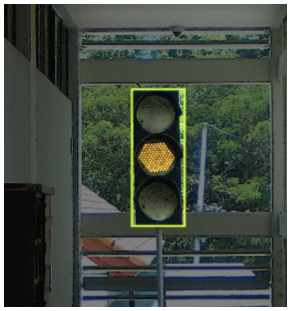
6조(김예진, 박신영, 백수안, 오예인, 최서윤)

<자율주행 자동차를 위한 신호등 감지> 코드 테스트 보고서

0. GOAL

미국의 자동차 제조사 T에서 자율주행자동차 소프트웨어로 활용할 수 있는 신호등 인식 AI 모델을 요청해왔다. Image Detection Model을 활용해 신호를 감지할 수 있는 모델을 구축해보자.

1. Traffic Light Data




```
{
    "boxes": [
        {
            "label": "yellow",
            "x": 694,
            "y": 687,
            "width": 236,
            "height": 610
        }
    ],
    "height": 1360,
    "key": "traffic light (101).jpg",
    "width": 1360
}
```


roboflow에서 다운로드한 cinTA_v2 Computer Vision Project의 신호등 이미지 데이터는 왼쪽 이미지와 같이 생겼으며 오른쪽과 텍스트와 같은 형태로 라벨링되어 있다.

2. Train and Evaluate Models

2.0 Select Model

2015년 처음 발표된 YOLOv1(You Only Look Once)은 객체 탐지를 위한 하나의 신경망 모델로 이미지는 S x S 그리드로 나누고 각 그리드 셀에 대해 바운딩 박스와 확률을 예측한다. YOLOv1의 핵심 아이디어는 이미지를 한번에 전체적으로 처리해 객체 탐지 속도를 크게 향상시키는 것이다. YOLOv1으로 시작한 YOLO 시리즈의 가장 최신 버전은 YOLOv8이다. YOLOv8은 컨벌루션 레이어와 트랜스포머 레이어를 결합한 하이브리드 아키텍쳐를 갖는다. 본 프로젝트에서는 YOLOv8을 활용해 신호등을 탐지하고자 한다.

모델	크기 (픽셀)	mAPval 50-95	속도 CPU ONNX (ms)	속도 A100 TensorRT (ms)	매개변수 (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

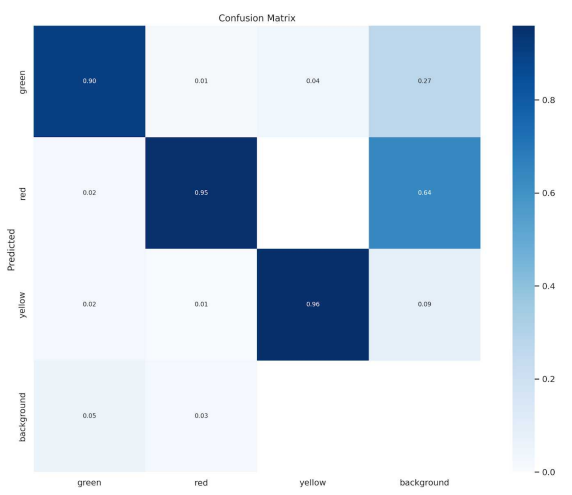
YOLOv8에는 5가지의 pretrained 모델이 제공된다. 위 표에는 n부터 x까지 작은 모델부터 큰 모델의 순서로 정리되어 있다. 작은 모델은 적은 수의 매개변수를 갖기 때문에 속도는 빠르나 정확도가 낮다은 반면, 큰 모델은 많은 매개변수를 갖기 때문에 속도는 느리지만 정확도가 높다. 본 프로젝트에서는 같은 데이터셋을 활용해 5개의 pretrained 모델 중 신호등 탐지에 가장 적합한 모델을 찾는 것을 목표로 했다.

