모델 보고서

JJJ

나승주, 민경준, 장수연

목차

1. 프로젝트 개요

가. 프로젝트 목적

산업현장 안전관리 시스템 구현

나. 데이터셋

crop	50%
rotate90	clockwise, counter-cw, upside down
rotation	15
blur	2px
cutout	percent 13% count 4
brightness	25%
grayscale	25%
hue	25%

아래의 모든 데이터는 위의 표에 나와있는 요소들을 적용하여 데이터를 변형하였고, raw data에서 3배로 증강하여 최종 데이터를 만들었다. 각 데이터의 개수와 특징은 아래와 같다.

	데이터①	데이터②	데이터③	데이터④	데이터⑤
이미지 개수(개)	5,557	7,172	318	923	186

1) 데이터① 특징

- 이미지 개수는 헬멧, 고글, 조끼 순으로 많다.
- 작업자들이 보호구를 착용하고 있는 사진으로 주로 구성되어 있으며, 각 보호구만 따로 촬영된 이미지들도 존재한다.

2) 데이터② 특징

- 헬멧 데이터 중, 노란 헬멧 이미지를 선별하여 만들었다.
- 다양한 고글 이미지를 더하였다.

3) 데이터③ 특징

- 각 경고코드 별로 안전보호구를 입고 직접 촬영한 사진을 라벨링하여 제작한 데이터다.
- 안경 이미지를 새로 추가하였으며, glasses라는 class로 분류하였다.

4) 데이터④ 특징

- 다양한 모자(야구모자, 챙모자 등)와 안경 이미지를 모은 데이터다.

5) 데이터⑤ 특징

- 안전 보호구를 입고 직접 촬영한 사진을 라벨링하여 제작한 데이터다.
- 다양한 모자(야구모자, 챙모자, 비니 등)를 추가하였으며, hat이라는 class로 분류하였다.

다. 예측 및 기대결과

학습시킬 데이터는 실사용 환경과 조건이 비슷하지 않은 사진이라도 탐지 대상이 포함된 사진이라면 모두 좋은 학습 데이터라고 생각하였다. 또한 학습시킨 데이터 양이 많을수록 모델의 성능이 향상될 것이라 예상하였다.

글자 인식 모델, 자세 추정 모델, 객체 탐지 모델을 모두 결합한 알고리즘은 Realtime(30fps)으로 연산을 완료할 것으로 예측하였다. 따라서 캠 영상을 웹페이지에 나타내는 동시에 탐지한 보호구에 박스를 그리는 등의 표시를 할 수 있을 것이라 기대하였다.

2. 모델

가. 사용 모델

1) 글자 인식 모델(OCR)

- Tesseract OCR

  구글에서 개발된 무료 오픈소스 OCR(광학 문자 인식) 엔진으로, LSTM 기반의 딥러닝 이용 문자 인식 수행한다. 다양한 언어를 지원하고, 오픈 소스로 공개되어 가장 쉽게 자료를 찾을 수 있어 선택하였다.

- Easy OCR

  JaidedAI에 의해 개발된 오픈 소스 OCR 도구로, pytorch 딥러닝 프레임워크를 기반으로 하여 정확도가 향상된 모델이다. 다양한 형식의 텍스트 인식이 가능하며, 개발자 커뮤니티가 활발한 점을 장점으로 꼽아 선택하였다.

- Paddle OCR

  PaddlePaddle에서 제공하는 오픈소스 OCR 도구다. 이미지 전처리, 텍스트 검출, 텍스트 인식 등 모듈들을 제공해 따로 이미지 전처리가 필요 없다. 정확도가 높으며 경량화된 모델이므로 실시간 처리가 가능하다.

2) 자세 추정(Pose Estimation) 모델

- Openpose

  Skeleton-based 자세 추정 모델로, 신체 위치를 찾기에 유리하다. 이 모델은 PAF(Part Affinity Fields)를 통해 실시간으로 Multi-Person Pose Estimation이 되는 Bottom-up방식의 모델이다. 오픈 소스로 공개된 최초의 실시간 자세 추정 모델이므로 관련 자료가 많아 선택하여 사용해보았다.

- Cascade Classifier

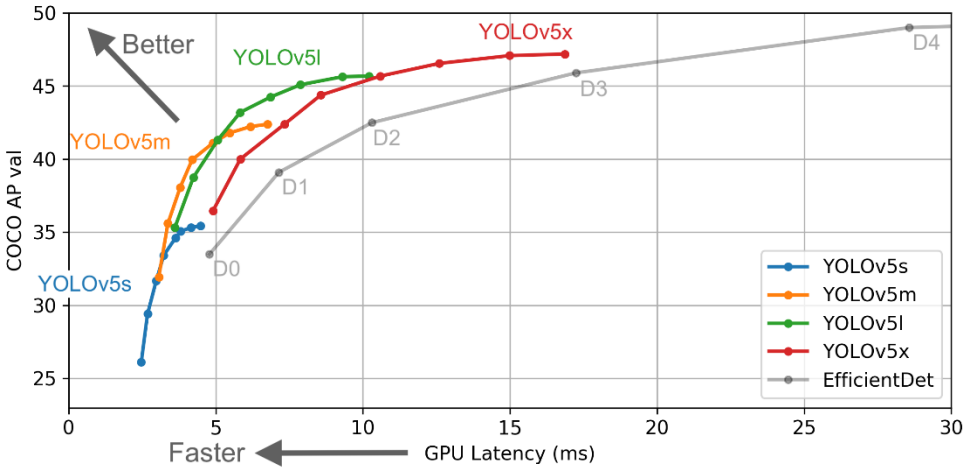
  Haar Cascade는 이미지 분류 및 객체 탐지를 위한 기술 중 하나다. 이미지에서 특정 패턴을 탐지해서 이미지를 분류하거나 객체를 탐지한다. 머신러닝 기반의 기술이므로 속도가 빨라 실시간 탐지에 유리할 것이라 생각하여 활용해보고자 했다.

