YOLOv8을 이용한 운전자 부주의 객체 검출

Streamlit으로 훈련시킨 YOLOv8 모델 적용

TEAM2 – *NewBees*
함은규
송은민
윤성진
조서현

CONTENTS

01 주제 선정

02 데이터 탐색 및 이미지 전처리

03 모델링

04 학습결과

05 웹 서비스 구현 (*Streamlit*)

06 개선 사항

CONTENTS

01 주제 선정

02 데이터 탐색 및 이미지 전처리

03 모델링

04 학습결과

05 웹 서비스 구현 (*Streamlit*)

06 개선 사항

개발 환경

PyTorch

PyTorch

2.2.0

Visual Studio Code

Streamlit

Streamlit

1.31.0

OpenCV

Open CV

4.9.0

Python

3.8.18

Kubeflow

ultralytics YOLOv8

Ultralytics YOLOv8.1.9 🚀 Python-3.8.10 torch-1.8.1+cu111 CUDA:0 (NVIDIA GeForce RTX 3060, 12044MiB)

1. 주제 선정

NEWS

- *최근 교통사고에서 사망 사고 원인별로 조사 결과, 졸음 및 주시 태만이 67.6%로 가장 높은 것으로 기록*
  https://thepoc.co.kr/58/?q=YToxOntzOjEyOiJrZXl3b3JkX3R5cGUiO3M6MzoiYWxsIjt9&bmode=view&idx=7008773&t=board

- *AI 및 센서 기술을 통하여 해당 운전자의 위험 행동을 감지하여 주의를 주는 서비스들이 개발되고 있음*
  https://www.sedaily.com/NewsView/29RZKXMF51

운전자_얼굴 # 안전 # 얼굴_객체탐지 # 운전자_부주의

1. 주제 선정

국내·외 카메라 기반 운전자 상태 모니터링 시스템

모니터링 장치	개발회사	작동원리
패신저 아이 (Passenger Eye)	일본 (덴소)	대시보드 계기판상단에 부착된 카메라로 눈꺼풀 변화 정도, 얼굴 끄덕임, 차량의 차로이탈 인식해 졸음운전 인식 및 경고
3D 페이스 트래커 (Face Tracker)	일본 (도요타)	계기판 전방에 설치된 카메라로 운전자 얼굴의 3차원 모델 변화와 얼굴형태 인식으로 운전자의 상태를 인식
어텐션 어시스터 (Attention Assist)	독일 (벤츠)	주행 중 운전시간과 운전대 조향정도 등으로 운전 피로도 예측 및 경고
운전자 주의 모니터링 장치 (Driver Attention Monitoring System)	미국 (씨잉머신)	핸들에 부착된 카메라로 눈동자 주시 패턴을 분석하여 피로도와 주의력 산만 상태 인식 및 경고
졸음운전 감지 및 경보시스템	디나로그	적외선 LED 및 카메라 영상기반 눈동자 움직임, 얼굴 위치 인식 및 경고
졸음운전 경고시스템	현대 모비스	차량 내부에 장착한 적외선 카메라로 운전자의 눈동자 움직임과 안면 근육의 변화를 통해 운전자의 상태를 파악

2. 데이터 탐색 및 이미지 전처리

- 데이터 탐색 -

- 7종의 운전자 얼굴 객체를 탐지 및 분류
  → 얼굴, 눈, 입, 담배, 핸드폰 등..
- Input
  → 운전자 얼굴 이미지 (800*1280)
- Output
  → 객체의 bounding box 좌표 및 클래스 (labeling)
- Class 종류
  → Face, Eye-opened, Eye-closed, Mouth-opened, Mouth-closed, Cigar, Phone

종류:	파일 폴더
위치:	C:₩Users₩user₩Desktop₩cvproject₩운전 사고 예방을
크기:	15.0GB (16,160,623,769 바이트)
디스크 할당 크기:	15.5GB (16,719,069,184 바이트)
내용:	파일 272,979, 폴더 0

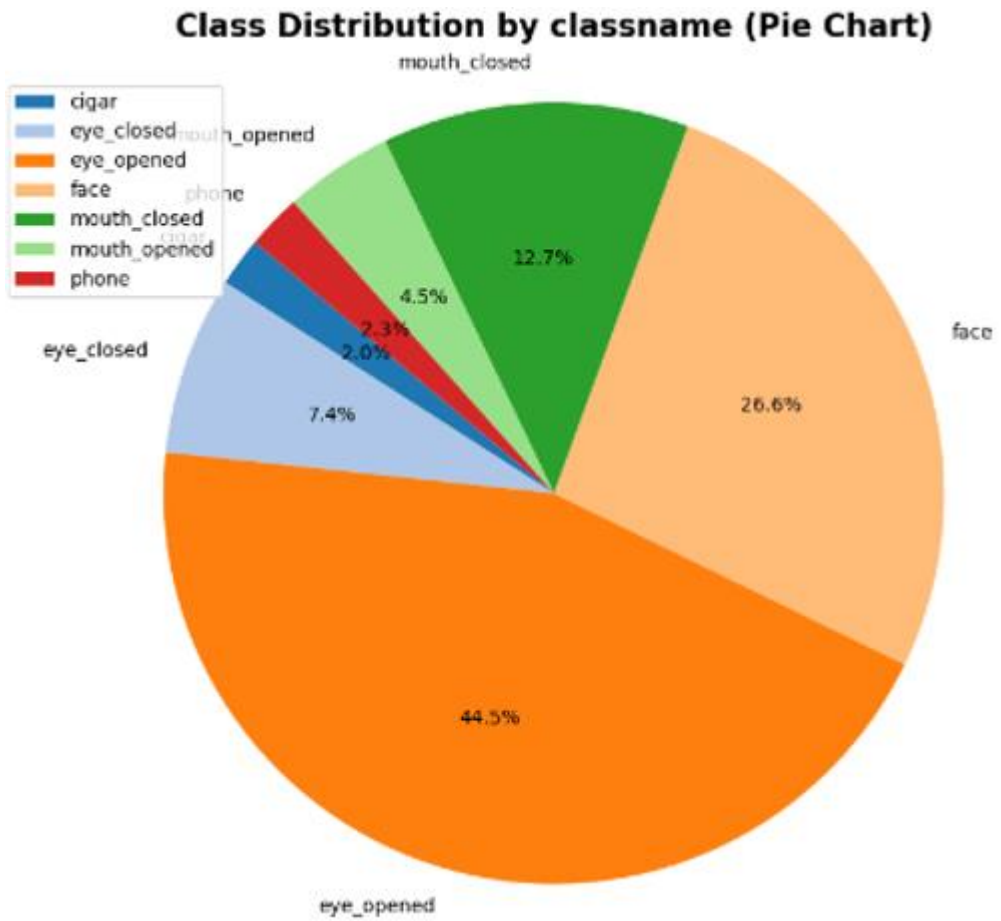
Dataset

AIHub "운전 사고 예방을 위한 운전자 부주의 행동 검출 모델"
https://aihub.or.kr/aihubdata/data/view.do?currMenu=120&topMenu=100&aihubDataSe=extrldata&dataSetSn=448