2.1 Training & Validation

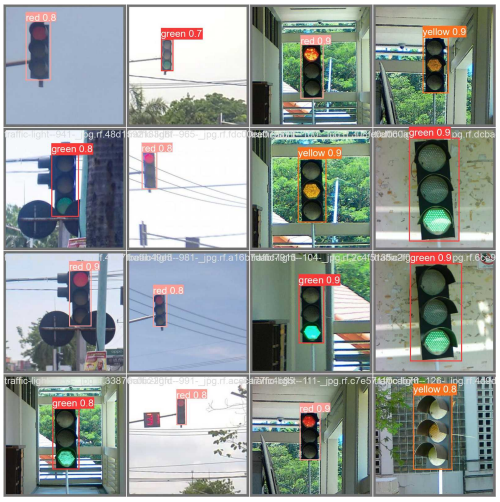
2.1.1 YOLOv8n

```
Model summary: 225 layers, 3011433 parameters, 3011417 gradients, 8.2 GFLOPs
```

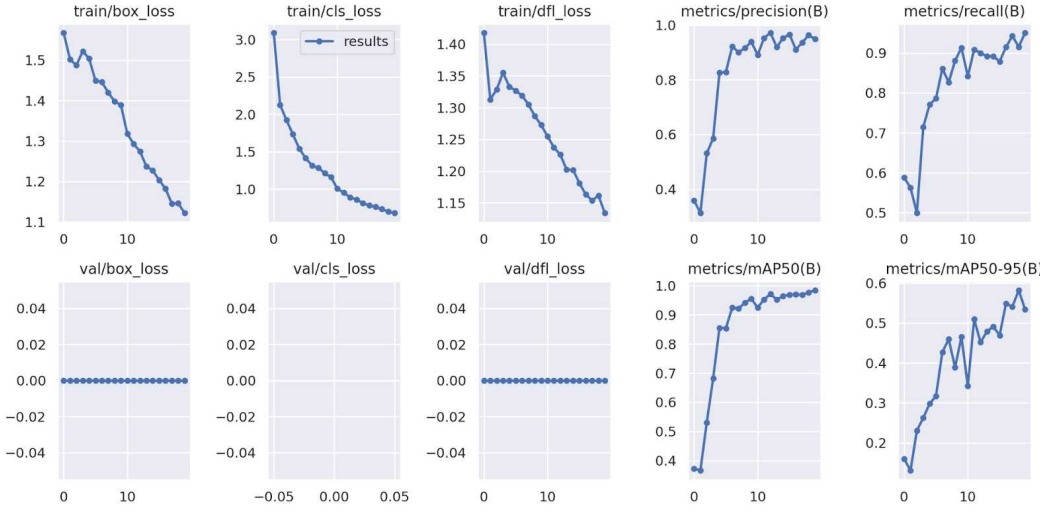
n모델은 225개의 레이어, 3,011,433개의 파라미터로 구성된다.



confusion matrix



predict validation data



x: epoch y: loss, precision, recall, mAP

train 결과, 모델의 성능은 첨부된 confusion matrix, 그래프와 같았으며 validation data를 predict한 결과는 첨부된 사진과 같았다.

val:

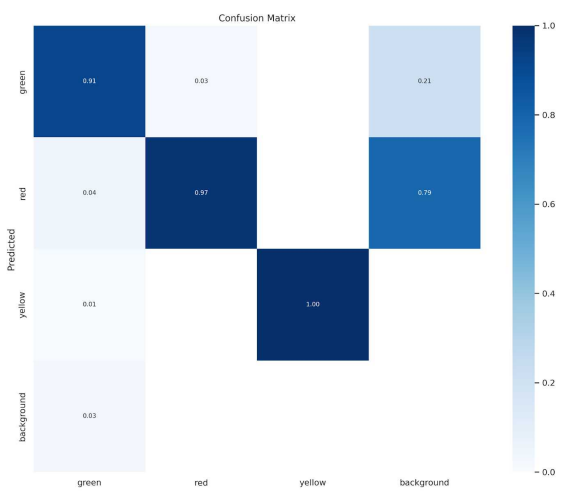
Class	Images	Instances	Box(P	R	mAP50	mAP50-95)
all	200	272	0.962	0.916	0.977	0.58
green	200	92	1	0.859	0.971	0.575
red	200	155	0.978	0.929	0.982	0.579
yellow	200	25	0.909	0.96	0.978	0.587

epoch을 돌리며 가장 성능이 좋았던 모델의 weight를 불러와 validation한 결과, 위와 같은 값이 출력되었다.

