
2024년 빅데이터 기반 AI 개발 전문가 「딥러닝 기반 차량번호 인식」 포트폴리오

2024. 7.

목 차

I.개요	1
1. 제안 배경	1
2. 목적(필요성)	1
II. 기술 및 구성	2
1. 아이템의 기술	2
2. 개발기술의 사용성	3
3. 프로젝트 구성도	3
III. 프로젝트 목표	4
1. 정성적 목표	4
2. 정량적 목표	4
IV. 개발내용	5
1. 순서도	5
V. 추진일정 및 진행과정	6
1. 학습 데이터 수집	6
2. 번호판 Label 작업	7
3. 번호판 훈련작업	8
4. 번호판 번호 label 및 훈련	11
5. 데이터 전송 및 저장	12
6. 센터프로그램	13
VI. 성능/효과	13
VII. 자원소요 계획	14
1. 인력 구성	14
IX. 기타	14
1. 데이터 정보	14
*. 별첨	15

I. 개요

1. 제안 배경



<Fig 01. 범죄 발생 추이 현황>



<Fig 02. 2023년 _ 전체 사건의 73% 해결한 제주도 CCTV>

<Fig 01>을 통해 2022년도까지의 범죄 발생 추이를 살펴보면, 전체 발생 건수는 2017년 대비 11% 감소했지만, 여전히 148만 건으로 많은 범죄가 발생하고 있음을 알 수 있음. 이러한 범죄를 해결하기 위해 많은 CCTV가 활용되고 있으며, **제주도의 경우 <Fig 02>와 같이 2023년 CCTV 1만 7000여 대 실시간 관제를 통해 얻은 영상정보 73%가 범인 검거 및 사건 해결에 결정적인 역할**을 한 것으로 조사됨.

이러한 사례들을 통해 CCTV는 범죄 해결에 중요한 열쇠임을 알 수 있으며, CCTV와 같이 영상을 활용해 **범죄 예방 및 검거**에 도움이 되는 차량 번호판 인식 프로젝트를 제안하게 됨.

2. 목적(필요성)

본 프로젝트는 차량 번호판을 **정확하게 인식하는 시스템**을 개발하는 것을 주요 목적으로 하고 있음. 이는 범죄 예방 및 해결에 있어 중요한 첫 단계로서 다음과 같은 필요성을 가짐.



<Fig 03. How AI helped catch a drug trafficker>



<Fig 04. 24시간 빈틈없는 경남재난안전상황실 운영>

1). 범죄 추적 용이

차량 번호판 인식을 통해 **범죄 추적이 용이**해짐. 실제로 미국에서는 인공지능 시스템이 마약 밀매범의 의심스러운 운전 습관을 분석하여 마약 밀매범을 식별하고 체포한 사례가 있음 (참조: Fig 03). 이러한 사례를 통해 범죄 예방 및 검거 효율성 향상이 필요함을 확인할 수 있음.

2). 신속한 대응

<Fig 04>와 같은 관제 상황실의 빈틈없는 실시간 모니터링은 범죄 발생 시 **신속한 대응**이 가능하게 함. 이를 통해 범죄뿐만 아니라 재난 및 교통사고와 같은 상황에서도 안전을 개선할 수 있음.

이와 같이 차량 번호판 인식 프로젝트는 범죄 예방 및 검거, 교통 관리 등 다양한 목적을 통해 사회 전반의 **안전과 효율성을 크게 향상**하게 시킬 수 있음..

II. 기술 및 구성

1. 아이템의 기술

항목	내용		
프레임워크	YOLOv5	기술 특징	  
하드웨어 사양	<div></div> <div>Core(TM) i5-13400 2.50 GHz32.0GB(31.7GB 사용 가능)Windows 11 ProNVIDIA GeForce RTX 3060</div>		
소프트웨어 및 환경	<div><pre>(base) C:\Users\Administrator> (yolo) C:\Users\Administrato (yolo) C:\Users\Administrator> Python 3.9.19 Python 3.9.19 CUDA Version: 12.4</pre><div>Python & CUDA 버전</div></div> <div><pre>(yolo) C:\Users\Administrato Python 3.9.19 (main, May 6 Type "help", "copyright", "c >>> import torch >>> print(torch.__version__) 2.3.0</pre><div>Torch 버전</div></div>		
모델 구성	<div></div> <div>1. YOLOv5 : car_plate(번호판인식)</div> <div><pre>p.stem 23조 3779 p.stem 24마 4141 global_list 23조 3779 global_list 24마 4141</pre><div>2. YOLOv5 : car_number(번호&문자인식)</div></div>		
데이터셋	 <div>이와 같은 사진 1,613장 사용</div>	 <div>이와 같은 사진 1,449장 사용</div>	
데이터셋 관리	 <div>이미지전송</div>	 <div>DataBase</div>	
훈련 파라미터	<pre>(yolo) C:\Users\Administrator>python trian.py --img 416 --batch 16 --epochs 300 --data data/coco128.yaml --cfg yolov5s.yaml --weights yolov5x.pt</pre>		
학습 파일 설정	yaml 파일: YOLOv5s 기준	Worker 수: 8	
YOLOv5 요구사항	*별첨		

