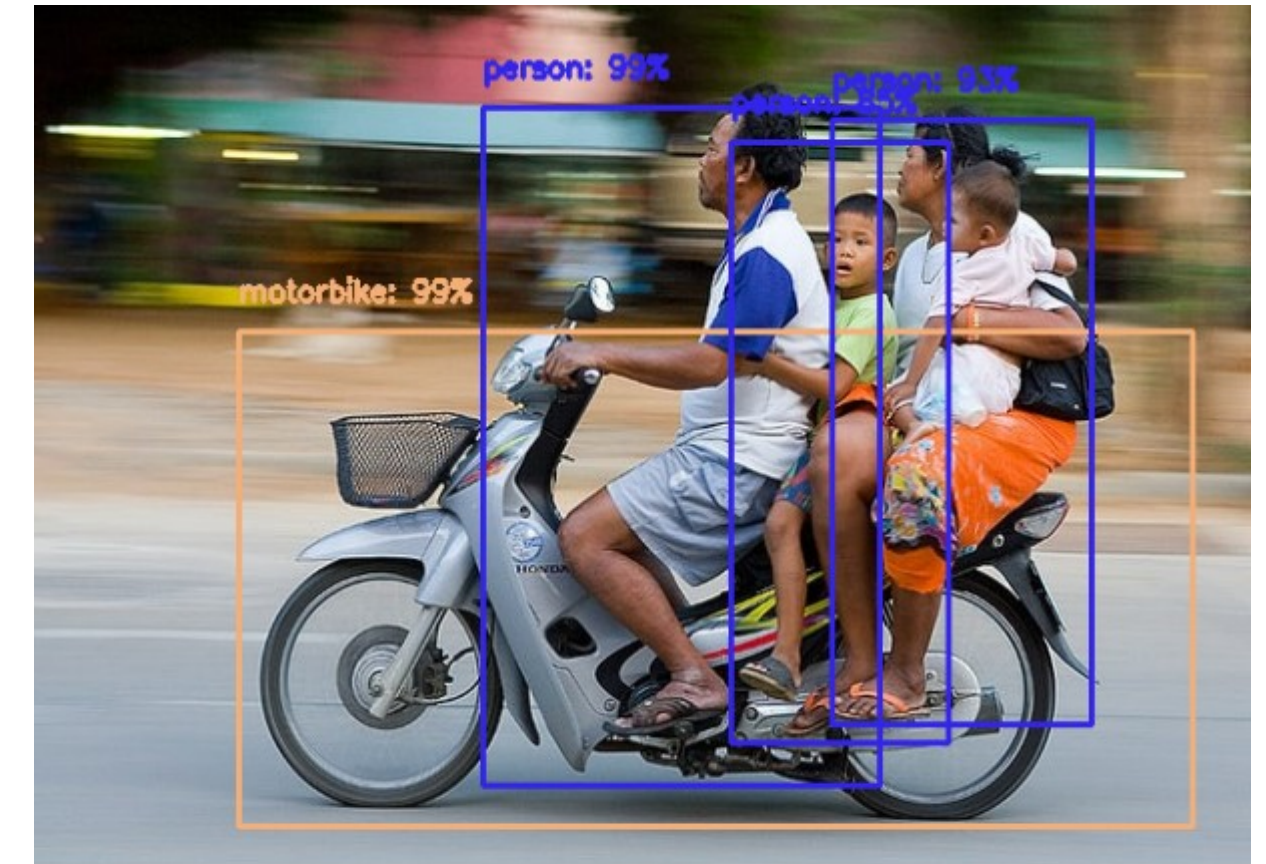
2 인 탑승 검사

```
# 총 'person' 객체의 수를 계산합니다
person_count = len(persons)

# 텍스트를 추가할 공간을 포함한 최종 이미지를 준비합니다
image_height, image_width = image.shape[:2]
space_height = 50 # 텍스트를 추가할 공간의 높이
final_image = np.ones((image_height + space_height, image_width, 3), dtype=np.uint8) * 255
final_image[:image_height, :] = image

# 2명 이상의 'person' 객체가 있을 때 메시지를 추가합니다
if person_count > 1:
    message = "More than 2 people detected"
    cv2.putText(final_image, message, (10, image_height + space_height - 10),
                cv2.FONT_HERSHEY_SIMPLEX, 1.0, (0, 0, 0), 2, cv2.LINE_AA)

return final_image
```