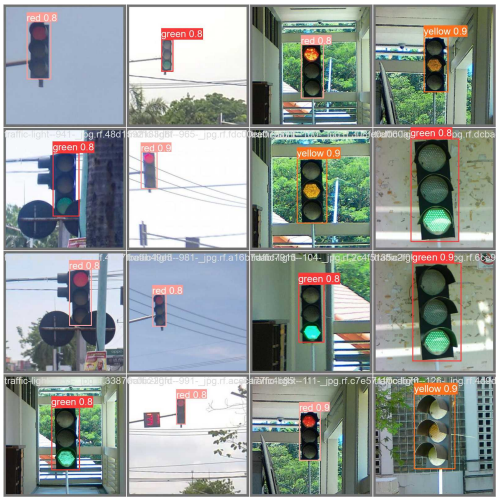
2.1.2 YOLOv8s

```
Model summary: 168 layers, 11126745 parameters, 0 gradients, 28.4 GFLOPs
```

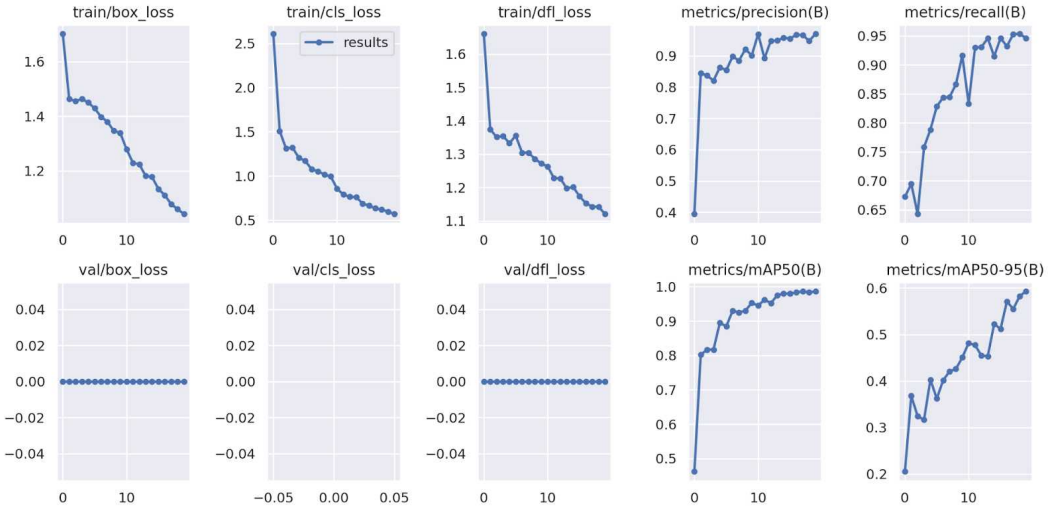
s모델은 168개의 레이어, 11,126,745개의 파라미터로 구성된다.



confusion matrix



predict validation data



x: epoch y: loss, precision, recall, mAP

train 결과, 모델의 성능은 첨부된 confusion matrix, 그래프와 같았으며 validation data를 predict한 결과는 첨부된 사진과 같았다.

