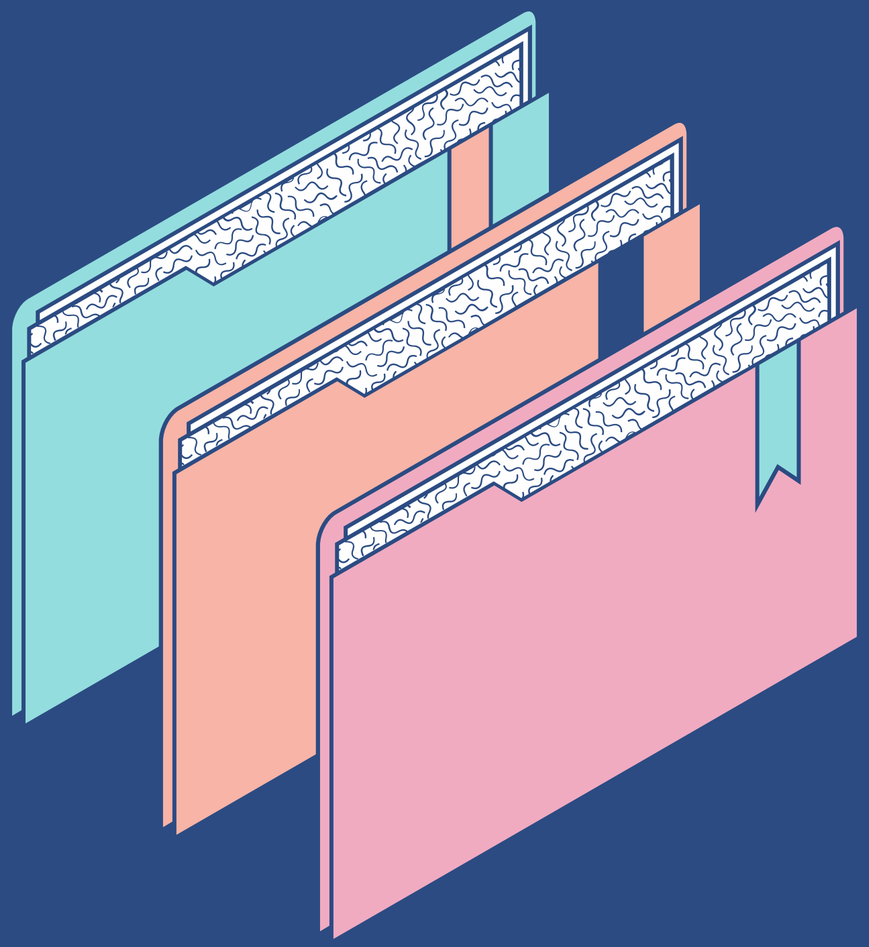
AI_16_강한나라

과적차량 단속 AI Detection

Code States Project 1

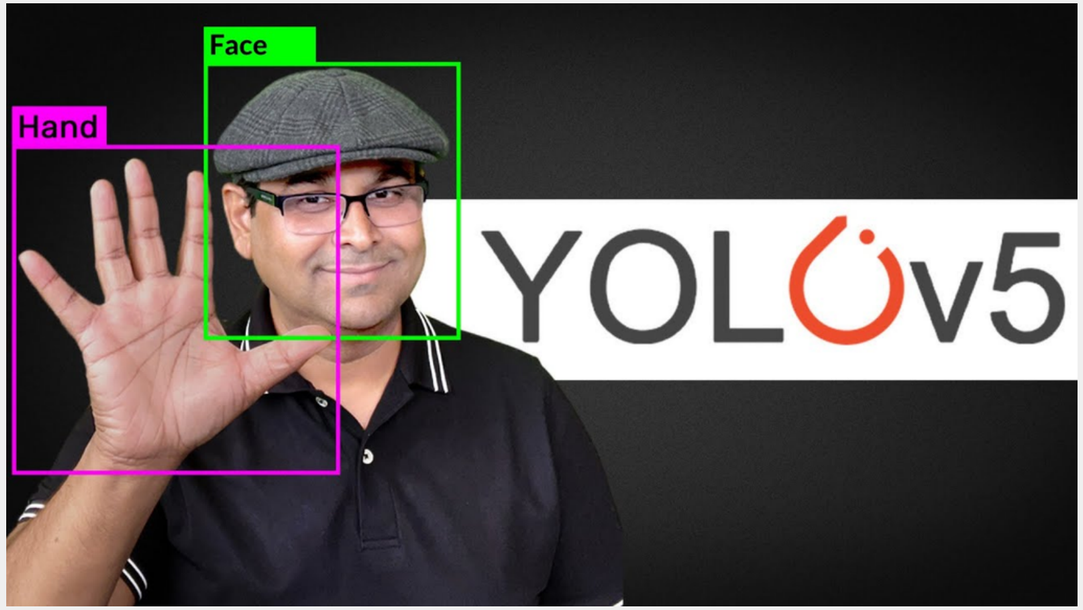


목차

이번 프레젠테이션에서 논의할 주요 주제

프로젝트 배경

- M 기업은 주차 공간 검색 서비스, 용의 차량 추적 보안 서비스 등 차량 기반 서비스를 제공하는 기업입니다.
- 이번에 M 기업은 과적 차량을 인식하여 단속하는 서비스를 개발하는 프로젝트를 새롭게 진행하려합니다.
- 해당 프로젝트의 베이스 모델로 주어진 영상 내 주행 차량을 인식하여 박스로 표시하는 모델을 제작해야합니다.



YOLOv5 특징

YOLOv5는 모델 탐지의 정확도가 높고, 초당 최대 140 프레임으로 연산속도가 매우 빠르다.

Yolov5는 모델의 가중치 파일 크기도 Yolov4 보다 작아서 실시간 탐지를 구현하기에 더 적합하다.

