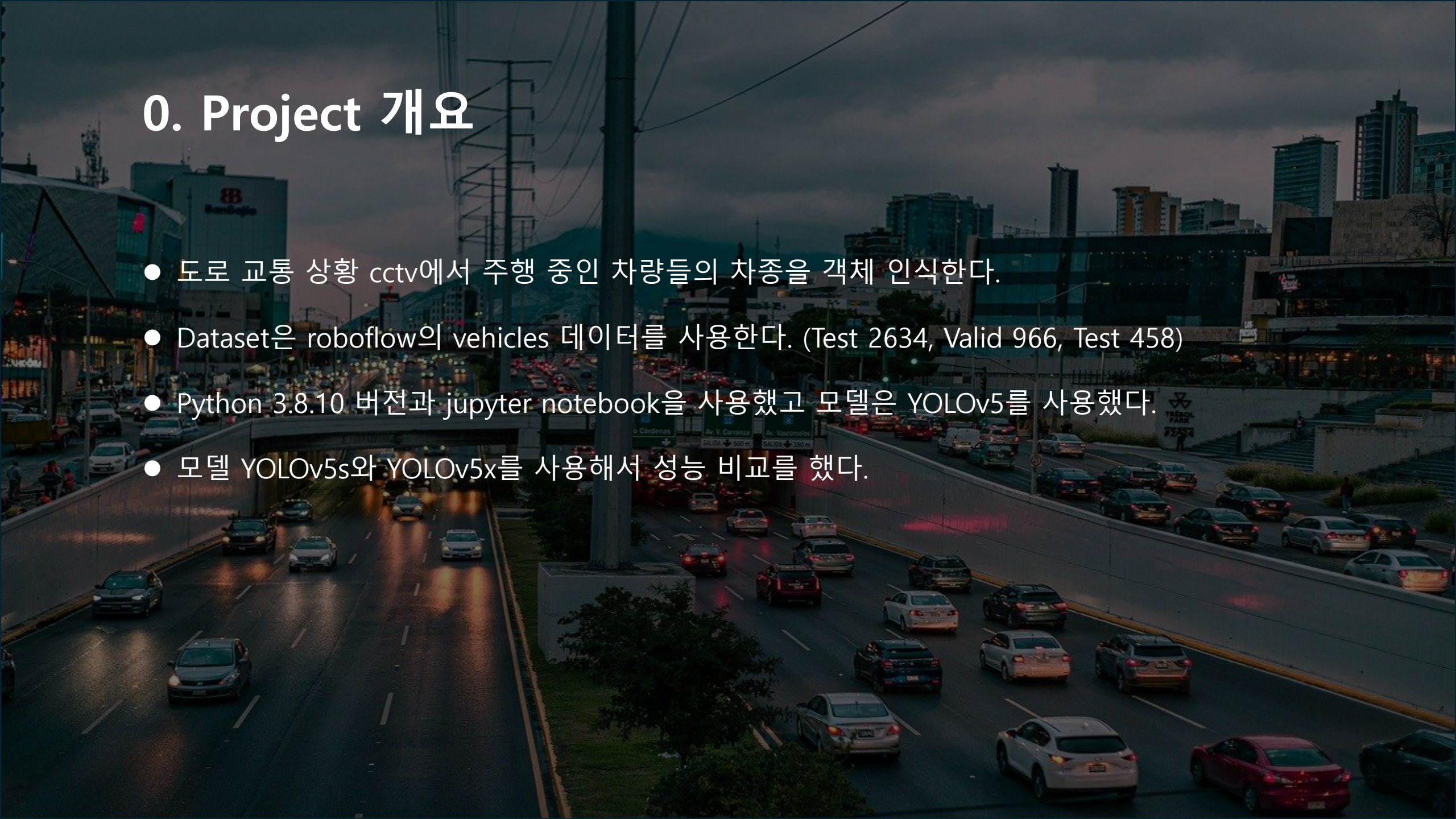
YOLOV5 MINI PROJECT

영상 속 차량 탐지

CONTENTS

0. Project 개요



- 도로 교통 상황 cctv에서 주행 중인 차량들의 차종을 객체 인식한다.
- Dataset은 roboflow의 vehicles 데이터를 사용한다. (Test 2634, Valid 966, Test 458)
- Python 3.8.10 버전과 jupyter notebook을 사용했고 모델은 YOLOv5를 사용했다.
- 모델 YOLOv5s와 YOLOv5x를 사용해서 성능 비교를 했다.