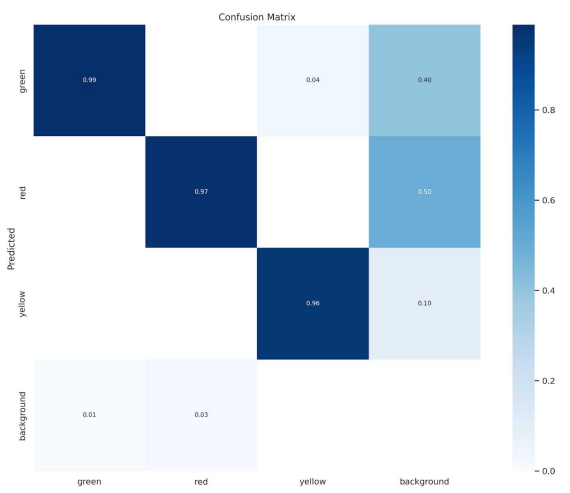
```
val:
 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)
   all        200        272      0.971      0.947      0.986      0.594
 green        200         92      0.988       0.89      0.976      0.558
   red        200        155      0.974      0.951      0.987      0.618
yellow        200         25      0.951          1      0.995      0.604
Speed: 2.8ms pre-process, 10.6ms inference, 0.0ms loss, 6.0ms post-process
per image
```

epoch을 돌리며 가장 성능이 좋았던 모델의 weight를 불러와 validation한 결과, 위와 같은 값이 출력되었다.

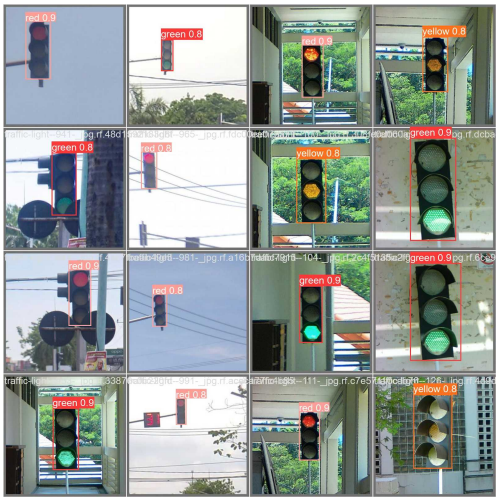
2.1.3 YOLOv8m

```
Model summary: 218 layers, 25841497 parameters, 0 gradients, 78.7 GFLOPs
```

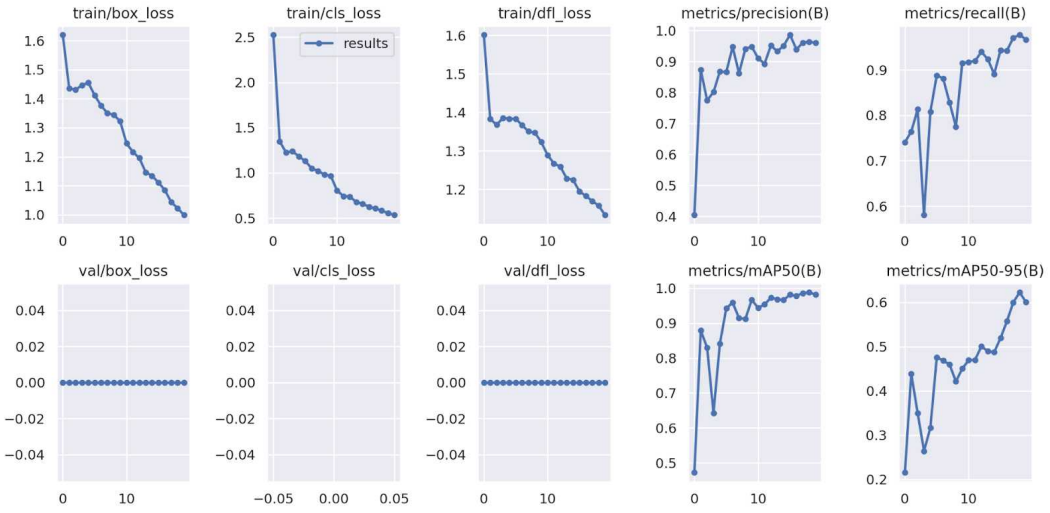
m모델은 218개의 레이어, 25,841,497개의 파라미터로 구성된다.



confusion matrix



predict validation data



x: epoch y: loss, precision, recall, mAP

train 결과, 모델의 성능은 첨부된 confusion matrix, 그래프와 같았으며 validation data를 predict한 결과는 첨부된 사진과 같았다.