2. 데이터 탐색 및 이미지 전처리

- 이미지 전처리 -

Train data before

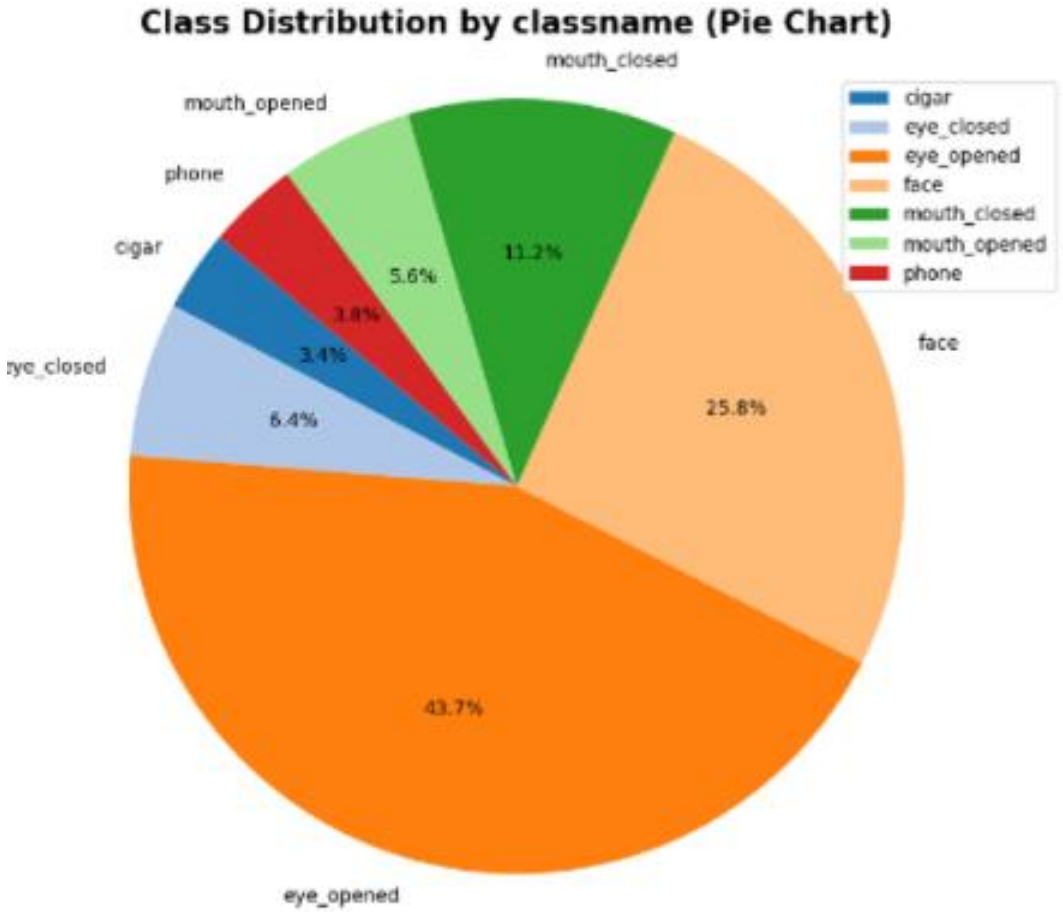


- cigar와 phone이 다른 class보다 너무 적음
- Data imbalance 생김
  → 모델 훈련 시키면서 empty Tensor로 error 발생
- cigar와 phone을 제외한 이미지만 랜덤으로 ½ 삭제

2. 데이터 탐색 및 이미지 전처리

- 이미지 전처리 -

Train data After

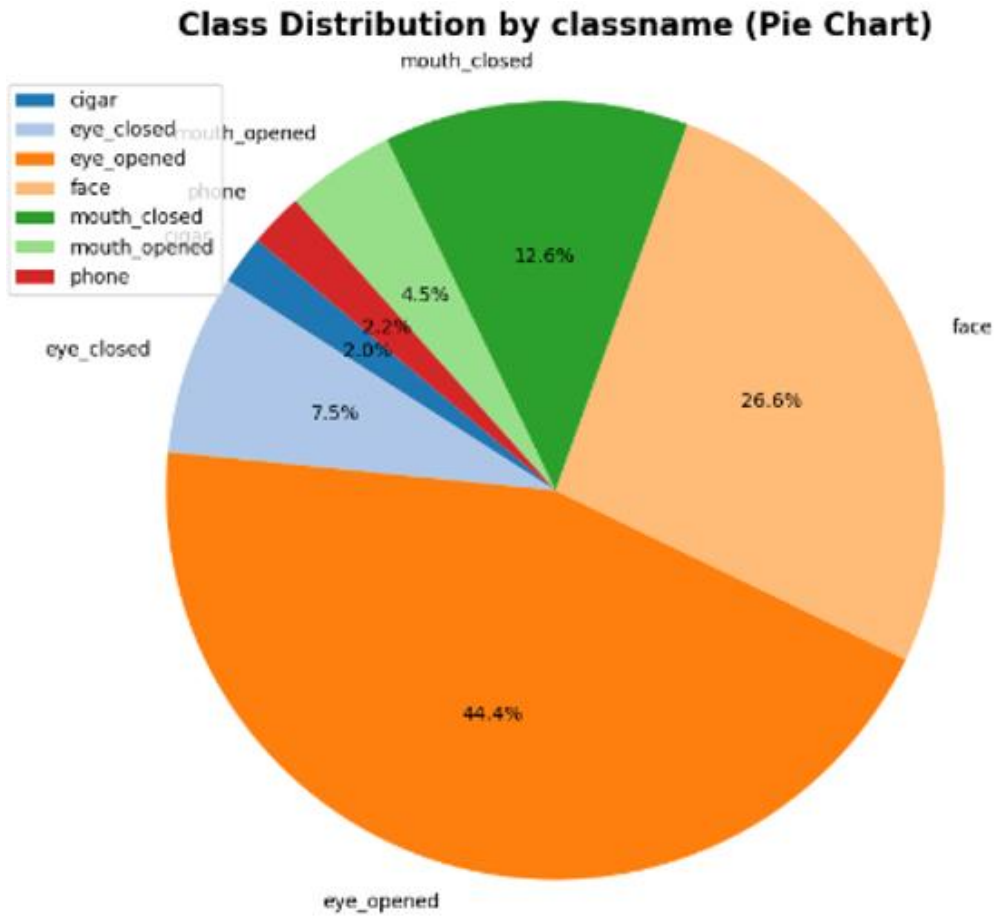



```python
# CSV 파일을 읽어들임
data = pd.read_csv("img_train.csv")

# imagename으로 그룹화하여 클래스 5 또는 6이 없는 이미지 찾기
grouped_data = data.groupby('imagename')
images_to_keep = []

for name, group in grouped_data:
    # 이미지 그룹에서 클래스 5 또는 6이 있는지 확인
    has_5_or_6 = any((group['class'] == 5) | (group['class'] == 6))
    if not has_5_or_6:
        images_to_keep.append(name)

# 이미지 중에서 랜덤으로 절반 선택하여 삭제
random.shuffle(images_to_keep)
images_to_delete = images_to_keep[:len(images_to_keep)//2]

# 삭제할 이미지들을 데이터프레임에서 제거
filtered_data = data[~data['imagename'].isin(images_to_delete)]
```