More than 2 people on the bike

< 출력 이미지 >

사람으로 인식된 객체의 수를 계산하여 몇명이 타고 있는지를 확인
-> 인식된 사람의 객체 중 전동 킥보드 객체와 일치하는 사람만 계산

키크보드 회사 판단

```
if boxes:
    # 'person' 박스의 지정된 영역 내에서 각 색상의 영역을 계산
    area_green = area_purple = area_yellow = 0
    for (startX, startY, endX, endY) in boxes:
        # ROI(관심 영역)를 박스의 하단 절반과 50 픽셀을 포함하는 영역으로 정의
        roi_startY = min(endY, startY + (endY - startY) // 2 + 50)
        roi = balloon_hsv[roi_startY:endY, startX:endX]

        # ROI의 마스크를 업데이트
        roi_mask_green = cv2.inRange(roi, lower_green, upper_green)
        roi_mask_purple = cv2.inRange(roi, lower_purple, upper_purple)
        roi_mask_yellow = cv2.inRange(roi, lower_yellow, upper_yellow)

        # ROI에서 각 색상의 영역을 계산
        area_green += np.sum(roi_mask_green > 0)
        area_purple += np.sum(roi_mask_purple > 0)
        area_yellow += np.sum(roi_mask_yellow > 0)

    areas = {'GCOO': area_green, 'Beam': area_purple, 'SSingSSing': area_yellow}
    dominant_color = max(areas, key=areas.get)
else:
    # 박스가 지정되지 않은 경우 전체 영역을 계산
    area_green = np.sum(mask_green > 0)
    area_purple = np.sum(mask_purple > 0)
    area_yellow = np.sum(mask_yellow > 0)
    areas = {'GCOO': area_green, 'Beam': area_purple, 'SSingSSing': area_yellow}
    dominant_color = max(areas, key=areas.get)

return result1, result2, result3, dominant_color
```



Company color: Beam

< 출력 이미지 >

키크보드 객체를 인식하여 키크보드 객체 속 색상만 추출 < 데이터셋 문제로 불가능 >

-> 키크보드에 탑승한 사람 객체의 절반 중 하단 부분에서 사람 객체 밖 50 픽셀까지 색상 추출

[싱싱]