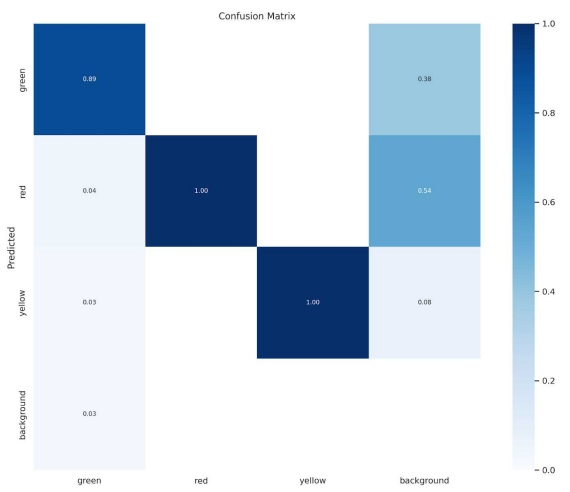
```
val:
 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)
   all        200        272      0.964      0.976      0.989       0.62
 green        200         92      0.952      0.989      0.979      0.617
   red        200        155       0.98      0.971      0.993      0.626
yellow        200         25       0.96      0.968      0.993      0.616
Speed: 1.5ms pre-process, 21.5ms inference, 0.0ms loss, 5.1ms post-process
per image
```

epoch을 돌리며 가장 성능이 좋았던 모델의 weight를 불러와 validation한 결과, 위와 같은 값이 출력되었다.

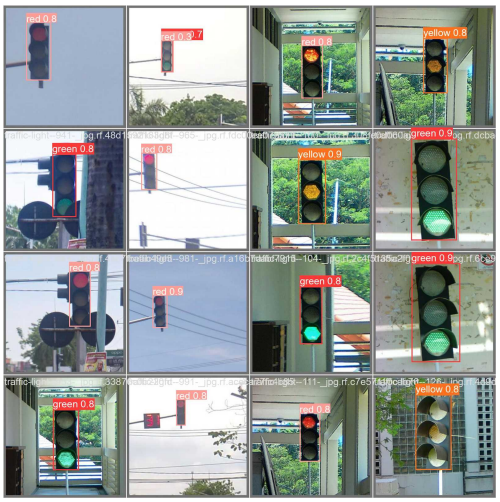
2.1.4 YOLOv8l

```
Model summary: 268 layers, 43608921 parameters, 0 gradients, 164.8 GFLOPs
```

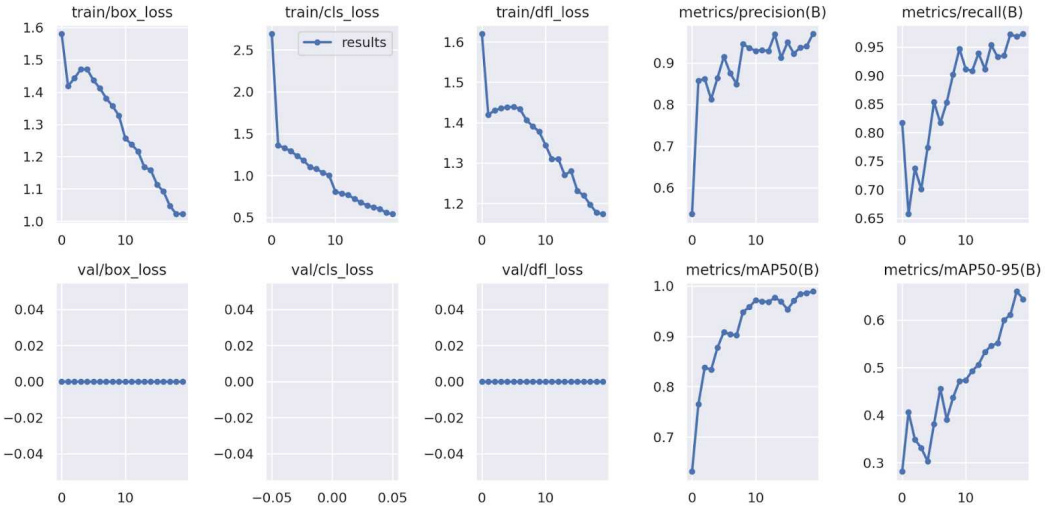
l모델은 268개의 레이어, 43,608,921개의 파라미터로 구성된다.