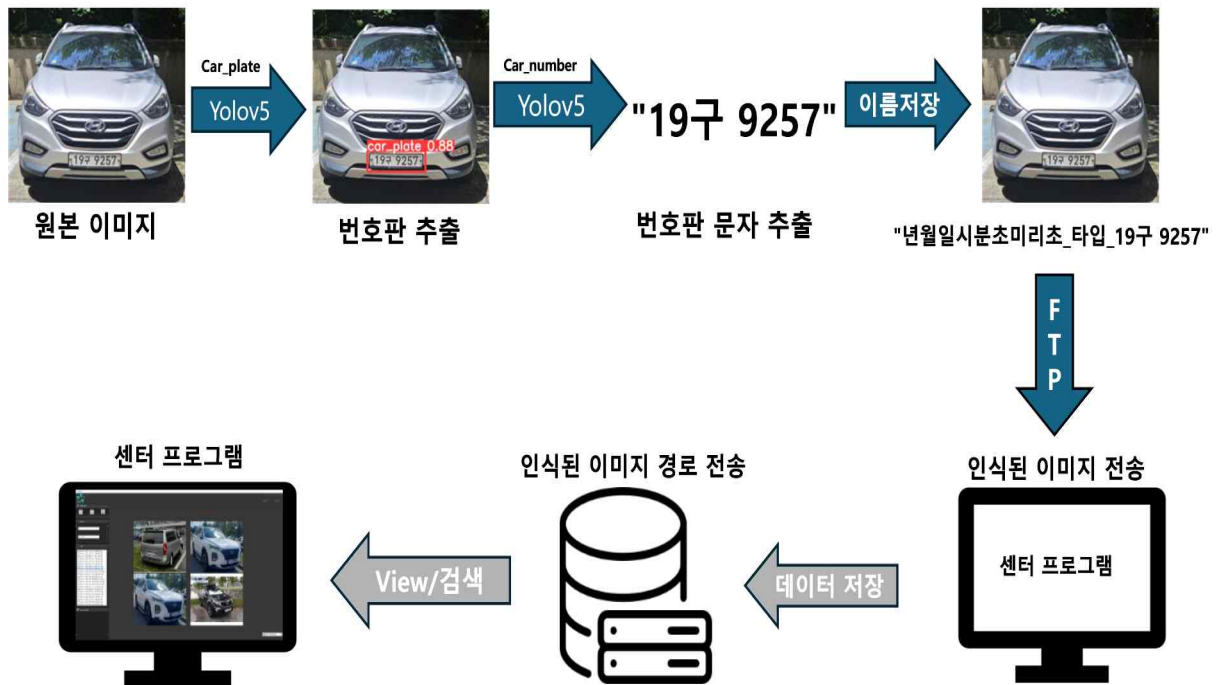
2. 개발기술의 사용성

□ Yolo 분석을 통한 객체 종류 분류 후 최적 알고리즘 선별 프로그램

딥러닝 구분	기본 사용	부가적 사용
차량번호판 인식	CNN	KNN - 24% CNN - 65% Yolov5 - 90%
차량번호 인식	숫자 문자인식	KNN - 20% CNN - 87% Yolov5 - 95%

KNN은 데이터 스케일링과 차원 축소와 같은 데이터 전처리과정에 어려움을 느꼈고, CNN의 경우 이미지 분류에 강점이 있지만, 차량번호판 인식에서는 65%, 차량번호 인식에는 87%로 객체 탐지 및 위치 추정 작업에 추가적인 처리가 필요함. 따라서 빠른 속도와 실시간 객체 인식이 가능한 YOLOv5로 결정.

3. 번호판 인식 프로그램 구성도

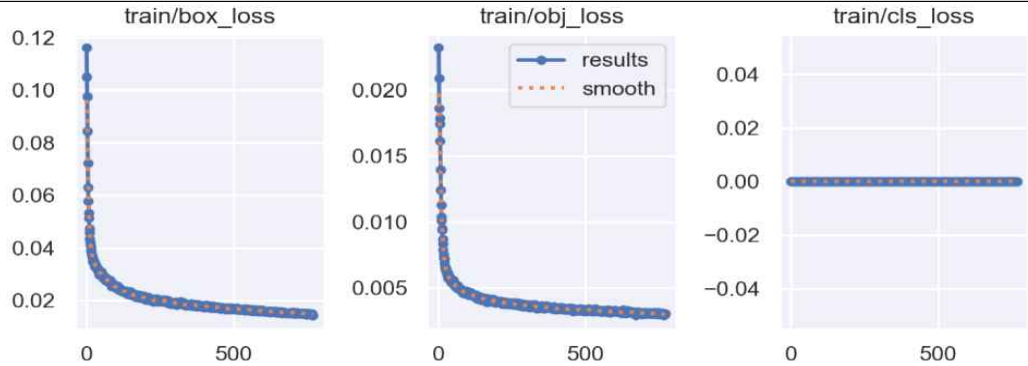
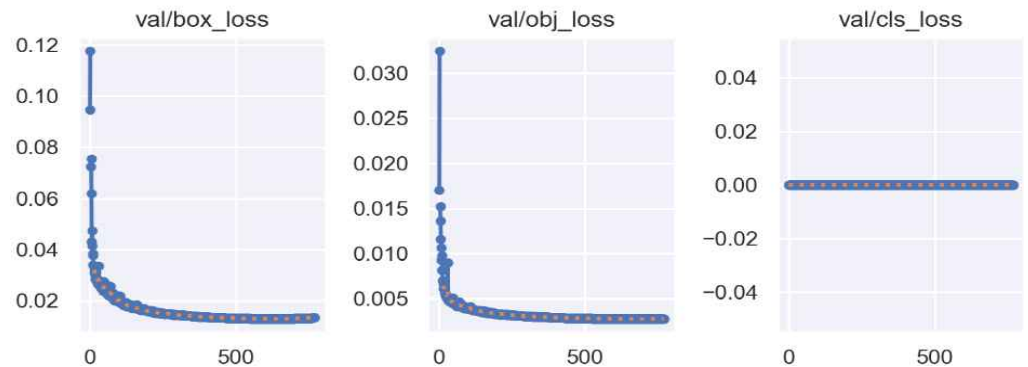