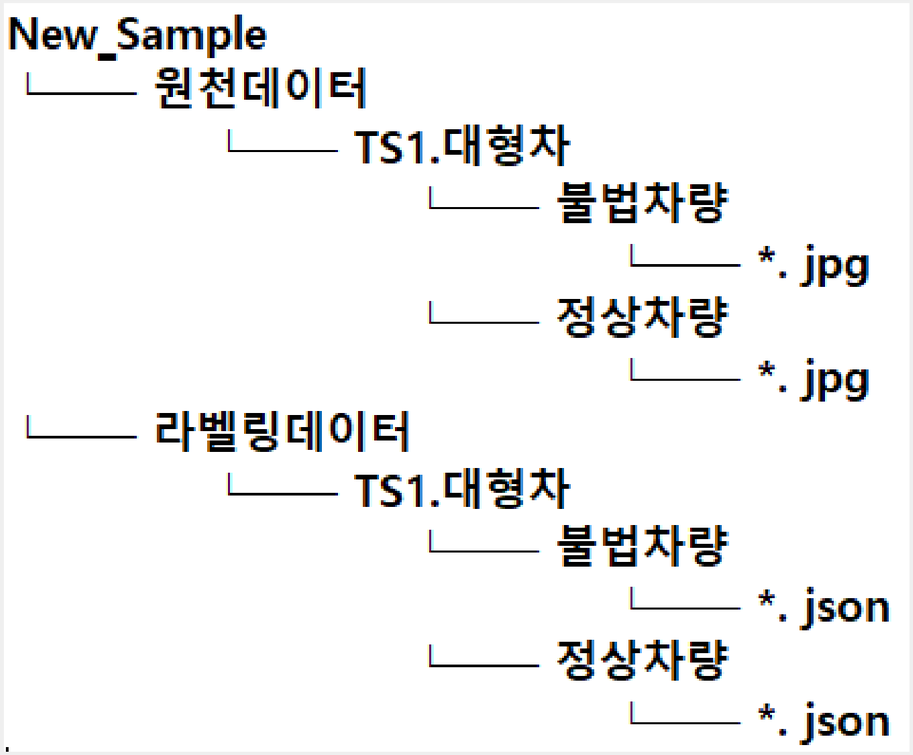
데이터

- 데이터과적 차량(170장) 및 정상 차량(170장) 이미지 및 라벨링 데이터(json)
- 데이터 링크: https://drive.google.com/file/d/1VR6eF8tusqG-tb33kN3GzbHacs6Wj-UQ/view?usp=sharing

사용가능 라이브러리 및 모델

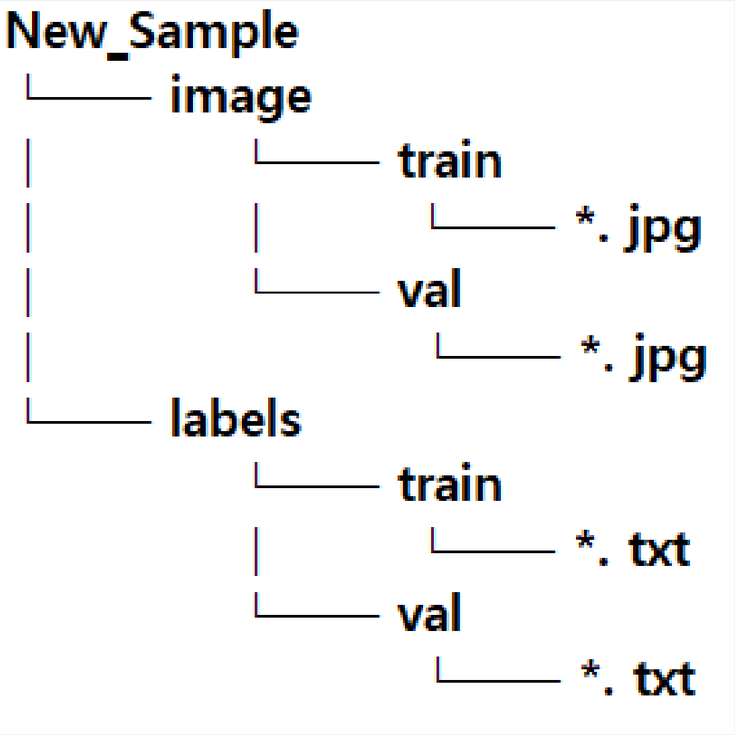
- 모델 라이브러리: yolov5
- OpenCV
- 주어진 모델 라이브러리를 제외한 다른 모델을 사용할 수 없습니다.
- 기본적인 라이브러리(numpy, pandas, glob 등)는 자유롭게 사용할 수 있습니다.

주어진 sample 데이터셋 구조



YOLOv5 를 쓰려면
이렇게 변경해야함

따라서 데이터를 합치고
json에 있는 데이터를
추출해서 txt로 변경 후
train, val로 나누어준다.



데이터 전처리와 모델구성

- json 파일 --> txt로 변경 (클래스매핑, 바운딩 박스)
- yaml 파일 생성 및 변경
- yolov5 훈련 파라미터 자동설정
- 과적170+일반170 총 340개의 데이터로 훈련
- yolov5s와 50epoch 설정