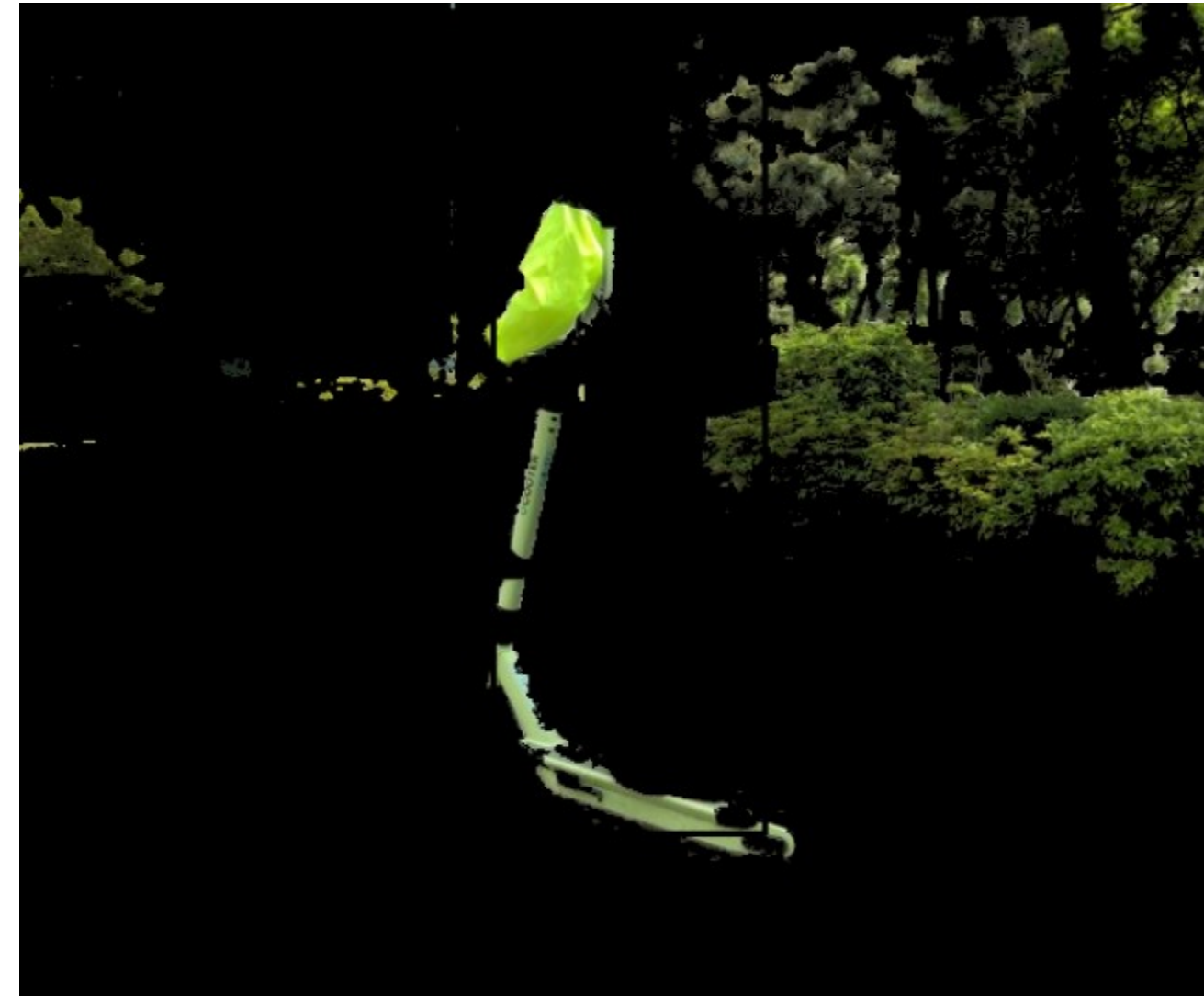
Company color: SSingSSing



[GCOO]