<Fig 05. 번호판 인식 프로그램 구성도>

차량 원본 이미지에서 번호판 및 문자를 인식해 "년월일시분초미리초__(이미지)타입_차량번호"파일로 저장하고 FTP를 통해 전송하여 데이터베이스에 저장한 다음 센터 프로그램에서 조회 및 검색할 수 있도록 하는 시스템 구성

Ⅲ. 프로젝트 목표

1. 정성적 목표

항목	내용
정확성	 <p><fig 06. 훈련 데이터셋 결과 그래프></p>  <p><fig 07. 검증 데이터셋 결과 그래프></p> <p>훈련 데이터셋(train)과 검증 데이터셋(val)의 그래프 모양이 유사하게 유지되며, 모델이 훈련 데이터에 과적합 되지 않음을 보장하고, 실제 환경에서도 높은 인식 정확성 유지 목표</p>

2. 성능 목표/정량적 목표

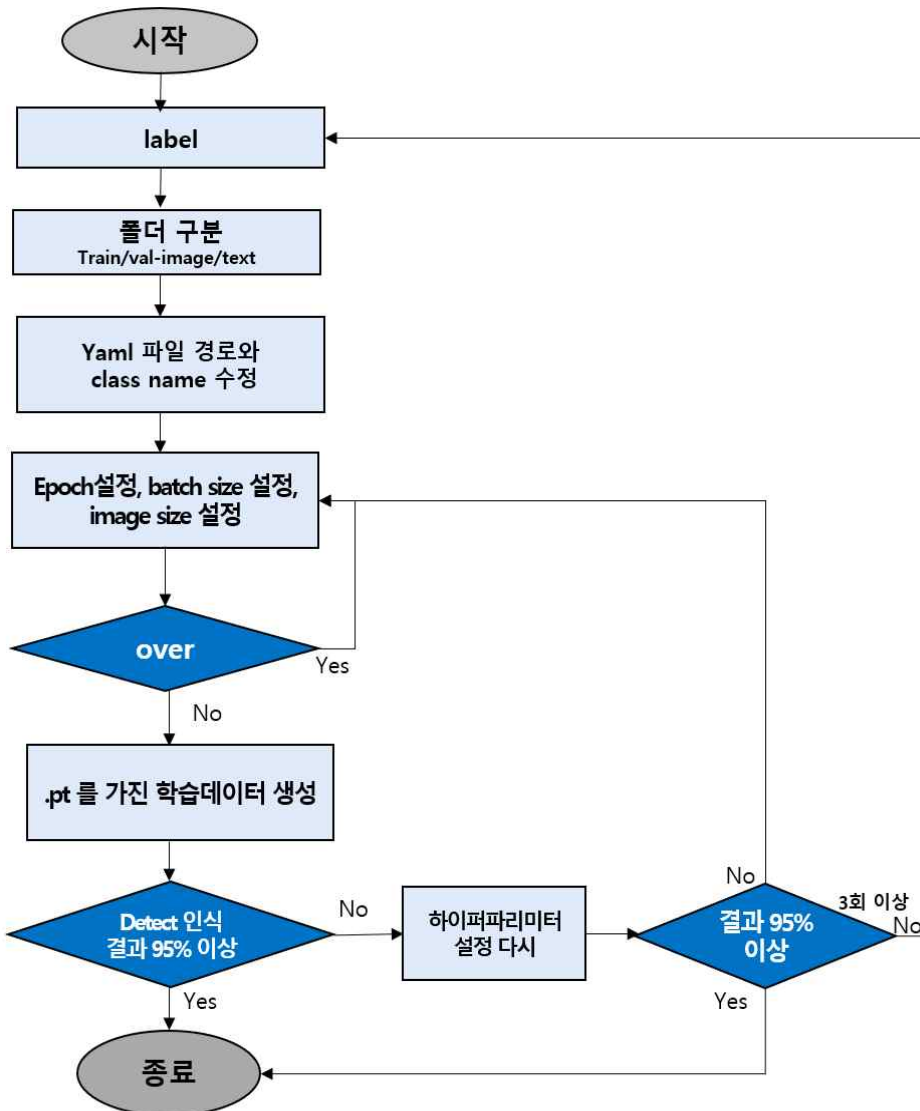
주요 성능지표	단위	개발목표	학습횟수	*개발결과		측정대상
				훈련	신규	
차량번호판	%	95%	4회	99%	91%	자체 검사
차량번호 (문자 및 숫자)	%	95%	3회	95%	97%	육안 검사
프로그램 오류 비율	%	오인식 5%	car_plate	0.1%	0.7%	자체 검사
			car_number	4.8%	2.1%	
	%	미인식 5%	car_plate	0.3%	8%	자체 검사
			car_number	0%	0%	
센터에 인식된 이미지 view	%	100%	-	100%		외부 3자 입회하에 검수
FTP통신 미전송	%	1%	-	1%		시간을 정해서 폴더 내의 이미지 수를 측정