학습 사진

```
classes = {'불법차량': 0, '정상차량': 1}
```

```
# 바운딩 박스 좌표값
dw = 1. / img_width
dh = 1. / img_height

x = x1 + x2 / 2.0
y = y1 + y2 / 2.0
w = x2
h = y2
```

```
names:
- 불법차량
- 정상차량
nc: 2
train: /content/drive/MyDrive/New_Sample/overloaded/train.txt
val: /content/drive/MyDrive/New_Sample/overloaded/val.txt
```

```
                 from  n    params  module                                  arguments
  0                -1  1      3520  models.common.Conv                      [3, 32, 6, 2, 2]
  1                -1  1     18560  models.common.Conv                      [32, 64, 3, 2]
  2                -1  1     18816  models.common.C3                        [64, 64, 1]
  3                -1  1     73984  models.common.Conv                      [64, 128, 3, 2]
  4                -1  2    115712  models.common.C3                        [128, 128, 2]
  5                -1  1    295424  models.common.Conv                      [128, 256, 3, 2]
  6                -1  3    625152  models.common.C3                        [256, 256, 3]
  7                -1  1   1180672  models.common.Conv                      [256, 512, 3, 2]
  8                -1  1   1182720  models.common.C3                        [512, 512, 1]
  9                -1  1    656896  models.common.SPPF                      [512, 512, 5]
 10                -1  1    131584  models.common.Conv                      [512, 256, 1, 1]
 11                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
 12          [-1, 6]  1         0  models.common.Concat                    [1]
 13                -1  1    361984  models.common.C3                        [512, 256, 1, False]
 14                -1  1     33024  models.common.Conv                      [256, 128, 1, 1]
 15                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
 16          [-1, 4]  1         0  models.common.Concat                    [1]
 17                -1  1     90880  models.common.C3                        [256, 128, 1, False]
 18                -1  1    147712  models.common.Conv                      [128, 128, 3, 2]
 19         [-1, 14]  1         0  models.common.Concat                    [1]
 20                -1  1    296448  models.common.C3                        [256, 256, 1, False]
 21                -1  1    590336  models.common.Conv                      [256, 256, 3, 2]
 22         [-1, 10]  1         0  models.common.Concat                    [1]
 23                -1  1   1182720  models.common.C3                        [512, 512, 1, False]
 24      [17, 20, 23]  1     18879  models.yolo.Detect                      [2, [[10, 13, 16, 30, 33, 23], [30,
YOLOv5s summary: 214 layers, 7025023 parameters, 7025023 gradients, 16.0 GFLOPs
```

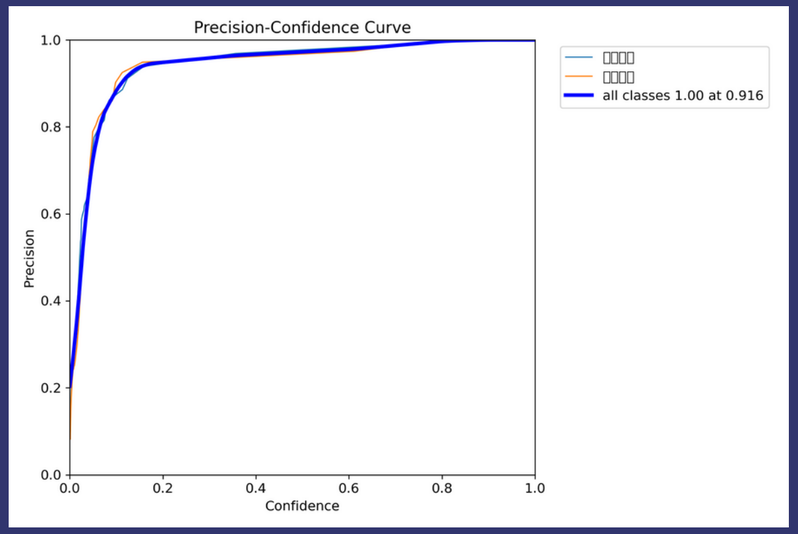
```
# 훈련
%cd /content/yolov5/

!python train.py --img 640 --batch 16 --epochs 50 --data /content/drive/MyDrive/New_Sample/overloaded/data.yaml --cfg ./models/yolov5s.yaml --weights yolov5s.pt --name KHNR_yolov5s_results
```

훈련 결과(image & video)