2. 데이터 탐색 및 이미지 전처리

- 이미지 전처리 -

Validation data before

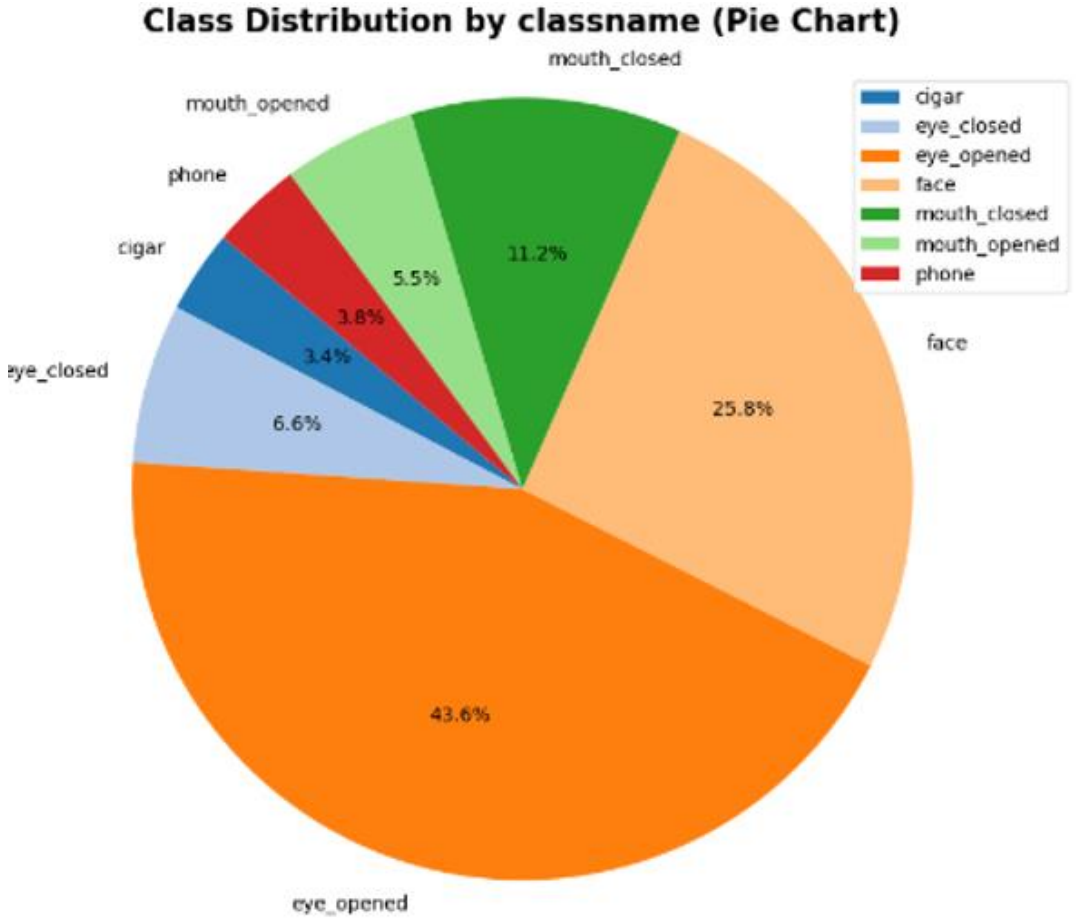


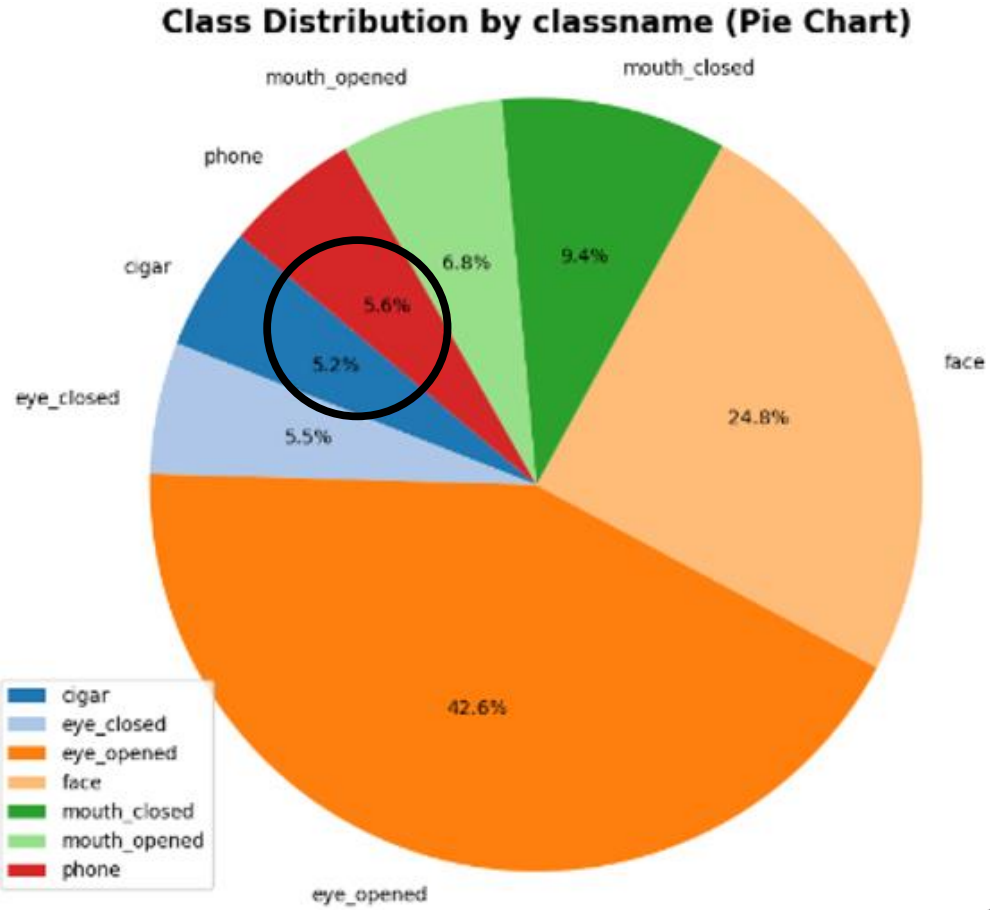
- cigar와 phone이 다른 class보다 너무 적음
- Data imbalance 생김
  → 모델 훈련 시키면서 empty Tensor로 error 발생
- cigar와 phone을 제외한 이미지만 랜덤으로 ½ 삭제
- 랜덤으로 ½ 삭제 한 번 더