* 목표정확도는 95% 선정하였지만 시간이 부족해지면 최소 80~85%

* 개발결과 *별첨2 참고

IV. 개발내용

1. 학습 순서도



<학습 순서도>

합이면 다시 하이퍼파라미터를과정에서 다시 설정해서 훈련시키고 과적합이 아닌 학습된 모델을 pt파일로 생성함. 여기서 나온 pt파일을 best.pt라고 부르면 이 pt파일을 이용해 Detect를 돌려서 Test 이미지에 어느 정도로 인식하는지 확인. 여기서 인식률이 95%가 안 넘으면 Train 설정 즉 하이퍼파라미터 설정을 다시 작업함. 95%가 안 넘으면 3회 이상 결과가 조건을 넘지 않는다면 추가로 데이터를 더 수집해 label 작업을 다시 진행. 95% 넘으면 종료 조건에 도달해서 이 프로세스를 종료함. 종료 조건에 도달하지 않았다면 학습을 계속 진행함.

이 순서도는 모델 학습 및 평가 과정에서 반복적인 학습과 평가를 통해 일정 기준(95% 성능)을 만족할 때까지 과정을 반복하는 것을 나타냄.

이 순서도는 프로젝트 모델의 학습 및 테스트 과정을 설명함.

프로세스를 시작하면 먼저 데이터에 라벨을 지정하는 작업을 함. 작업 완료 후 나온 txt 파일을 라벨 작업한 이미지와 함께 훈련, 검증 폴더에 미리 정한 훈련 비율대로 폴더에 나눠 넣어서 훈련을 준비함.

그리고 YAML 파일에 훈련, 검증 데이터가 있는 경로와 훈련할 class 이름을 수정. 학습할 모델의 하이퍼파라미터 튜닝 epoch 수, batch size, 이미지 크기를 설정함.

모델이 과적합인지 아닌지 확인하여 과적

V. 추진일정 및 진행과정

세부내용	W1			W2			W3			비 고
1. 학습 데이터 Set 수집										
2. 번호판 & 번호 Label 작업										
3. 번호판 & 번호 학습 진행										
4. 인식률 95% 성능 테스트										
5. 센터프로그램 기획 및 구현										
6. 최종 확인 및 서류작업										

1. 학습 데이터 수집

1) AI를 활용한 이미지생성

Karlo 2.0을 사용하여 번호판이 보이는 차량 이미지를 생성하려 했으나, 각도와 이미지 품질 등 **편파적인 이미지로** 생성이 되고 중요한 번호판이 생성되지 못하는 점에서 실사 이미지를 사용하는 것이 더 효과적이라는 결론 도출



<Fig 08. Karlo 2.0에서 생성한 차량이미지1>



<Fig 09. Karlo 2.0에서 생성한 차량이미지2>

2) RTSP 데이터 사용

RTSP를 통해 수집한 데이터는 **화질이 너무 낮아** 번호판을 인식하기 어려움, 화질이 좋은 데이터를 확보하기 위해 **유튜브의 주행 영상 및 실제 사진** 활용 결정



<Fig 10. 고창담양고속도로 캡처본1>



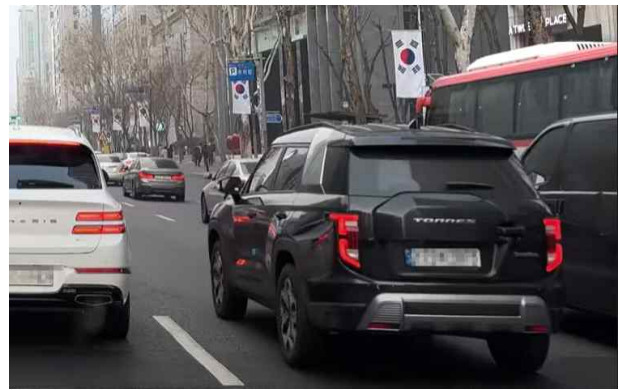
<Fig 11. 남해고속도로 캡처본2>

3) 실제 데이터 수집

- 유튜브 주행 영상 캡처 : 다양한 주행 환경과 각도에서 번호판을 인식할 수 있도록 유튜브에 업로드된 **고화질 주행 영상**을 캡처
 - 주변 차량 촬영 : **실제 도로**에 있는 차량을 촬영하여 다양한 각도와 조명 조건에서의 데이터 확보
- > 이를 통해 더욱 정밀한 번호판 인식을 학습할 수 있음



<Fig 12. 주변 도로 촬영 사진>



<Fig 13. 유튜브 주행 영상 캡처본>

2. 번호판 Label 작업

실제 범죄 해결에서는 **CCTV 영상의 낮은 화질**에서 번호판을 인식해야 하는 경우가 많음. 따라서 프로젝트의 현실성을 높이기 위해 낮은 화질의 데이터를 보완하는 **해상도 조정**과 정면이 아닌 측면으로 치우친 번호판에 대비한 **전단 변환 기법**을 활용하기로 결정.

→ 이를 통해 모델이 고화질 데이터뿐만 아니라 다양한 화질의 영상에서도 **번호판을 정확히** 인식할 수 있도록 할 것.