1. YOLOV5 불러오기

```
!git clone https://github.com/ultralytics/yolov5.git
```

Github에서 yolov5 프로젝트 복제

```
%cd yolov5
```

```
!pip install -qr requirements.txt
```

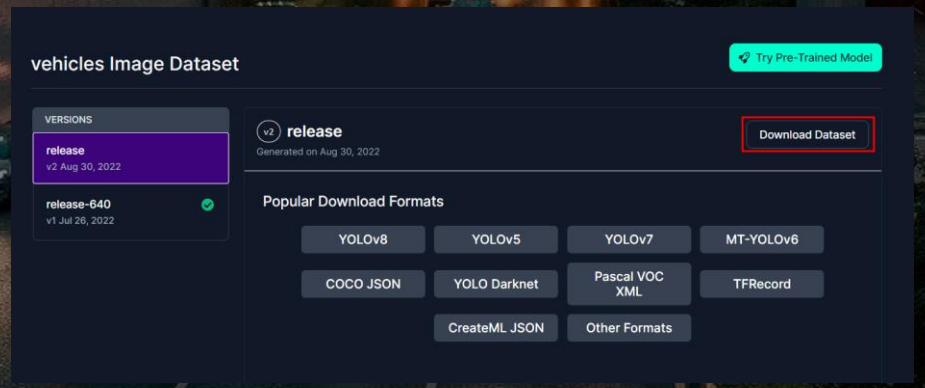
yolov5 폴더로 이동 후
Requirements.txt 파일에 명시된
Python 패키지를 모두 설치

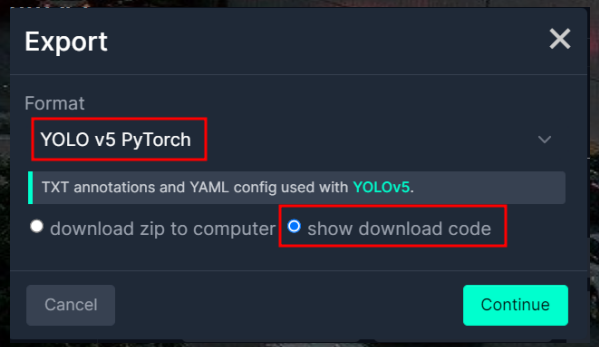
2. Train Data Download

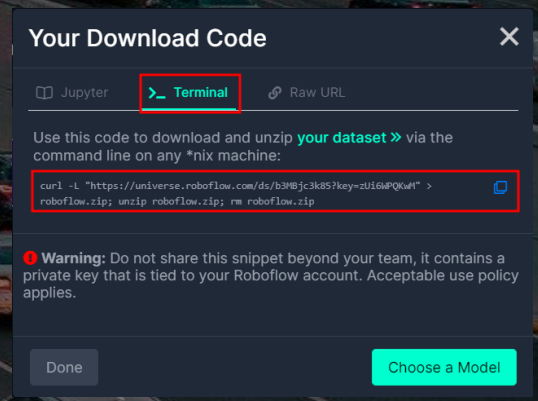
roboflow 사이트에서 사용할 dataset 다운

● 참고 사이트

https://universe.roboflow.com/roboflow-100/vehicles-q0x2v/dataset/2







2. Train Data Download

```
%cd ..
%mkdir vehicles_yolo
%cd vehicles_yolo
```

훈련 데이터를 다운받을 폴더 생성 후 이동

```
!curl -L "https://universe.roboflow.com/ds/b3MBjc3k85?key=zUi6WPQKwM" > roboflow.zip; unzip roboflow.zip; rm roboflow.zip
```

roboflow에서 dataset download 코드 사용

압축 파일을 다운 받아 압축 풀기 후 압축 파일은 제거

3. Yaml 파일 수정

```
%cd ../yolov5
```

```
import yaml
with open('../vehicles_yolo/data.yaml','r') as f:
    data=yaml.load(f,Loader=yaml.FullLoader)

    data['train'] = '../vehicles_yolo/'
    data['test'] = '../vehicles_yolo/'
    data['val'] = '../vehicles_yolo/'

    with open('../vehicles_yolo/data.yaml','w') as f:
        yaml.dump(data, f)
    print(data)
```

yolov5 폴더로 이동

yolov5를 사용할 train, test, val dataset이 있는 폴더로 경로를 설정.