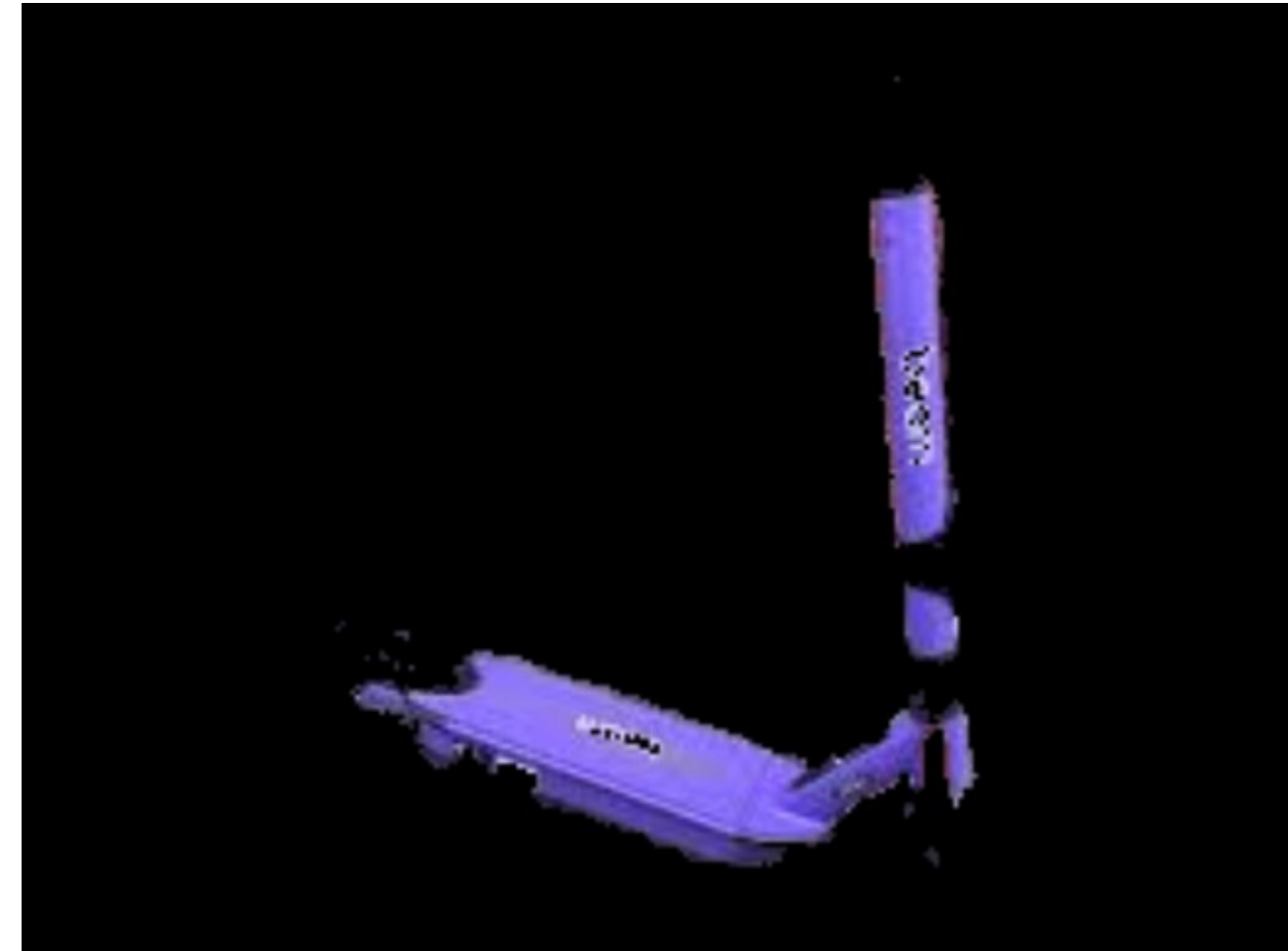
Company color: GCOO



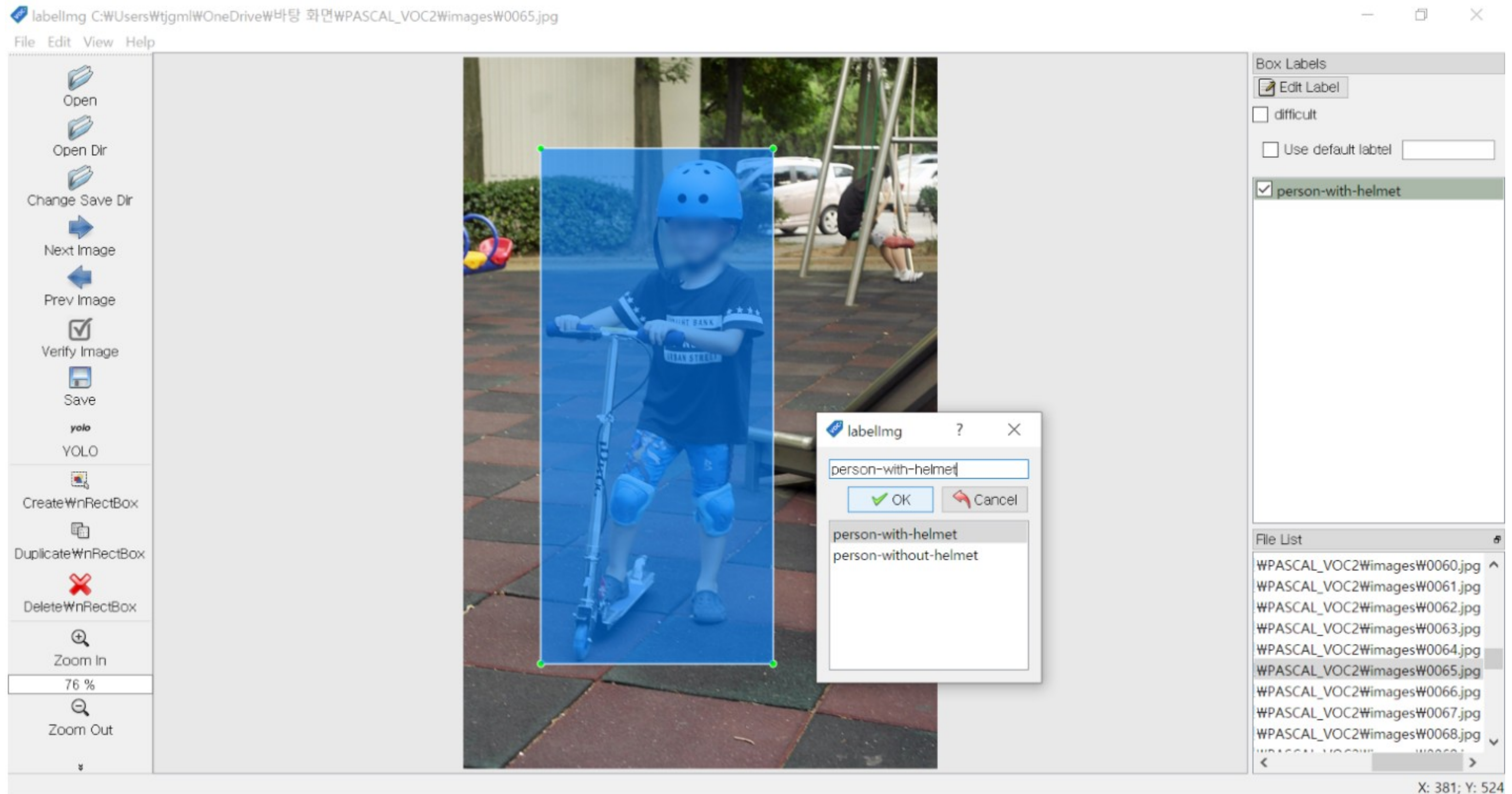
[Beam]