1) 데이터 해상도 조정

수집된 데이터를 라벨링하는 과정에서 **픽셀 단위**로 라벨링 결과가 **달라진다**는 사실을 알게 됨. 데이터 수집 단축과 양 증대를 위해 이미지 해상도를 조정하여 데이터 확보 후 라벨링 진행. 여러 해상도 조정을 통해 효율적인 라벨링 수행 및 모델의 성능을 향상시키는 데 도움이 됨



<Fig 14. 원본 이미지>



<Fig 15. 1500px 조정>



<Fig 16. 크기 50% 조정>



<Fig 17. 크기 25% 조정>

2) 전단 변환 기법

다양한 각도로 회전하여 데이터의 다양성을 확보함. **전단 변환 기법**을 통해 모델은 여러 각도에서 찍힌 다양한 차량 번호판 상황에서 더 나은 성능을 발휘할 수 있음. 인식률이 잘 나오는 **정위치가 아닌** 다양한 각도에서도 **인식할** 수 있도록 도와줌.



<Fig 18. 원본 이미지>



<Fig 19. 데이터 전단 변환 1>



<Fig 20. 데이터 전단 변환 2>

3. 번호판 훈련작업

1). 번호판 인식 오류

기존 **500번**의 epochs로 훈련한 결과, 이미지 해상도가 변하면 인식 대상이 아닌 것을 번호판으로 잘못 인식하는 경우가 발생함. 이러한 결과는 epochs 값이 너무 낮아서 발생한 것으로 판단됨. 따라서 훈련 시 epochs 값을 500 이상으로 설정하여 성능을 **향상**시킬 계획.



<Fig 21. epochs 500, 픽셀 500x413>



<Fig 22. epochs 500, 픽셀 1,500x1,070>

2). 하이퍼파라미터 튜닝 설정

기본 설정 : `--img 416 --batch 16 --cfg yolov5s.yaml --weights yolov5x.pt`

2-1) 500 epochs:

- **사진 해상도 및 비율**에 따라 번호판 인식 차이가 있음



<Fig 23. epochs 500, 비율 25%>



<Fig 24. epochs 500, 비율 50%>

2-2) 1,000 epochs:

- 500 epochs 미인식 번호판을 잡음
- 하지만 500 epochs가 잡은 번호판을 인식 못 함 -> **과적합** 발생



<Fig 25. epochs 1,000, 번호판 미인식>



<Fig 26. epochs 500, 번호판 인식>

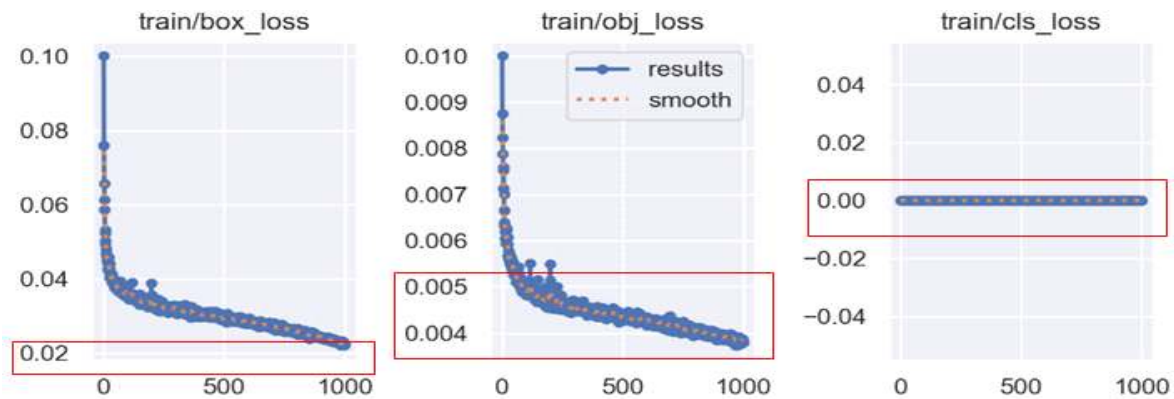
2-4) 1,200 에포크:

- 추가 에포크로 성능 **향상이 거의 없음**을 확인.

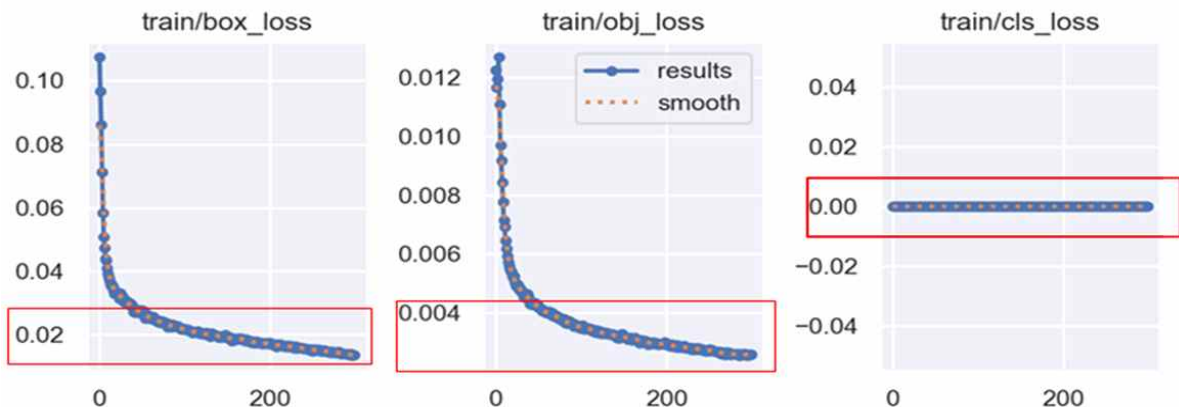
2-5) 최종 결정: epochs 300

```
python train.py --img 416 --batch 16 epochs 300 --data data/coco128.yaml --cfg yolov5s.yaml --weights yolov5x.pt
```