4. 모델 훈련

```
!python train.py --img 480 --batch 16 --epochs 10 --data ../vehicles_yolo/data.yaml
--cfg ./models/yolov5x.yaml --weights yolov5x.pt --name vehicles_yolo_Result --workers 0
```

	학습 스크립트인 Train.py 실행
--img	이미지 크기를 480*480 설정
--batch	Batch size 16
--epochs	Epoch (학습 반복 횟수) 10
--data	학습에 필요한 데이터 구성을 지정하는 YAML 파일
--cfg	YOLOv5 모델의 구조를 정의하는 YAML 파일
--weights	모델 가중치 설정
--name	학습 결과를 저장할 폴더 이름
--workers	DataLoader에 사용할 CPU 작업자의 수 설정. 0으로 설정하여 모든 작업을 메인 프로세스에서 처리

```
Validating runs/train/vehicles_yolo_Result10/weights/best.pt...
Fusing layers...
YOLOv5x summary: 322 layers, 86247433 parameters, 0 gradients, 204.0 GFLOPs
                 Class     Images  Instances          P          R      mAP50
                   all       4058      51575      0.716      0.626      0.606      0.435
               big bus       4058        816      0.601      0.892      0.722      0.535
             big truck       4058       3631      0.733      0.814      0.843      0.562
                bus-l-       4058        398       0.33      0.736      0.329      0.251
                bus-s-       4058        148          1          0     0.0414      0.034
                   car       4058      31641      0.834      0.892      0.916      0.588
             mid truck       4058        703      0.815      0.289      0.449      0.348
             small bus       4058        263          1          0      0.148      0.122
           small truck       4058       5841      0.743      0.802      0.829      0.537
               truck-l-      4058       2278      0.654      0.864      0.844       0.62
               truck-m-      4058       3672      0.715      0.859      0.846      0.634
               truck-s-      4058       1363      0.606      0.551      0.554      0.406
              truck-xl-      4058        821      0.564      0.819      0.748      0.582
Results saved to runs/train/vehicles_yolo_Result10
```

결과 화면 및 저장한 파일의 경로 출력

5. Object Detection 수행

```
!python detect.py --weights 'runs/train/vehicles_yolo_Result10/weights/best.pt'
--img 480 --conf 0.3 --source ../video.mp4
```

	객체 탐지 스크립트인 detect.py 실행
--weights	학습된 모델의 가중치 파일
--img	입력 영상의 크기 480*480
--conf	객체를 감지하는 임계값 설정. 0.3보다 높은 객체만 검출
--source	탐지를 수행할 소스 설정.

```
Speed: 0.3ms pre-process, 23.7ms inference, 0.8ms NMS per image at shape (1, 3, 480, 480)
Results saved to runs/detect/exp7
```