Company color: Beam



[헬멧 착용 여부]



[헬멧 착용 여부]

300 epochs completed in 0.689 hours.
Optimizer stripped from runs/train/exp/weights/last.pt, 14.3MB
Optimizer stripped from runs/train/exp/weights/best.pt, 14.3MB

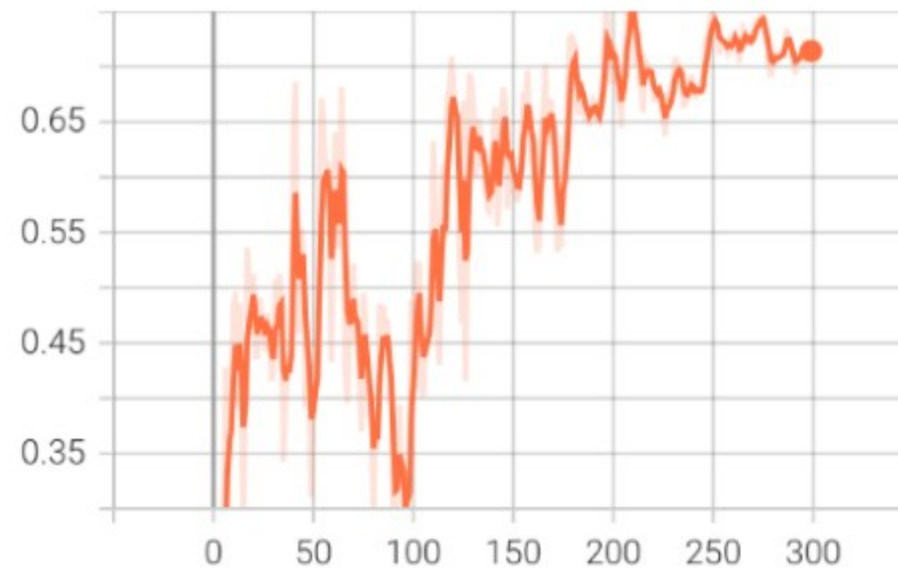
Validating runs/train/exp/weights/best.pt...
Fusing layers...