- 에포크가 높으면 인식률이 좋을 거로 생각했는데 1,000 에포크 모델을 보면서 과적합을 의심하게 되며 **epochs와 데이터셋 이미지 수 비율이 맞아야** 한다는 것을 배움
- 차량번호판 인식률 **95%** 달성
- 따라서 최종 에포크는 300으로 결정하고, **후처리**하기로 결정



<Fig 26. epochs 1,000, train 그래프>



<Fig 27. epochs 500, train 그래프>

성능 지표로는 box, obj, cls 손실률이 다음 3가지 조건을 모두 충족하지 않으면 사용할 필요가 없음

1.train/box_loss: 0.02미만

2. train/obj_loss: 0.005미만

3. train.cls_loss: 0

따라서 손실률이 더 낮은 epochs 300을 최종 모델로 선정

3). epochs 조정 후에도 발생하는 번호판 인식 오류: 후처리 방안 고려

번호판 인식 프로젝트에서 잘못된 대상이나 희미한 번호판을 처리하는 두 가지 접근 방법

가. 모든 대상을 가져와 후처리하는 방법

나. 처음부터 인식률 기준을 높게 설정한 후 후처리하는 방법

프로젝트는 후자를 선택함. 이 선택은 **코드를 간결화**하고 유지보수성을 높이며, **처리 시간**을 단축하여 실시간 시스템의 성능을 향상시키고 자원을 효율적으로 활용할 수 있게 함. 또한 오류를 줄여 **최종 인식 결과의 정확성을 향상**시킴.

3-1) 인식률 결정(conf-thres/iou-thres)

인식률을 결정하기 위해 인식률 기존 설정인 **25%에서 순차적으로** 올려서 최종 범위를 결정하기로 함.



<Fig 28. conf-thres 0.25>



<Fig 29. conf-thres 0.5>



<Fig 28. conf-thres 0.7>

번호판이 아닌 다른 객체도 인식	
멀어서 인식하기 힘든 번호판 인식o	인식x

최종적으로 인식률과 정확성 사이의 균형을 잘 맞추는 **80%로** 결정

3-2) 바운딩 박스 후처리

위에서 인식률 범위를 80%로 설정했지만 잡은 번호판에서 너무 작은 crop 이미지는 번호 판임에도 불구하고 **숫자를 인식하기 어렵거나 반 잘린 번호판**이라 후처리하기로 결정.



<Fig 29. 잘린 번호판 인식한 detect>



<Fig 30. detect crop 이미지 2개>



<Fig 31. 후처리 crop 이미지>

보통 번호판은 너비가 높이의 2배 이상이지만, 잘린 번호판은 2배 미만인 경우가 많음. 따라서 바운딩 박스의 좌표를 추출하여 너비와 높이를 계산하고, **너비와 높이를 이용해 종횡비를 구한 뒤 종횡비가 2보다 큰 경우만 crop 이미지로 저장하기로 함.** <Fig 31>를 통해 실제 종횡비를 계산한 후처리 결과를 알 수 있음

4). 훈련결과

구분	인식률	인식데이터 수
훈련데이터	99%	1605/1613
신규데이터	91%	1028/1128

번호판 인식률 범위를 80%로 잡아서 훈련데이터에서 80% 미만인 번호판 8개는 못 잡았지만 번호판이 아닌 다른 범위를 잡은 detect 결과는 나오지 않아서 번호판 번호 작업 진행하기로 결정

4. 번호판 번호 label 및 훈련

1) 번호판 데이터 수

항목	차량번호판	차량번호
데이터셋		
	<Fig 32. 차량번호판 1,613장 사용>	<Fig 32. 차량번호 1,449장 사용>

데이터 수가 다른 이유는 프로젝트 설정한 기간 더 많은 라벨링 작업을 하기에는 **시간이 촉박**하여 시간 내에 라벨링 작업을 완료한 데이터만 사용 → 추후 **증가할 계획**

2) 번호판 번호 및 문자 label 기준

위 번호판 detect 결과에서 나온 crop 이미지들을 대상으로 문자 labeling을 진행함

- 번호 : 0~9
- 문자 : <fig 33. 한글 기준표>참조

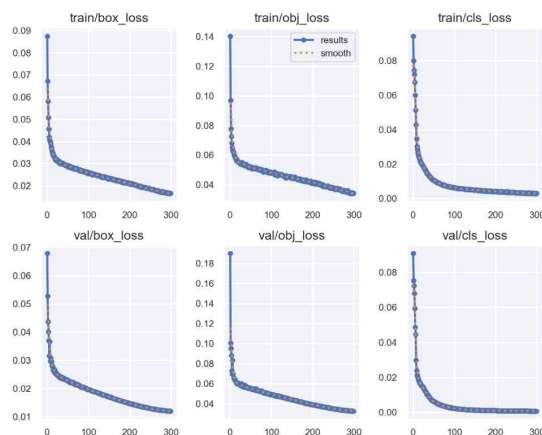
가	a1	거	a6	고	a15	구	a24	아	b1	배	c1	하	d1
나	a2	너	a7	노	a16	누	a25	바	b2			허	d2
다	a3	더	a8	도	a17	두	a26	사	b3			호	d3
라	a4	러	a9	로	a18	루	a27	자	b4				
마	a5	머	a10	모	a19	무	a28						
		버	a11	보	a20	부	a29						
		서	a12	소	a21	수	a30						
		어	a13	오	a22	우	a31						
		저	a14	조	a23	주	a32						