confusion matrix



predict validation data



x: epoch y: loss, precision, recall, mAP

train 결과, 모델의 성능은 첨부된 confusion matrix, 그래프와 같았으며 validation data를 predict한 결과는 첨부된 사진과 같았다.

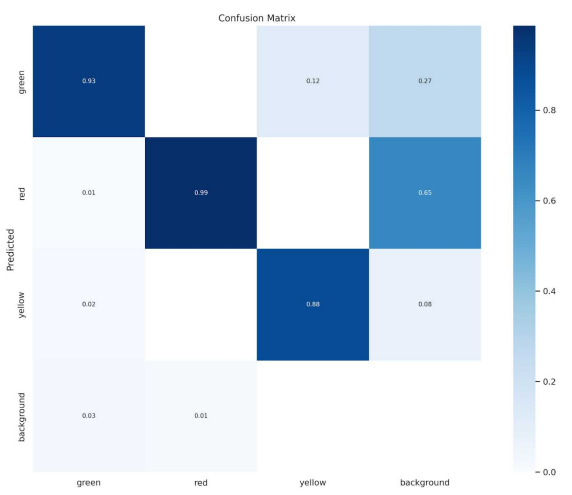
```
val:
 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)
   all        200        272      0.938      0.968      0.986      0.662
 green        200         92      0.988      0.931      0.974      0.624
   red        200        155      0.963      0.974      0.991      0.653
yellow        200         25      0.863          1      0.992      0.708
Speed: 1.4ms pre-process, 38.0ms inference, 0.0ms loss, 7.6ms post-process
per image
```

epoch을 돌리며 가장 성능이 좋았던 모델의 weight를 불러와 validation한 결과, 위와 같은 값이 출력되었다.

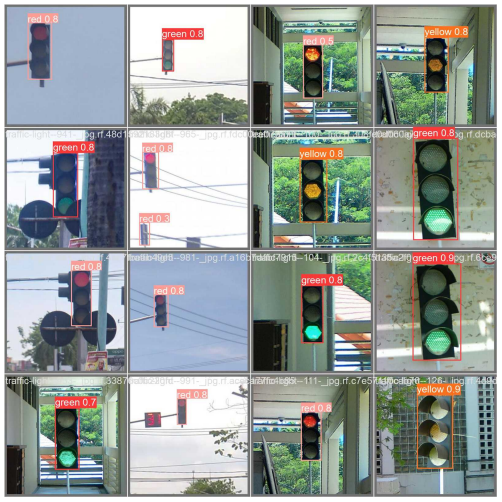
2.1.5 YOLOv8x

```
Model summary: 268 layers, 68126457 parameters, 0 gradients, 257.4 GFLOPs
```