2. 데이터 탐색 및 이미지 전처리

- 이미지 전처리 -

Validation data After



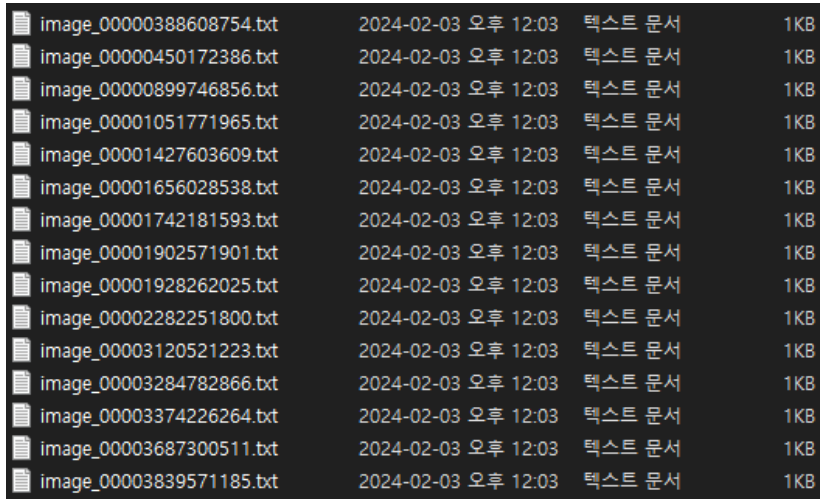


2. 데이터 탐색 및 이미지 전처리

Json → txt

```
import json

# json파일 loading
with open("labels.json", "r") as jfile:
    jdata = json.load(jfile)
Key_1st = jdata.keys()
Key_1st = str(list(Key_1st)[0])
jadata_list = jdata[Key_1st]
jadata_list
```

```
objects_clases = []
classes_dict = {'face': 0, 'eye_opened':1, 'eye_closed': 2,
                'mouth_closed': 3, 'mouth_opened': 4, 'cigar': 5, 'phone': 6}

for ind1 in range(len(jadata_list)):
# for ind1 in range(0,3): # test용
    image_id = jadata_list[ind1]["image_id"]
    file_name = jadata_list[ind1]["file_name"][:-5]
    objects = jadata_list[ind1]["objects"]
    print(f"\nimage_id: {image_id},\nfile_name: {file_name},\nobjects:")

    object_classname_list =[]
    object_position_list =[]

    for ind2 in range(len(objects)):
        # object_id = objects[ind2]["object_id"]
        object_class = objects[ind2]["class"] # 바운딩박스: 클래스 이름
        objects_clases.append(object_class)
        object_position = objects[ind2]["position"]  # 바운딩박스: xyxy좌표

        object_classname_list.append(object_class)
        object_position_list.append(object_position)

        # print(f"진행확인용 {ind1}, {ind2}")
        print(f"id: {object_class}, xyxy: {object_position}")

    with open(f"train_txt\{file_name}.txt", "w") as f:
        for inx in range(len(object_classname_list)):
            classes_key = object_classname_list[inx]
            class_num = classes_dict[classes_key]
            f.write(f"{class_num}
                    {object_position_list[inx][0]}
                    {object_position_list[inx][1]}
                    {object_position_list[inx][2]}
                    {object_position_list[inx][3]}\n")
```



2. 데이터 탐색 및 이미지 전처리

Image & Label Copy

```python
import pandas as pd
import os
import shutil

# Image Copy
def image_copy(data, column, input_dir, output_dir):
    # CSV 파일을 읽어들임
    data = pd.read_csv(data)
    print("Read CSV File")

    # 이미지 파일이 있는 디렉토리
    input_dir = input_dir
    print("input_dir")
    # 저장할 디렉토리
    output_dir = output_dir
    print("output_dir")

    # 이미지 파일 이름 가져오기
    image_files = os.listdir(input_dir)
    print("-- image_files --")

    cnt = 0
    # 이미지 파일 이름과 ['imagename'].unique()에 있는 이미지 이름 비교 후, 겹치는 것만 선택하여 저장
    for imagename in data[column].unique():
        cnt += 1
        if f"{imagename}.jpg" in image_files:
            source_path = os.path.join(input_dir, f"{imagename}.jpg")
            dest_path = os.path.join(output_dir, f"{imagename}.jpg")
            shutil.copyfile(source_path, dest_path)
            print(f"Image Copy, {cnt}")
```

Image와 label 일치하는지 확인

```
파일 'image_00000899746856.jpg'와 'image_00000899746856.txt'의 이름이 일치합니다.
파일 'image_00001656028538.jpg'와 'image_00001656028538.txt'의 이름이 일치합니다.
파일 'image_00001928262025.jpg'와 'image_00001928262025.txt'의 이름이 일치합니다.
파일 'image_00004027738437.jpg'와 'image_00004027738437.txt'의 이름이 일치합니다.
파일 'image_00004886775796.jpg'와 'image_00004886775796.txt'의 이름이 일치합니다.
파일 'image_00007153948965.jpg'와 'image_00007153948965.txt'의 이름이 일치합니다.
파일 'image_00007169839388.jpg'와 'image_00007169839388.txt'의 이름이 일치합니다.
파일 'image_00011125802937.jpg'와 'image_00011125802937.txt'의 이름이 일치합니다.
파일 'image_00013744059628.jpg'와 'image_00013744059628.txt'의 이름이 일치합니다.
파일 'image_00014257303739.jpg'와 'image_00014257303739.txt'의 이름이 일치합니다.
파일 'image_00014673091407.jpg'와 'image_00014673091407.txt'의 이름이 일치합니다.
파일 'image_00018153933337.jpg'와 'image_00018153933337.txt'의 이름이 일치합니다.
파일 'image_00018584174976.jpg'와 'image_00018584174976.txt'의 이름이 일치합니다.
파일 'image_00018990181056.jpg'와 'image_00018990181056.txt'의 이름이 일치합니다.
파일 'image_00022799478235.jpg'와 'image_00022799478235.txt'의 이름이 일치합니다.
파일 'image_00023236367506.jpg'와 'image_00023236367506.txt'의 이름이 일치합니다.
파일 'image_00023259095790.jpg'와 'image_00023259095790.txt'의 이름이 일치합니다.
파일 'image_00023732354223.jpg'와 'image_00023732354223.txt'의 이름이 일치합니다.
파일 'image_00024063648384.jpg'와 'image_00024063648384.txt'의 이름이 일치합니다.
파일 'image_00025376428909.jpg'와 'image_00025376428909.txt'의 이름이 일치합니다.
```