<Fig 33. 한글 기준표>

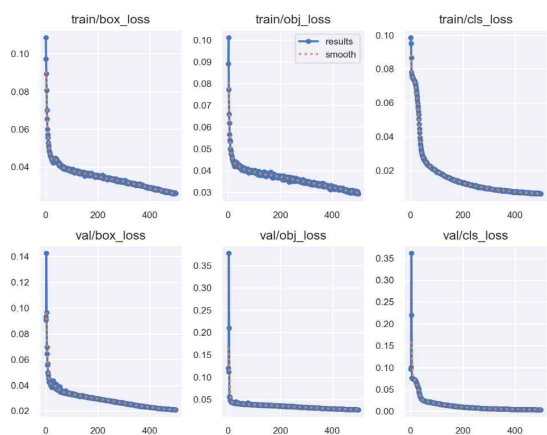
3) 훈련 방법

번호판 인식 모델과 같은 방식으로 img size, batch, epochs 결정

```
--img 416 --batch 16 --cfg yolov5s.yaml --weights yolov5x.pt
```



<Fig 34. epochs 300 성능 그래프>



<Fig 35. epochs 500 성능 그래프>

훈련 데이터셋(train)과 검증 데이터셋(val)의 **유사한 그래프 모양**을 통해 훈련 데이터와 새로운 데이터 훈련 시 **큰 차이가 없다**는 것을 알 수 있고, 훈련 손실 기준도 충족하여 두 epochs 중 효율적으로 훈련할 수 있는 **epochs 300** 선택

4) 훈련결과

구분	인식률	인식데이터 수
훈련데이터	95%	1378/1449
신규데이터	97%	1006/1028

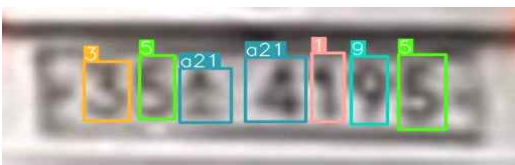
4-1)오인식



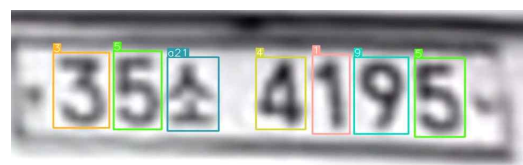
<Fig 36. "5" 오인식 이미지>



<Fig 38. "5" 오인식 이미지 >



<Fig 38. "4" 오인식 이미지>



<Fig 39. 정확한 인식 이미지>

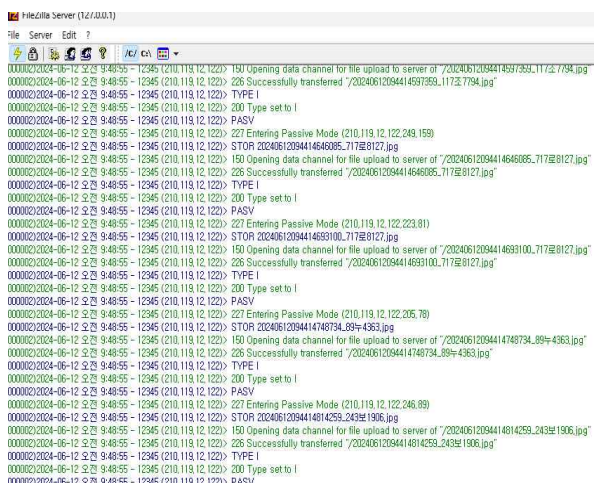
현재 모델은 픽셀 단위가 변할 때 숫자 및 문자 인식에 일부 어려움을 겪고 있음. 그러나 모든 범위에서 인식이 불가능한 것은 아니며, 일부 숫자나 문자만 인식하지 못하는 수준.

따라서 현재 모델은 전체적으로 괜찮은 성능을 보이므로 데이터를 늘려서 모델을 다시 훈련하는 대신 일부 오인식 오류를 감수하고 진행하기로 결정.

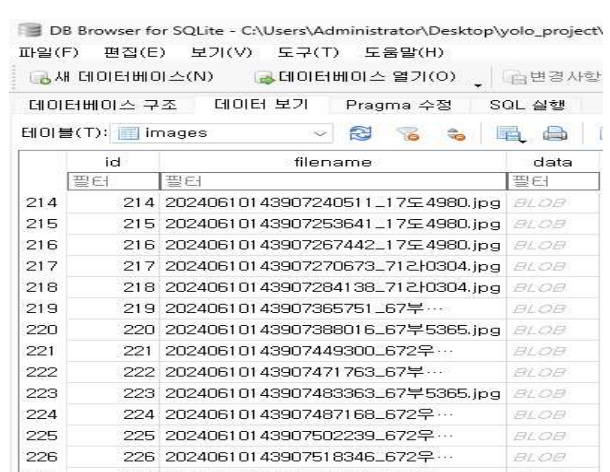
5. 데이터 전송 및 저장

차량 번호판의 문자와 숫자가 인식된 사진을 센터 프로그램에 전송. 먼저 사진의 파일 이름을 "년월일시분초미리초_타입_차량번호" 형식으로 변경하고, crop된 이미지나 detect로 도출된 이미지들과 같은 불필요한 이미지는 삭제하기로 함. 이미지타입은 crop된 이미지는 c로 원본 이미지는 o로 구분하여 저장함

이후, FTP 통신을 이용하여 센터 프로그램에 이미지를 전송함. 받은 이미지의 경로는 센터 프로그램에서 데이터베이스에 저장되며 이러한 과정을 통해 데이터 전송 및 저장이 완료됨.