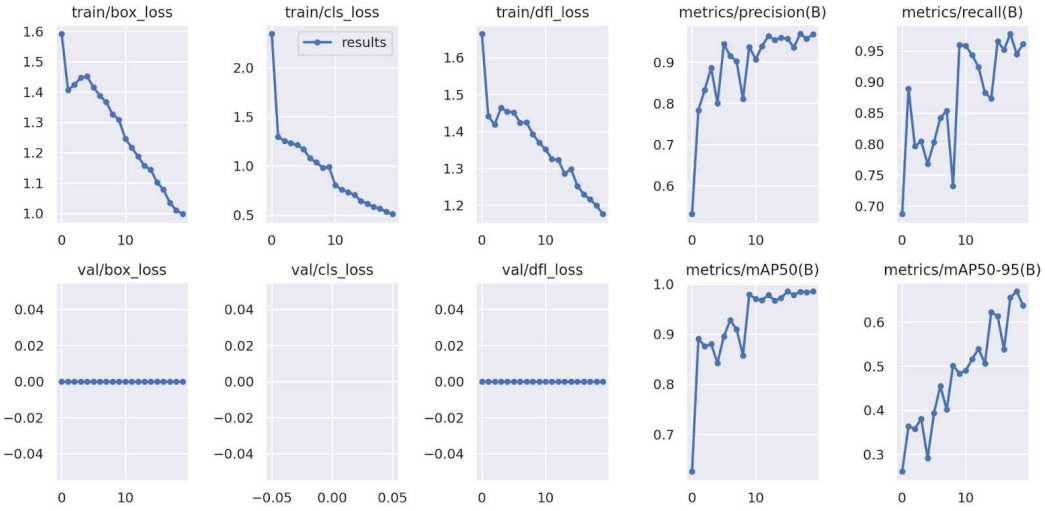
x모델은 268개의 레이어, 68,126,457개의 파라미터로 구성된다.



confusion matrix



predict validation data



x: epoch y: loss, precision, recall, mAP

train 결과, 모델의 성능은 첨부된 confusion matrix, 그래프와 같았으며 validation data를 predict한 결과는 첨부된 사진과 같았다.

```
val:
 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)
   all        200        272      0.952       0.87      0.919      0.619
 green        200         92      0.964      0.876      0.921      0.591
   red        200        155      0.967      0.935      0.964      0.643
yellow        200         25      0.925        0.8      0.874      0.622
Speed: 5.4ms pre-process, 59.9ms inference, 0.0ms loss, 12.3ms post-process
per image
```

epoch을 돌리며 가장 성능이 좋았던 모델의 weight를 불러와 validation한 결과, 위와 같은 값이 출력되었다.

2.2 Model Comparison

	YOLOv8n	YOLOv8s	YOLOv8m	YOLOv8l	YOLOv8x
speed	14.9ms	18.9ms	28.2ms	44.6ms	66.2ms
mAP50	0.977	0.986	0.989	0.986	0.919
mAP50-95	0.580	0.594	0.620	0.662	0.619

YOLOv8의 5가지 pretrained 모델의 mAP 값과 시간 효율을 정리해보면 위의 표와 같다. 시간을 x축으로 그래프를 그렸을 때 mAP50 값은 m모델에서, mAP50-95 값은 l 모델에서 변곡점을 가질 것이다. 고로 mAP를 평가지표로 했을 때 m모델과 l모델의 성능이 가장 좋다는 것을 확인할 수 있다.

3. Apply a Model on Test Dataset

3.1.1 YOLOv8n

```
Speed: 0.5ms pre-process, 6.7ms inference, 7.7ms postprocess per image at
shape (1, 3, 640, 640)
```

YOLOv8n test result

test data에 대해 prediction을 진행한 결과, 추론에 (1, 3, 640, 640) 이미지 당 14.9ms의 시간이 걸렸으며 예측 결과는 위와 같았다.

3.1.2 YOLOv8s

```
Speed: 0.5ms pre-process, 9.3ms inference, 9.1ms postprocess per image at
shape (1, 3, 640, 640)
```

YOLOv8s test result

test data에 대해 prediction을 진행한 결과, 추론에 (1, 3, 640, 640) 이미지 당 18.9ms의 시간이 걸렸으며 예측 결과는 위와 같았다.