모듈화 `%%writefile ./cvproject/copy_directory.py`

3. 모델링

- YOLOv8 -

Model	Size (pixels)	mAP^{val} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

▲ YOLOv8의 예측 속도

- 위 표와 같이 YOLOv8의 속도가 빠르다는 것과 성능 또한 다른 모델에 비해 우수한 것을 확인 가능

3. 모델링

- YOLOv8 훈련-

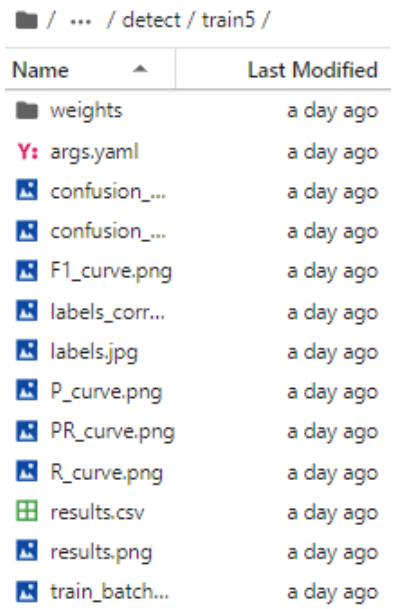
Install

```
!pip install ultralytics
```

Model load train

```
from ultralytics import YOLO
import torch
import os

model = YOLO("yolov8n.pt")
results = model.train(data=os.path.join('data.yaml'),
                      epochs= 20, batch=51, optimizer='AdamW')
```

Model save result



작업 경로에 runs 파일에 자동적으로 결과 저장

3. 모델링

```yaml
path: ../dataset_v2
train: ./images/train
val: ./images/valid

# Classes
names:
    0: face
    1: eye_opened
    2: eye_closed
    3: mouth_closed
    4: mouth_opened
    5: cigar
    6: phone
```