결과 화면 및 저장한 파일의 경로 출력

6. 성능 검증

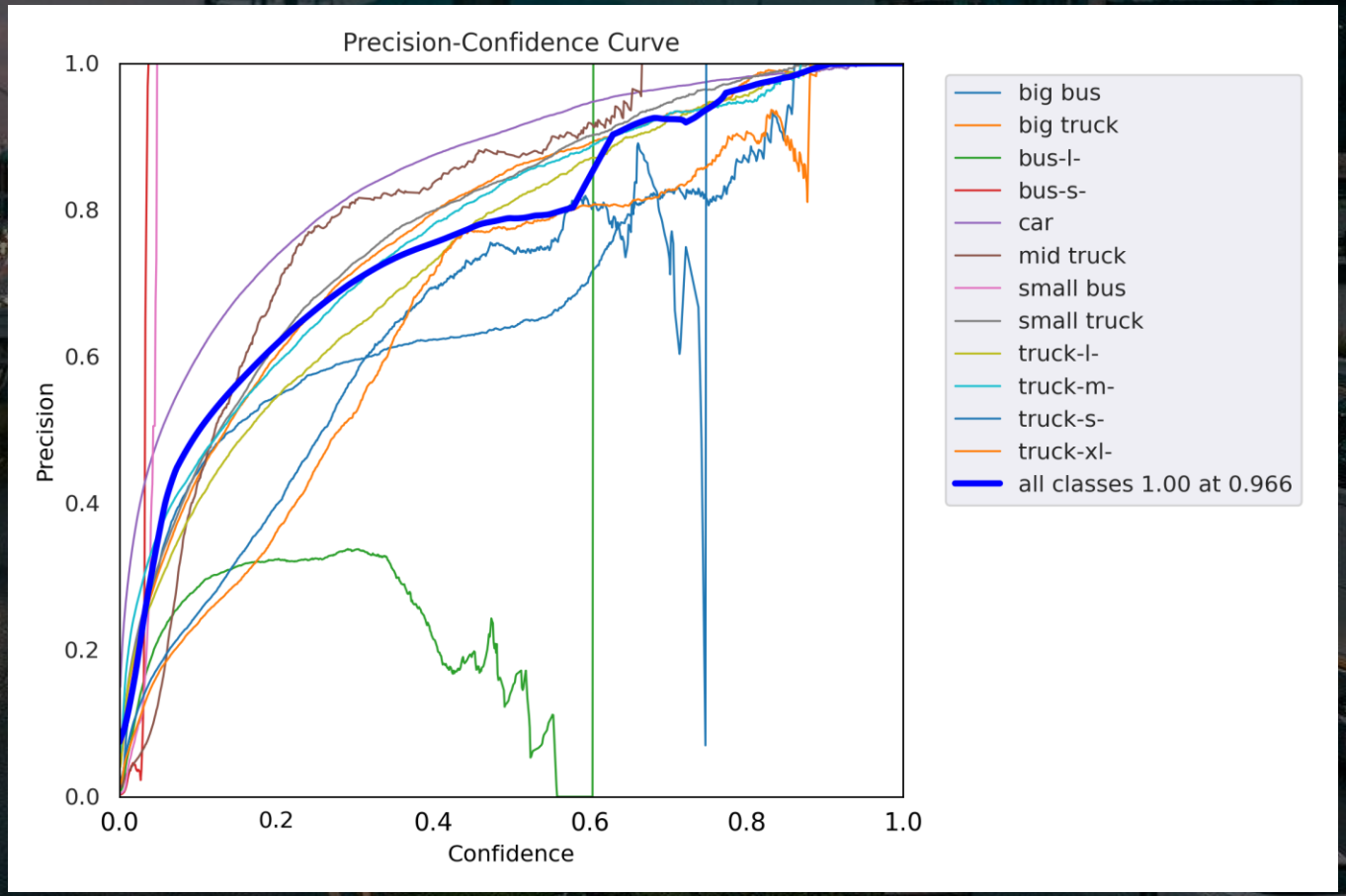
```
!python val.py --weights ../yolov5/runs/train/vehicles_yolo_Result10/weights/best.pt
--data ../vehicles_yolo/data.yaml --img 480 --batch-size 16 --conf 0.001 --device 0
```

	검증 스크립트인 val.py 실행
--weights	검증에 사용할 모델의 가중치 파일
--data	검증에 필요한 데이터 구성을 지정하는 YAML 파일
--img	입력 이미지 크기 480*480
--batch-size	배치 크기 16
--conf	객체를 감지하는 임계값 설정 0.001보다 높은 신뢰도를 가진 객체 검출
--device	모델을 실행할 GPU 선택. 0번 GPU 사용