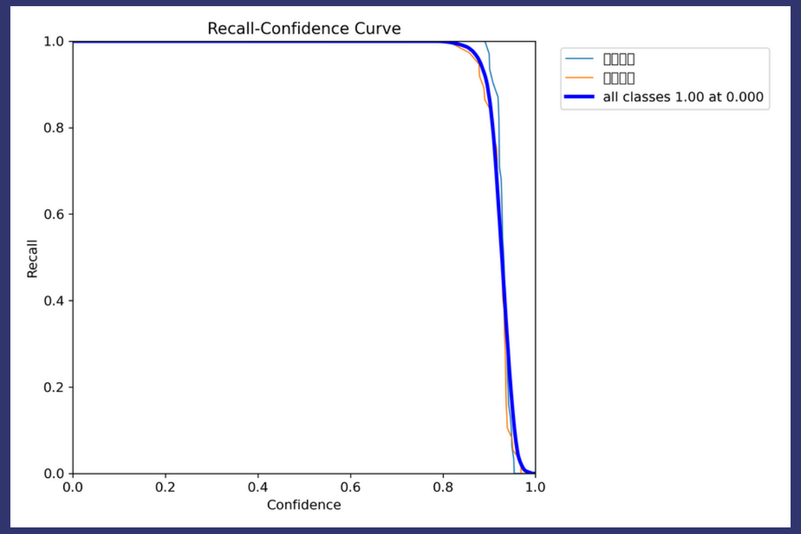
정밀도 그래프

예측을 Positive로 한 대상중에 예측값과 실제값이 Positive로 일치한 데이터의 비율



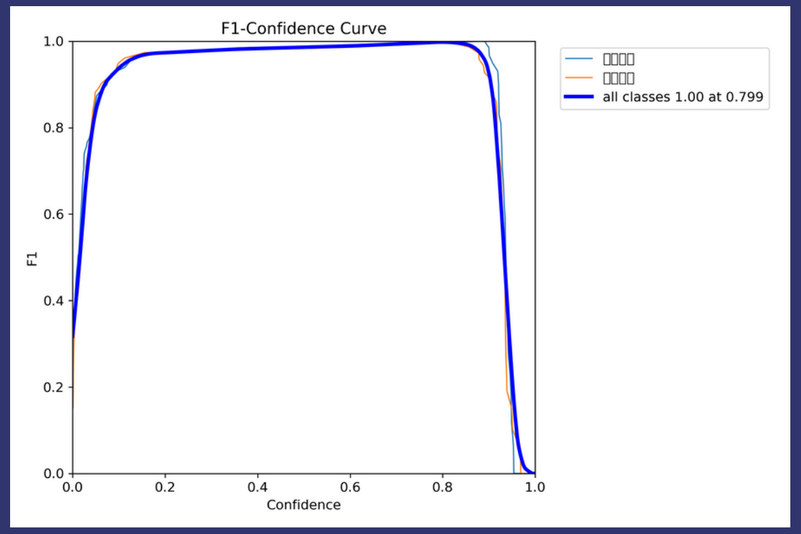
재현율 그래프

실제값이 Positive인 대상중에 예측값과 실제값이 Positive로 일치한 데이터의 비율



F1_score그래프

정밀도와 재현율의 조화평균



훈련 결과(image & video)

정상 차량과 불법차량을 정확하지는 않지만
어느정도 판별하는 것 같다.

화물 차량의 데이터로만 학습을 하여서
화물 트럭의 경우로 잘 판별을 한다.

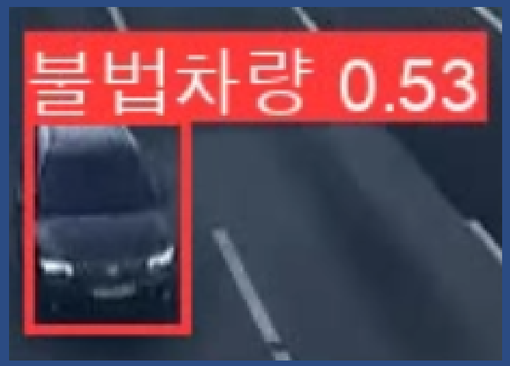
훈련 결과(image & video)

사람의 경우도 불법차량인 경우로 잘 못판단 하는
경우도 있다.