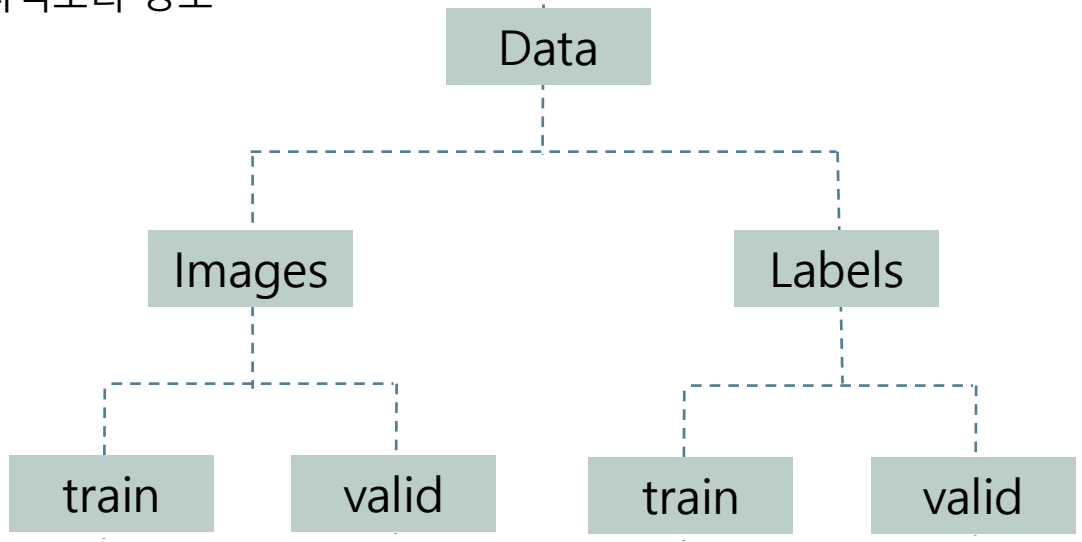
Dataset이 저장된 디렉토리 경로

훈련 데이터셋이 있는 서브 디렉토리 경로

검증 데이터셋이 있는 서브 디렉토리 경로

클래스 레이블과 해당 클래스 설명 이름을 매핑한 사전

▲ Yaml File

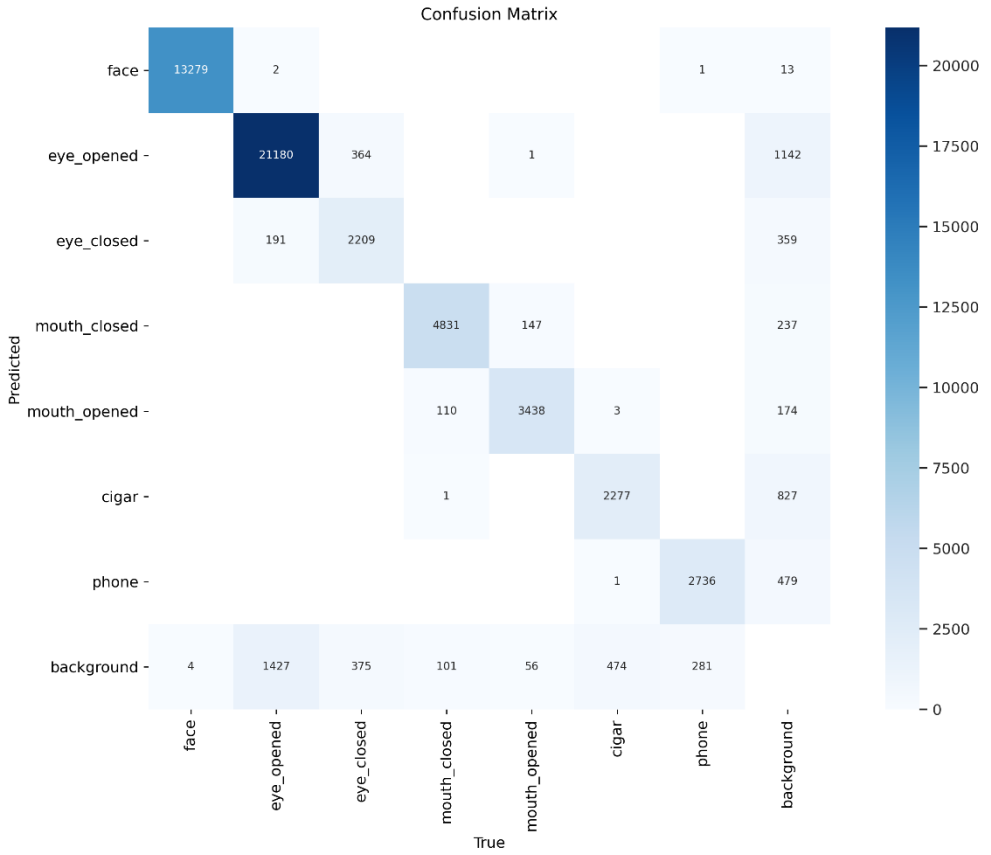


▲ Tree 구조

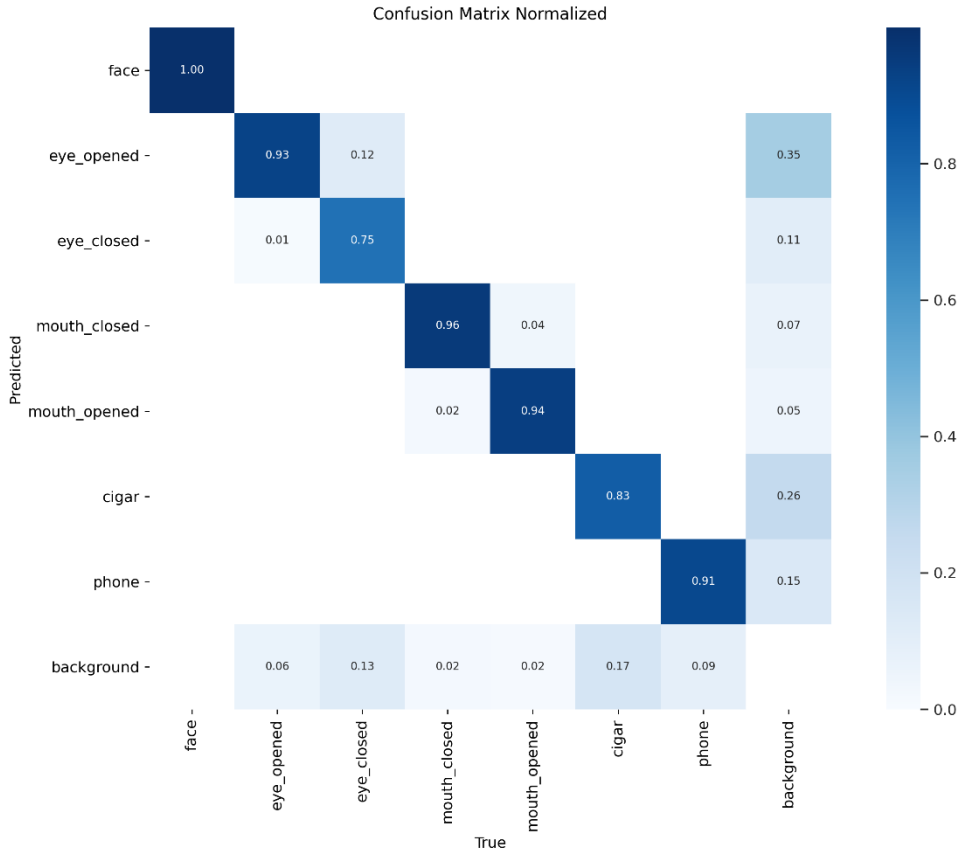
본 데이터셋은 주로 이미지 기반의 다중 클래스 분류 작업을 위해 사용

4. 학습 결과

- Confusion Matrix -



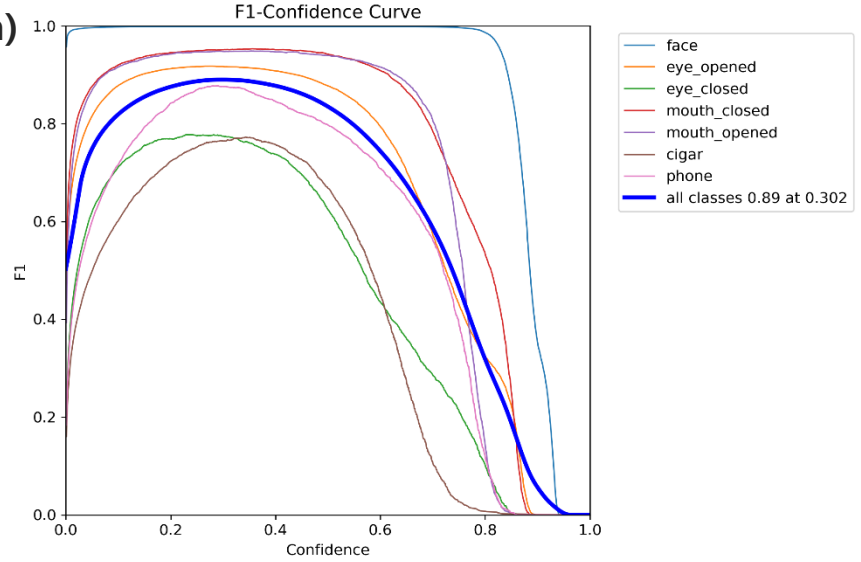
(a)



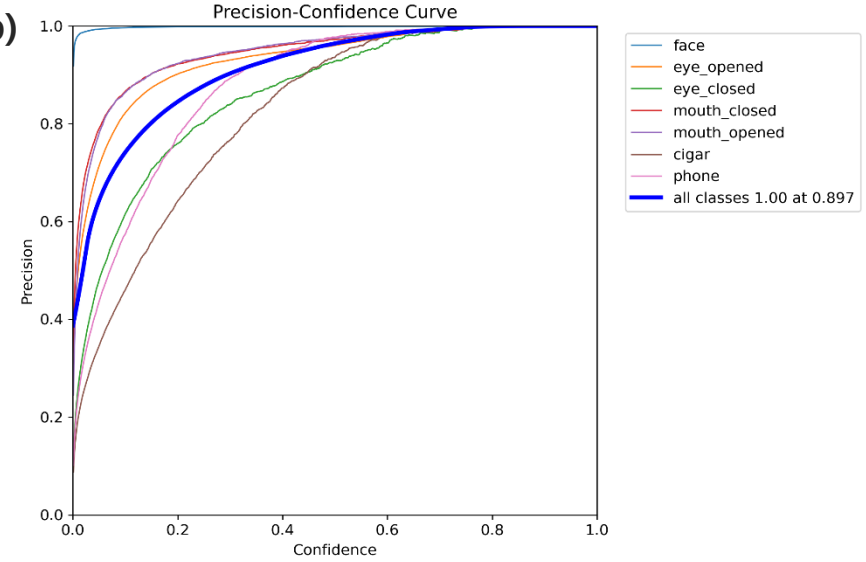
(b)