Model Summary: 213 layers, 7015519 parameters, 0 gradients, 15.8 GFLOPs

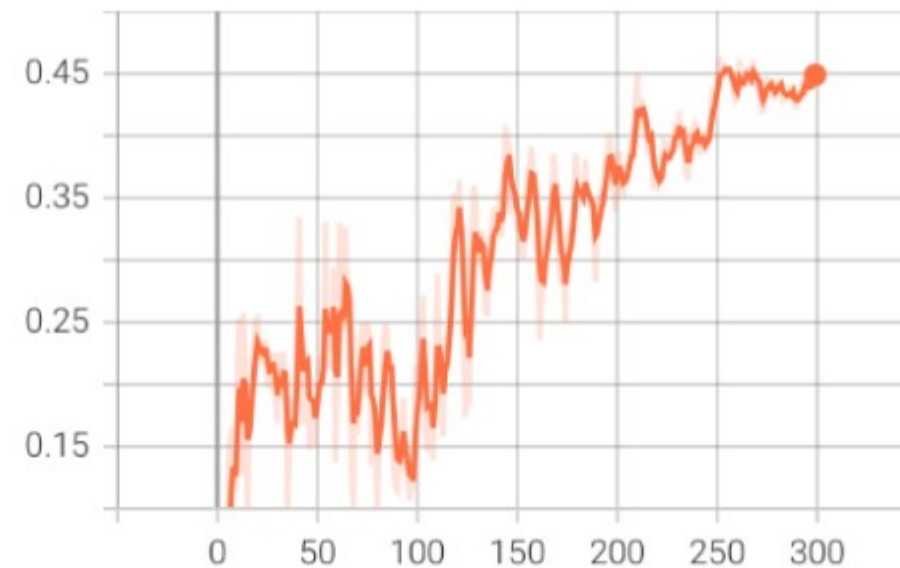
	Class	Images	Labels	P	R	mAP@.5	mAP@.5:.95: 100% 1/1 [00:00<00:00, 3.07it/s]
	all	21	31	0.609	0.789	0.746	0.464
	helmet	21	12	0.639	1	0.912	0.584
	no-helmet	21	19	0.578	0.579	0.579	0.344

Results saved to runs/train/exp

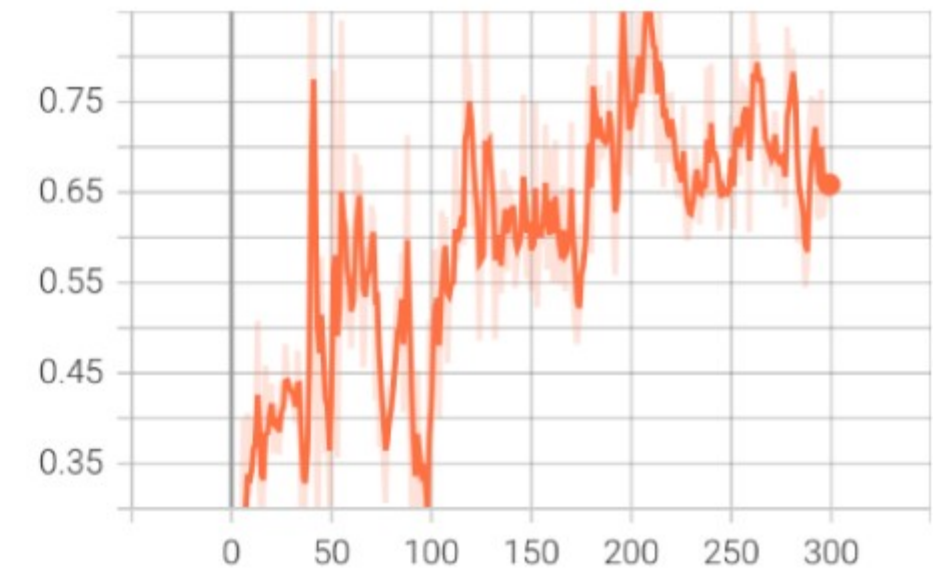
metrics/mAP_0.5
tag: metrics/mAP_0.5



metrics/mAP_0.5:0.95
tag: metrics/mAP_0.5:0.95



metrics/precision
tag: metrics/precision



[헬멧 인식]



[결론]



- 사람 인식 & 회사 구분 코드와 헬멧 인식 코드 결합
- 사진에서 전동 킥보드에 탑승한 사람이 2인 이상인지만 구분하면 되지만 모든 사람에 대해 인식



1. 헬멧 인식 코드를 공부하여 자체적으로 개발
2. 두 개로 나뉜 코드를 통합
3. 해당 사진 속 정확률 체크 여부

[AI 영상인식 특강]

THANK YOU



문 효 원 | 박 유 진 | 이 호 현