다른 큰 차량 (버스, SUV)의 경우로
잘 판단하지 못하는 경우가 있다.

앞 모습 같은경우도 판단을 잘 못하는 경우도 있다.

학습 데이터가 매우 적어서 이런 결과가
나오는 것이라고 판단된다.



노트북으로 실행 할 때

- test_overload.ipynb 를 직접 다운로드하거나 눌러서 open in colab 확장프로그램을 이용하여 노트북을 엽니다.
- 노트북의 순서에 따라서 번호 밑의 재생버튼을 클릭하거나 ctrl+F9 를 통해서 전체 실행을 해줍니다.
  - 1 : git clone을 이용해서 모델과 requirements.txt 불러와서 로드하는 과정 꼭 한번만 눌러주세요
  - 2 : 테스트 영상를 업로드 하는과정 테스트파일이 없다면 video_samples폴더에서 랜덤으로 다운로드하여 사용합니다.
  - 3 : 업로드한 영상속 화물차량이 불법차량인지 정상차량인지 판단하는 과정. 테스트가 완료된 영상은 AI_16_CP1_DS -> output에서 확인가능합니다
  - 3.1 : 버튼을 누릅니다. > input 폴더에 검증했던 영상을 삭제합니다. > 2번 부터 다시 시작합니다.
    - 3.1 재시작 버튼은 재시작 할 때 한 번만 눌러야합니다.
    - 모두 실행(ctrl+F9)하였을 때는 누르지 않고 input 폴더에 검증했던 영상을 삭제 후 2번으로 다시시작합니다.

로컬 환경에서 실행 할 때

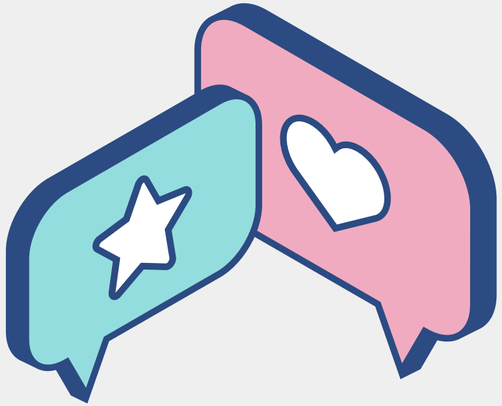
1. 레포지토리 클론

- `git clone https://github.com/KANGHANNARA/AI_16_CP1_DS`

2. `cd AI_16_CP1_DS`로 폴더로 들어가서 python==3.8로 가상환경을 만듭니다.

- `pip install -r requirements.txt` 필요한 라이브러리와 모듈을 설치합니다.

3. test/test_cv2.py 파일에서 검증 할 영상의 경로를 입력합니다.(default는 샘플영상의 ./video_samples/sample5.mp4)
4. `python test/test_cv2.py` 로 실행합니다.
5. 결과영상은 result 폴더에 저장됩니다.



Feed back

동기들의 도움을 많이 받았습니다.
도움 받은 코드를 하나하나 뜯어보면서 이해를 하였고
유튜브의 yolov5 커스텀 튜토리얼을 보면서 그 코드도 하나
하나 뜯어보면서 이해를 하며 실행하여서 재미있게 프로젝트
를 한 것 같습니다.
추후에 샘플데이터 170개가 아닌 소형,중형,대형으로 나뉘어
진 aihub의 400,000장의 데이터로 중복이미지를 제거하여
학습을 시키고 yolov5s가 아닌 M이나 L의 경우로
학습을 시켜보는 것이 목표입니다.

감사합니다!