4. 학습 결과

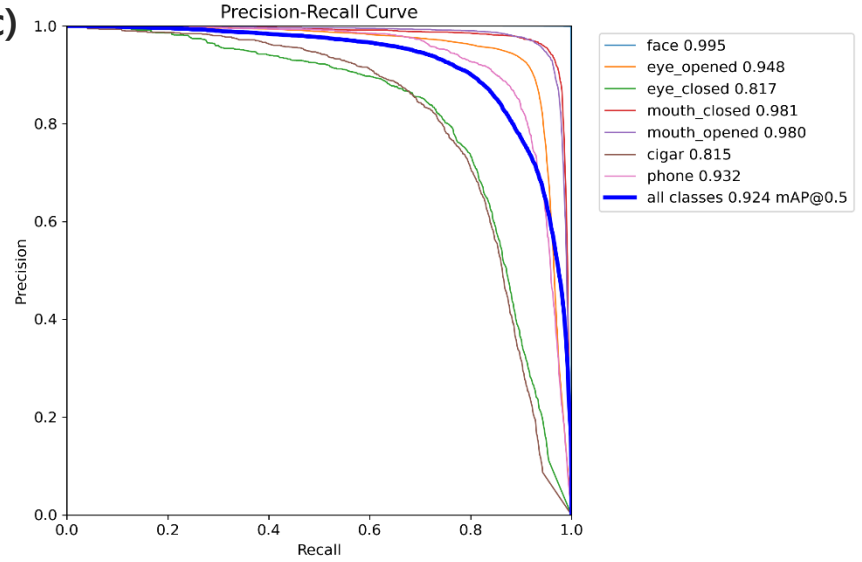
(a)


F1-Confidence Curve
F1
Confidence
face
eye_opened
eye_closed
mouth_closed
mouth_opened
cigar
phone
all classes 0.89 at 0.302

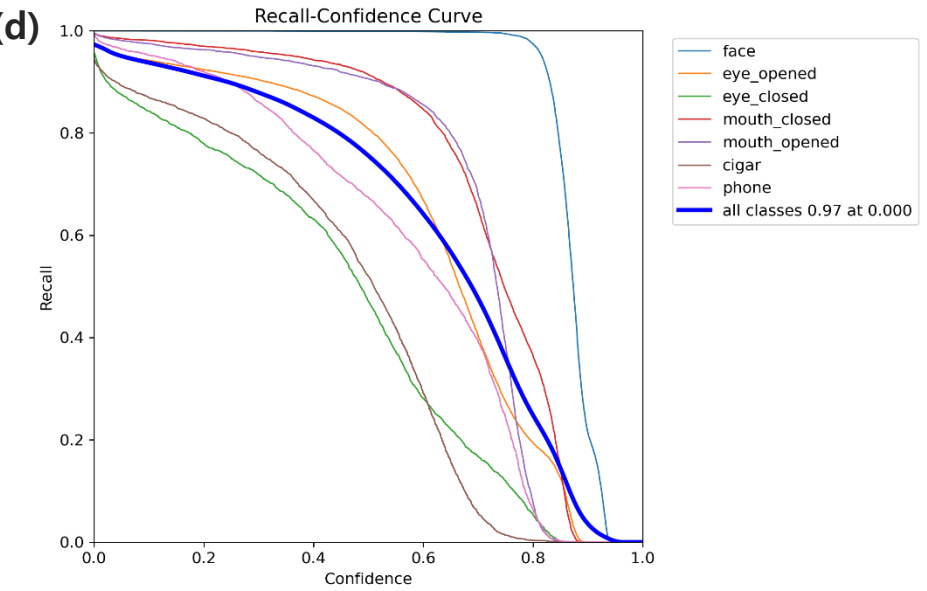
(b)


Precision-Confidence Curve
Precision
Confidence
face
eye_opened
eye_closed
mouth_closed
mouth_opened
cigar
phone
all classes 1.00 at 0.897

(c)

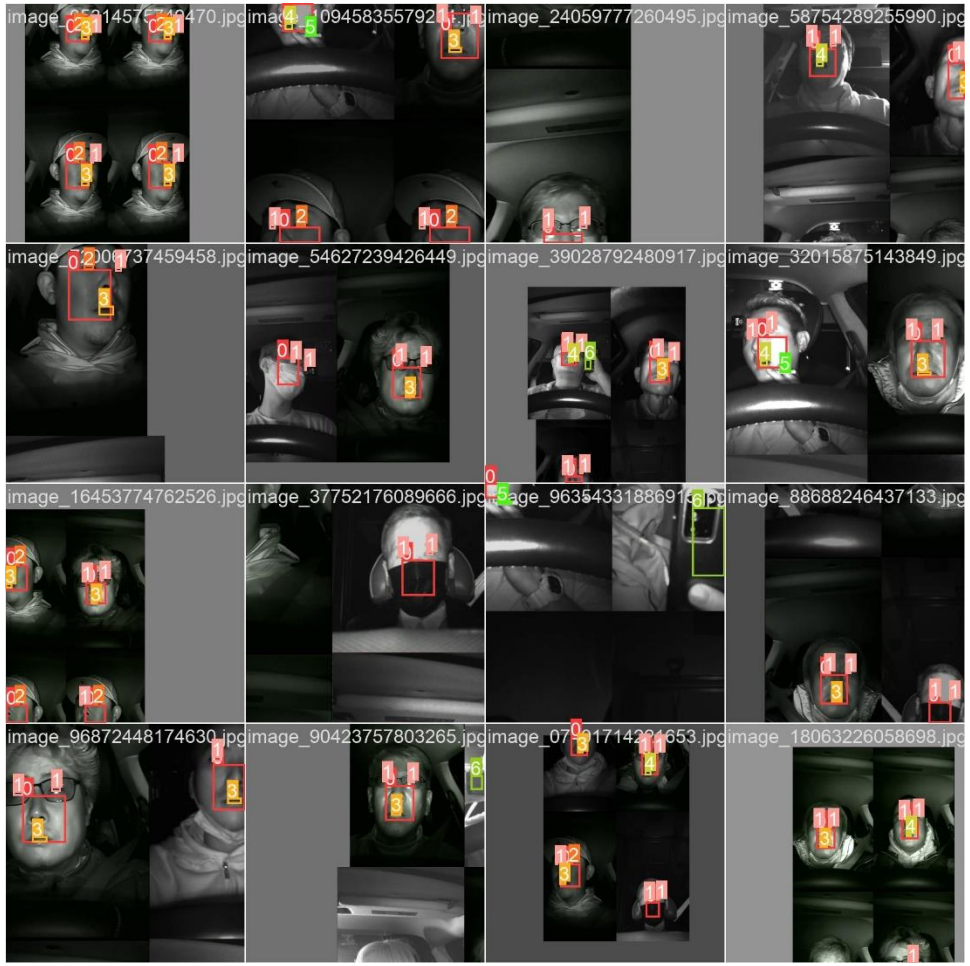

Precision-Recall Curve
Precision
Recall
face 0.995
eye_opened 0.948
eye_closed 0.817
mouth_closed 0.981
mouth_opened 0.980
cigar 0.815
phone 0.932
all classes 0.924 mAP@0.5