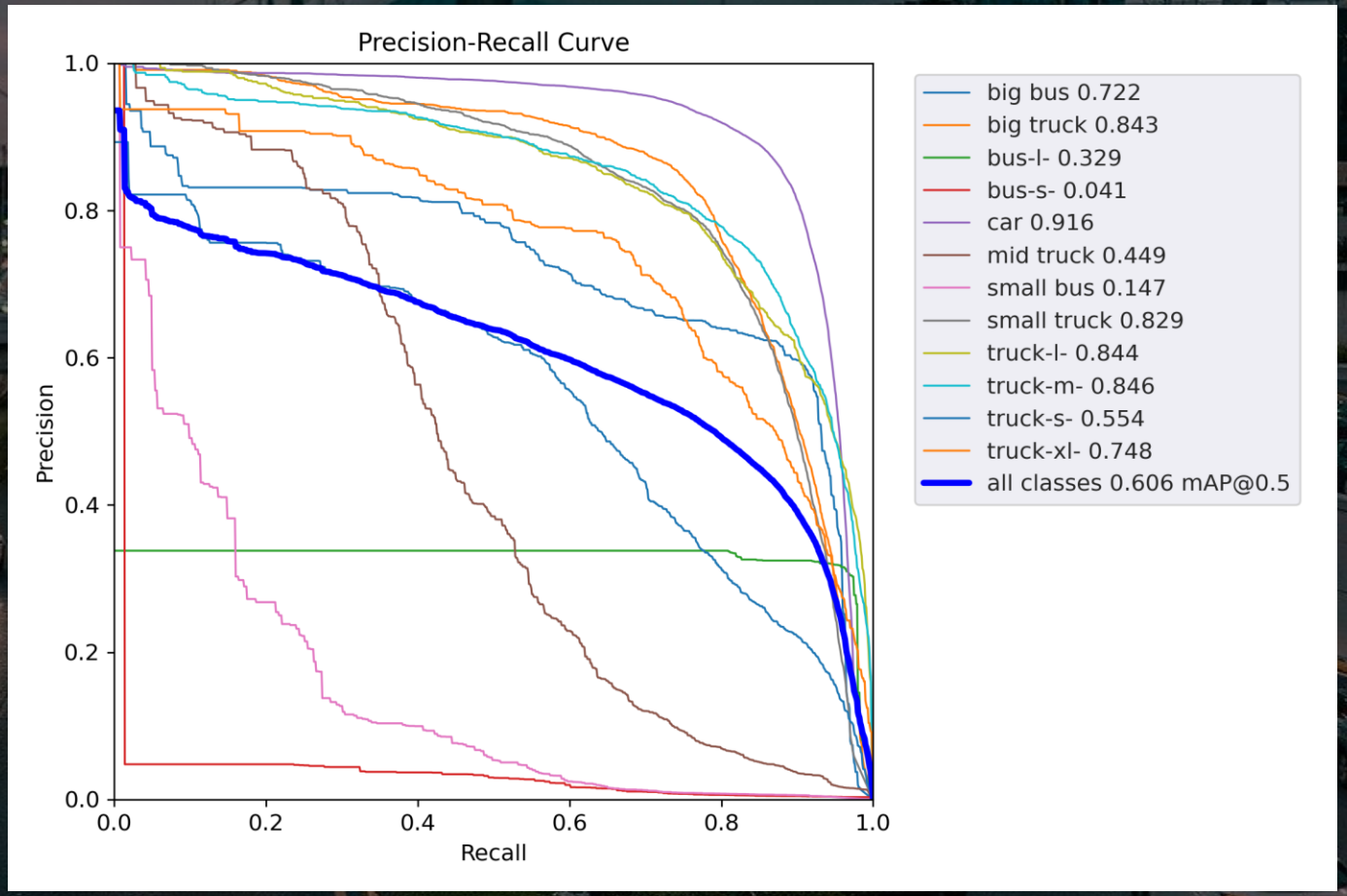
```
Fusing layers...
YOLOv5x summary: 322 layers, 86247433 parameters, 0 gradients, 204.0 GFLOPs
val: Scanning /workspace/vehicles_yolov5/vehicles_yolo/test/labels.cache... 4058
train: WARNING ⚠ /workspace/vehicles_yolov5/vehicles_yolo/train/images/adit_mp4-1357_jpg.rf
                 Class     Images  Instances          P          R      mAP50
                   all       4058      51575      0.716      0.627      0.606      0.435
               big bus       4058        816      0.601      0.892      0.722      0.535
             big truck       4058       3631      0.733      0.814      0.843      0.563
                bus-l-       4058        398      0.331      0.738      0.329      0.251
                bus-s-       4058        148          1          0     0.0415     0.0341
                   car       4058      31641      0.834      0.892      0.916      0.588
             mid truck       4058        703      0.815      0.289      0.449      0.349
             small bus       4058        263          1          0      0.147      0.122
           small truck       4058       5841      0.744      0.802      0.829      0.538
               truck-l-      4058       2278      0.653      0.864      0.844       0.62
               truck-m-      4058       3672      0.715      0.859      0.846      0.633
               truck-s-      4058       1363      0.606      0.551      0.554      0.406
               truck-xl-     4058        821      0.564      0.819      0.748      0.581
Speed: 0.1ms pre-process, 20.4ms inference, 2.2ms NMS per image at shape (16, 3, 480, 480)
Results saved to runs/val/exp4
```