3.1.3 YOLOv8m

```
Speed: 0.5ms pre-process, 20.0ms inference, 7.7ms postprocess per image at shape (1, 3, 640, 640)
```

YOLOv8m test result

test data에 대해 prediction을 진행한 결과, 추론에 (1, 3, 640, 640) 이미지 당 28.2ms의 시간이 걸렸으며 예측 결과는 위와 같았다.

3.1.4 YOLOv8l

```
Speed: 0.6ms pre-process, 36.4ms inference, 7.6ms postprocess per image at shape (1, 3, 640, 640)
```

YOLOv8l test result

test data에 대해 prediction을 진행한 결과, 추론에 (1, 3, 640, 640) 이미지 당 44.6ms의 시간이 걸렸으며 예측 결과는 위와 같았다.

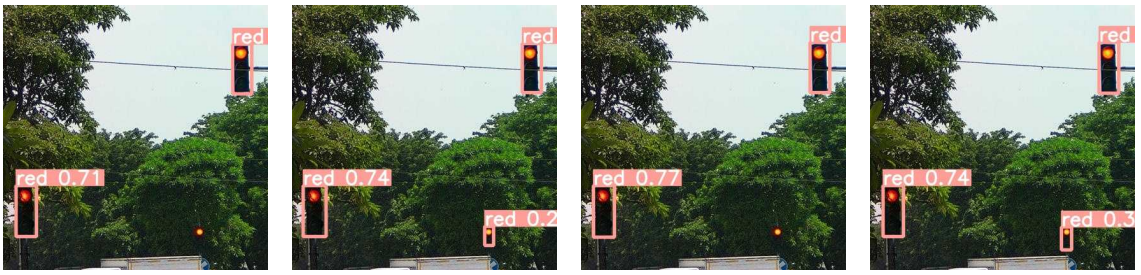
3.1.5 YOLOv8x

```
Speed: 0.5ms pre-process, 57.9ms inference, 7.8ms postprocess per image at shape (1, 3, 640, 640)
```

YOLOv8x test result

test data에 대해 prediction을 진행한 결과, 추론에 (1, 3, 640, 640) 이미지 당 66.2ms의 시간이 걸렸으며 예측 결과는 위와 같았다.

3.2 Model Comparison



YOLOv8s YOLOv8m YOLOv8l YOLOv8x

predict 결과를 보면 YOLOv8 s와 l 모델의 첫번째 사진을 제외하고는 전부 바른 prediction을 한 것으로 보인다.

4. Conclusion

4.1 Insight

validation result와 predict result를 확인한 결과 YOLOv8의 m 모델이 자율주행자동차에 활용하기 가장 적합한 모델인 것으로 보인다.

4.2 Futhermore

해외 차량용 신호등과 국내 차량용 신호등의 가장 큰 차이는 신호의 나열이다. 본 프로젝트는 미국의 T 회사를 위한 AI 모델의 개발에는 세로로 나열된 신호등 데이터를 활용해 학습을 진행했다. 이렇게 학습한 모델이 가로로 나열된 국내 신호등 영상에도 개발한 모델이 적용될 수 있을까에 적용해보았다.

기존 모델 적용 270도 회전 후 학습한 모델 적용

기존 모델을 그대로 적용한 결과, 어떤 신호등도 predict하지 못했다. 하지만 한국 신호등과 같이 가로 방향(빨-노-초)으로 나열될 수 있도록 학습 데이터를 시계방향으로 270도 회전 시킨 후 학습을 진행한 모델을 적용하자 신호등이 predict됐다. 하지만 해외 차량용 신호등 test 결과와 비교해 전체적으로 낮은 confidence의 값을 보였으며 감지되지 않는 신호등이 존재하는 등의 문제가 발생했다.

국내 차량용 신호등 감지 AI 모델의 개발을 위해서는 이미 학습된 해외의 AI 모델을 그대로 활용하는 것이 아니라 국내 차량용 신호등 이미지 데이터를 활용한 학습이 필요한 것으로 보인다.