<Fig 40. FTP 데이터 전송>



<Fig 41. DB 데이터 저장>

6. 센터프로그램

6-1) 사용된 데이터

데이터베이스에 저장된 이미지 경로를 센터프로그램 리스트뷰(ListView)에 연결해서 사용.

6-2) 기능

기능	실제 동작화면	
화면분할 기능		
	<Fig 42. 메인 화면>	<Fig 43. 1분할 화면>
		
	<Fig 44. 2분할 화면>	<Fig 45. 4분할 화면>
검색 기능		
	<Fig 46. 번호 검색>	<Fig 47. 날짜 검색>
동일한 차량번호 표시		
	<Fig 48. 차량번호 동일한 crop이미지>	<Fig 49. 차량번호 동일한 원본이미지>

VI. 성능/효과

• 적용 범위 확대

다양한 해상도 조정과 각도 변환을 통해 다양한 상황에서의 인식 성능을 확보함으로써, 교통 단속, 주차 관리, 범죄 수사 등 다양한 분야에 적용할 수 있는 범용적인 모델 구축.

VIII. 자원소요 계획





1. 인력 구성

순번	시나리오	비고
1	차량번호 인식 시스템 시나리오 작성	최호진, 최균호
2	YoLo Dataset 모음	최호진, 최균호
3	차량번호판 Crop 라벨링 진행	최호진, 최균호
4	Train.py 학습 진행	최호진, 최균호
5	detect.py로 결과 확인	최호진, 최균호
6	Crop된 차량번호판 문자 라벨링 진행	최호진, 최균호
7	Train.py 학습 진행	최호진, 최균호
8	detect.py로 결과 도출	최호진, 최균호
9	인식된 이미지 FTP로 센터에 전송	최호진, 최균호
10	인식된 이미지 경로 DB에 전송	최호진, 최균호
11	센터프로그램 구현(View/검색)	최호진, 최균호
12	검색 조건 설정 기능 구현	최호진, 최균호
13	차량번호 앞/뒤 /날짜 입력으로 이미지 리스트 표출	최호진, 최균호
14	차량이미지와 번호이미지 동시 표출	최호진, 최균호

※ 기여도에 따라 굵은 표시로 기재

IX. 기타

1. 데이터 정보

데이터 수집	유튜브 영상 캡처, 실제 사진	데이터 활용	car_plate , car_number
데이터 편집	    알집 픽셀 변환		데이터 해상도 조정, 데이터 증강

*. 별첨01. YOLOv5 요구사항

```
# YOLOv5 requirements
# Usage: pip install -r requirements.txt
# Base -----
gitpython>=3.1.30
matplotlib>=3.3
numpy>=1.23.5
opencv-python>=4.1.1
pillow>=10.3.0
psutil # system resources
PyYAML>=5.3.1
requests>=2.23.0
scipy>=1.4.1
thop>=0.1.1 # FLOPs computation
torch>=1.8.0 # see https://pytorch.org/get-started/locally (recommended)
torchvision>=0.9.0
tqdm>=4.64.0
ultralytics>=8.0.232
# protobuf<=3.20.1 # https://github.com/ultralytics/yolov5/issues/8012
# Logging -----
# tensorboard>=2.4.1
# clearml>=1.2.0
# comet
# Plotting -----
pandas>=1.1.4
seaborn>=0.11.0
# Export -----
# coremltools>=6.0 # CoreML export
# onnx>=1.10.0 # ONNX export
# onnx-simplifier>=0.4.1 # ONNX simplifier
# nvidia-pyindex # TensorRT export
# nvidia-tensorrt # TensorRT export
# scikit-learn<=1.1.2 # CoreML quantization
# tensorflow>=2.4.0,<=2.13.1 # TF exports (-cpu, -aarch64, -macos)
# tensorflowjs>=3.9.0 # TF.js export
# openvino-dev>=2023.0 # OpenVINO export
# Deploy -----
setuptools>=65.5.1 # Snyk vulnerability fix
# tritonclient[all]~2.24.0
# Extras -----
# ipython # interactive notebook
# mss # screenshots
# albumentations>=1.0.3
# pycocotools>=2.0.6 # COCO mAP
wheel>=0.38.0 # not directly required, pinned by Snyk to avoid a vulnerability
```

***. 별첨02. 훈련결과**

1. car_plate 훈련결과

구분	훈련데이터	신규데이터	훈련데이터		신규데이터	
			미인식	오인식	미인식	오인식
인식률	99%	91%	0.3%	0.1%	8%	0.7%
인식데이터 수	1605/1613	1028/1128	6/1613	2/1613	91/1128	9/1128

2. car_number 훈련결과

구분	훈련데이터	신규데이터	훈련데이터		신규데이터	
			미인식	오인식	미인식	오인식
인식률	95%	97%	0%	4.8%	0%	2.1%
인식데이터 수	1378/1449	1006/1028	0/1449	71/1449	0/1028	22/1028