결과 화면 및 저장한 파일의 경로 출력

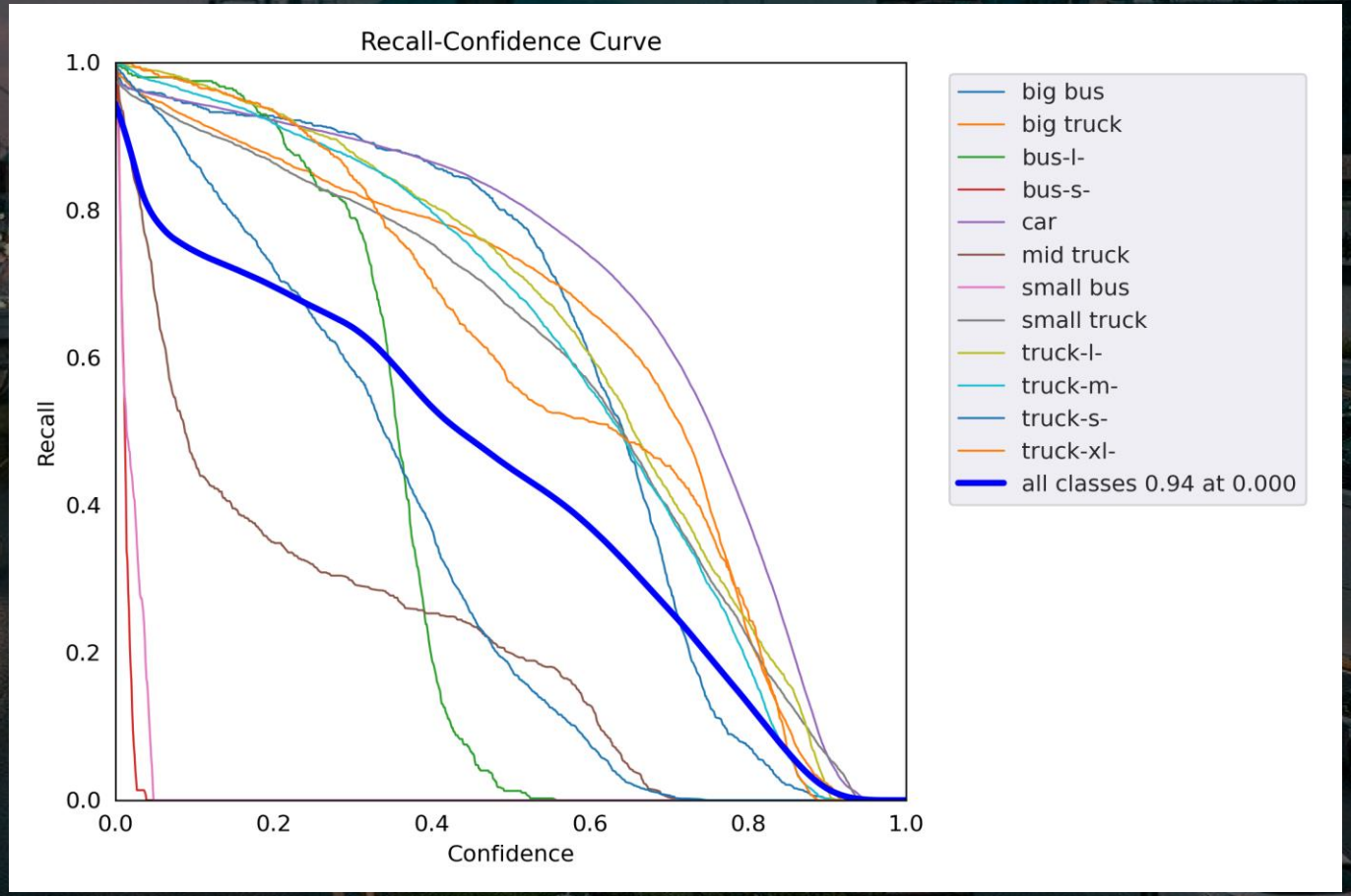
6. 성능 검증
Precision-Confidence



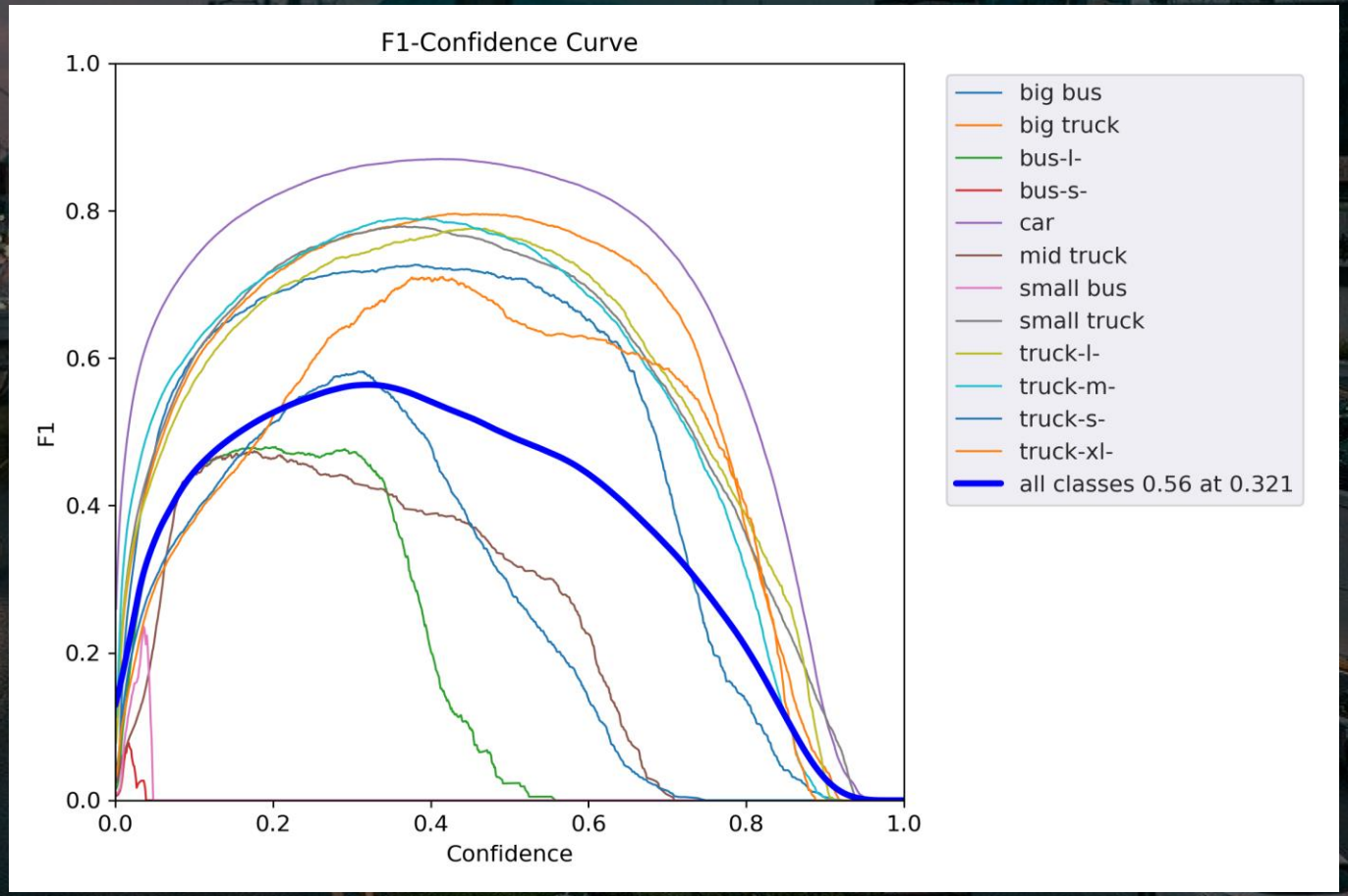
6. 성능 검증
Precision-Recall



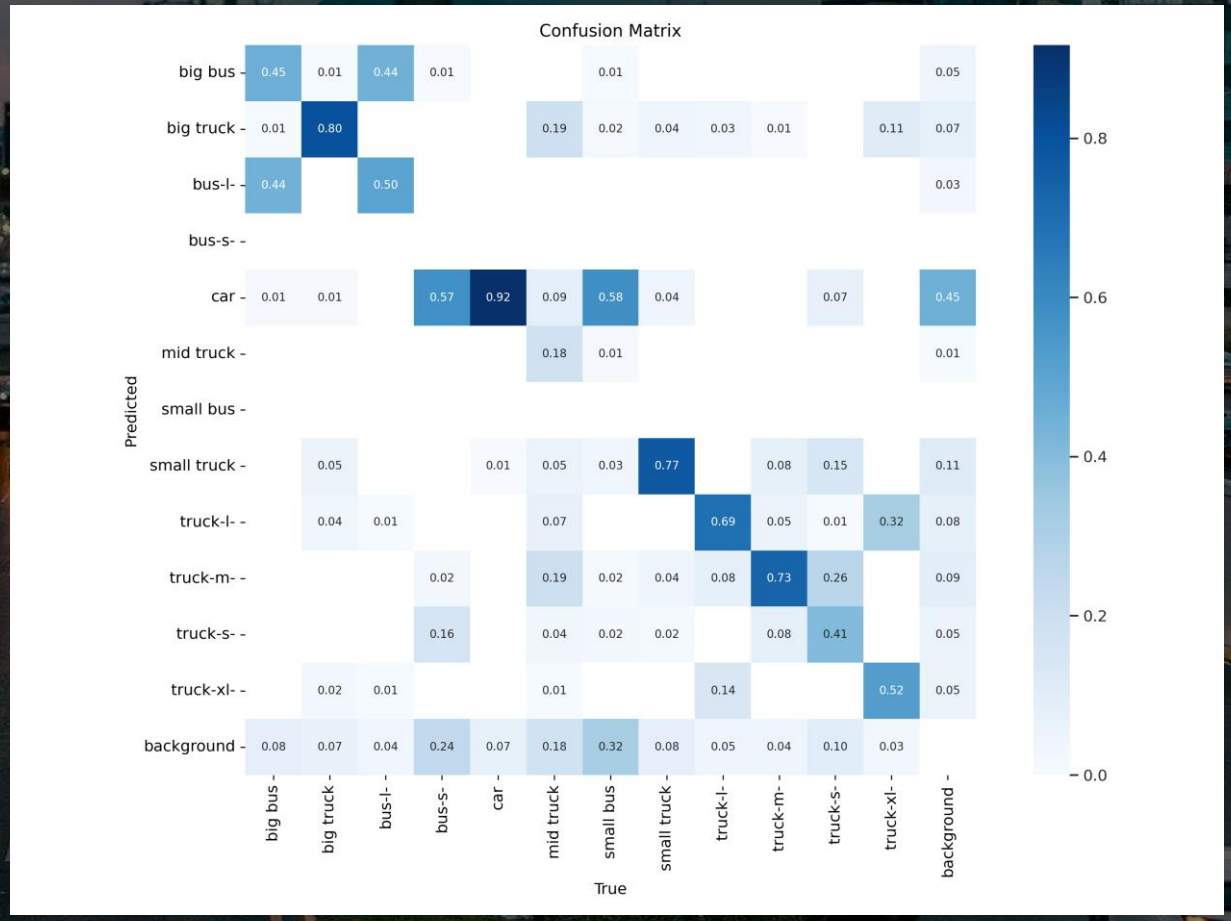
6. 성능 검증
Recall-Confidence



6. 성능 검증
F1-Confidence



6. 성능 검증
Confusion Matrix



7. YOLOv5s와 YOLOv5x 영상 비교

YOLOv5s

YOLOv5x

YOLOv5x가 YOLOv5s보다 많은 차량, 다양한 차종을 인식한다.

8. 한계 및 개선 방향

- 훈련 dataset 이미지에 12개의 클래스 중 'car'가 전체의 61%였다. 'car' 클래스는 인식을 잘하지만 다른 클래스는 상대적으로 학습이 적게 되어 인식률이 낮다.
- 훈련 dataset 이미지는 차량의 모습이 앞 또는 뒤 모습이다. 따라서 옆 모습이 나오는 영상에서는 인식이 거의 되지 않는다.
- 차량의 옆 모습까지 훈련하고 버스나 트럭 등 다양한 차량을 훈련시키면 정확도를 높일 수 있다.
- 이러한 객체 탐지를 이용해서 실시간으로 도로 교통 상황이 혼잡 또는 원활한지 파악할 수 있다.

THANK YOU

전체 코드 및 영상

링크