(d)

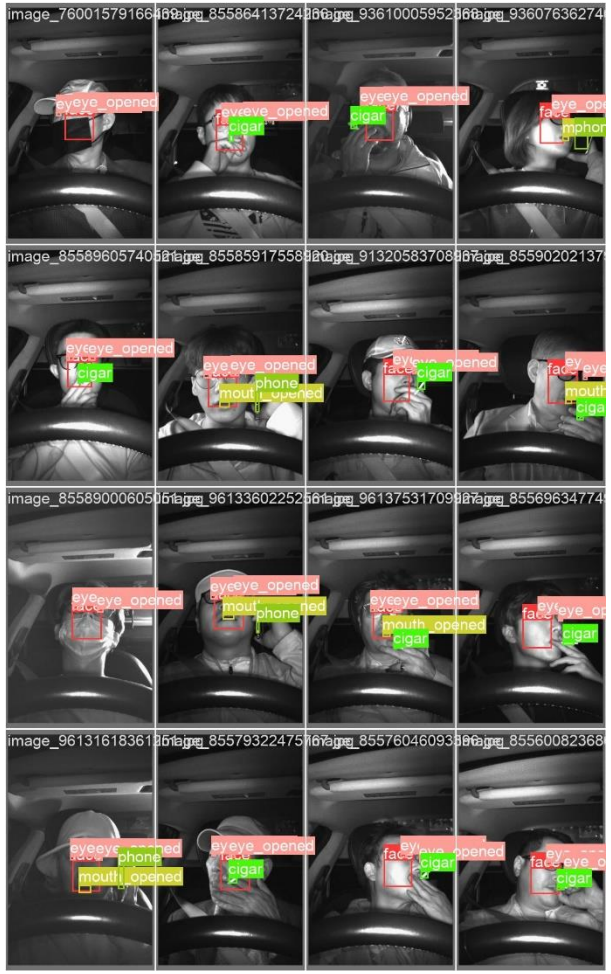

Recall-Confidence Curve
Recall
Confidence
face
eye_opened
eye_closed
mouth_closed
mouth_opened
cigar
phone
all classes 0.97 at 0.000

4. 학습 결과

- 예시 사진 -



Train

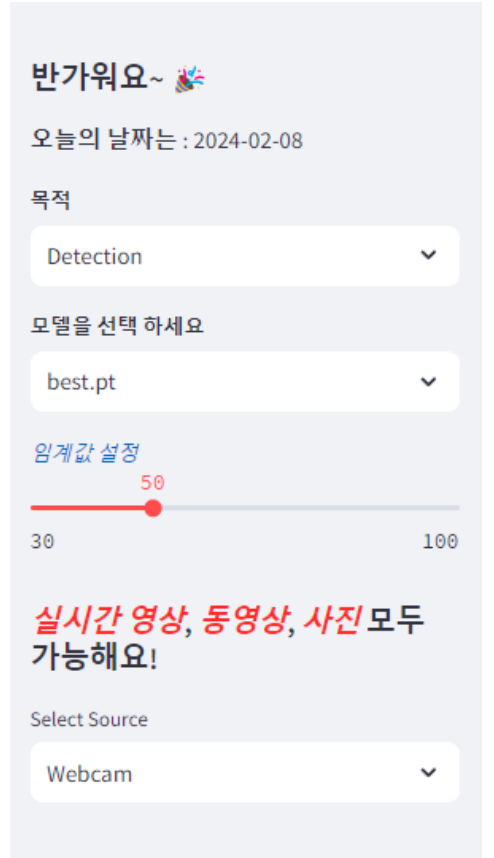


validation

5. 웹 서비스 구현 *streamlit*

회원 정보 확인 후 이동

파일의 형식 선택



5. 웹 서비스 구현 *streamlit*

운전자 행동 분석

운전자의 행동을 자사의 *안전 메뉴얼*과 대조합니다.

✅ 안전한 운전을 하는 당신이 아름답습니다.

- 실시간으로 객체 탐지 결과를 표현 → But 물체의 감지 성능이 아쉬움
- 실시간 영상의 결과를 YOLO의 save_txt로 저장

6. 개선 사항

01 **아두이노, 라즈베리파이를 사용해 제품화**

➡ Socar 등 렌트카 회사 등

02 **Class 수가 적었던 이미지 데이터를 증가시켜 더 좋은 성능 유도**

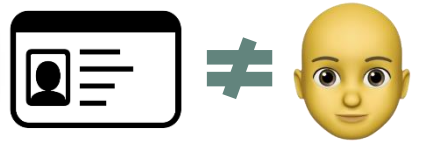
➡ 운전자의 핸드폰, 담배 사용의 이미지 데이터를 추가

➡ Object Cut-Mix 기법을 사용하여 해당 객체의 경계 상자를 자른 후 다른 이미지에 삽입하는 방식으로 데이터를 확장하면 모델의 일반화 성능을 향상시키고 더 좋은 성능 유도 할 것으로 보임

03 **배포하기**

➡ 배포할 계획

04 **운전자 안면 인식 기술을 추가하여 얼굴을 비교했을 때 다르면 시동이 켜지지 않게 하기**



➡ 얼굴의 특징 점을 이용해 자동 신원 조회 기술 구현

➡ 부주의한 행동에 운전자에게 직접적인 영향을 줄 수 있는 기술 구현

Reference

- 이민혜, 강선경, 신성윤, 임순자, "안면인식 기술을 활용한 차량 시동 제어 시스템" 원광대학교, 군산대학교, 한국정보통신학회, p425-426, 2021
- 오예진, "교통사고 사망 3명중 2명은 '졸음운전·전방주시 태만'이 원인", 2021.01.29, https://thepoc.co.kr/58/?q=YToxOntzOjEyOiJrZXl3b3JkX3R5cGUiO3M6MzoiYWxsIjt9&bmode=view&idx=7008773&t=board
- 김윤수, ""쉬어가세요" · · ·운전중 하품하면 AI가 경고", 2023.07.03, https://www.sedaily.com/NewsView/29RZKXMF51
- Ultralytics YOLOv8 문서, "YOLOv8", https://docs.ultralytics.com/ko, 2024.02.17
- 조재익, 이성주, 정호기, 박강령, 김재희 "Vision-based method for detecting driver drowsiness and distraction in driver monitoring system" SPIE, 2011

Q & A

Thank you for your listening

[Team GitHub] https://github.com/songeunmin/New_Bees