- MediaPipe

  구글에서 제공하는 AI 프레임워크로서, 영상 데이터를 이용한 다양한 비전 AI 기능을 파이프라인 형태로 손쉽게 사용할 수 있도록 제공된 모델이다. 라이브러리를 불러 사용하듯 간편하게 호출하여 사용할 수 있어 선택하였다.

3) 객체 탐지 모델

YOLO는 one-stage object detection 딥러닝 기법을 사용하는 매우 빠른 속도의 모델이다. 실시간 영상 탐지에 유리할 것이라 생각하여 이 모델을 선정하였다.



- YOLO v5

YOLO v5의 Backbone은 이미지에서 feature map을 추출하는 부분으로 CSPNet기반의 CSP-Darknet을 사용한다. YOLO v5가 이전 YOLO 모델들과 다른 점은, backbone을 depth multiple과 width multiple를 기준으로 하여 크기별로 YOLO v5 s, YOLO v5 m, YOLO v5 l, YOLO v5 x로 나눈다는 점이다.

위의 그래프에서 알 수 있듯이 여러 크기의 모델 중 가장 가벼우면서 빠른 모델 (depth_multiple: 0.33, width_multiple: 0.50)은 YOLO v5 s다. 이 모델은 YOLO v5 중 가장 빠르지만 정확도가 비교적 낮으며, YOLO v5 x는 가장 느리지만 정확도는 높다. 이 프로젝트에서 가장 중요한 점은 정확성과 실시간성임을 염두해두고 YOLO v5 모델을 여러 방법으로 학습시켜 테스트해보았다.

- YOLO v8

YOLO 모델을 위한 완전히 새로운 리포지토리를 이용하는 모델이다. 객체 탐지, 인스턴스 세분화 및 이미지 분류 모델을 학습하기 위한 통합 프레임워크로 구축되어 있다. 이 모델은 앵커박스의 offset 대신에 객체의 중심을 직접 예측하는 앵커프리 모델로, 이전 모델에 비해 속도가 향상되었다.

나. 알고리즘 정의서

1) 출근 및 퇴근 확인 알고리즘

attend app

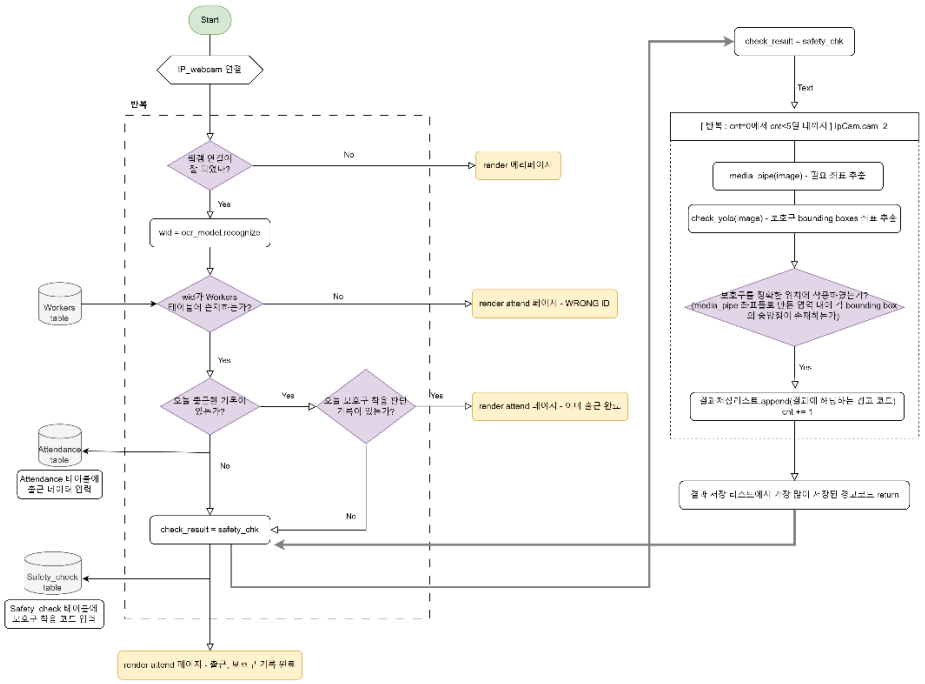
1. check_attendance() : 출근 입력
[input] image path(연결 캠 url 주소)
[output] render attend/webcam.html

1) safety_chk() : 보호구 착용 확인safety_check 테이블 데이터 입력 함수
[input] worker 정보, attendance 정보, image path
[output] IpCam.cam_2 - result

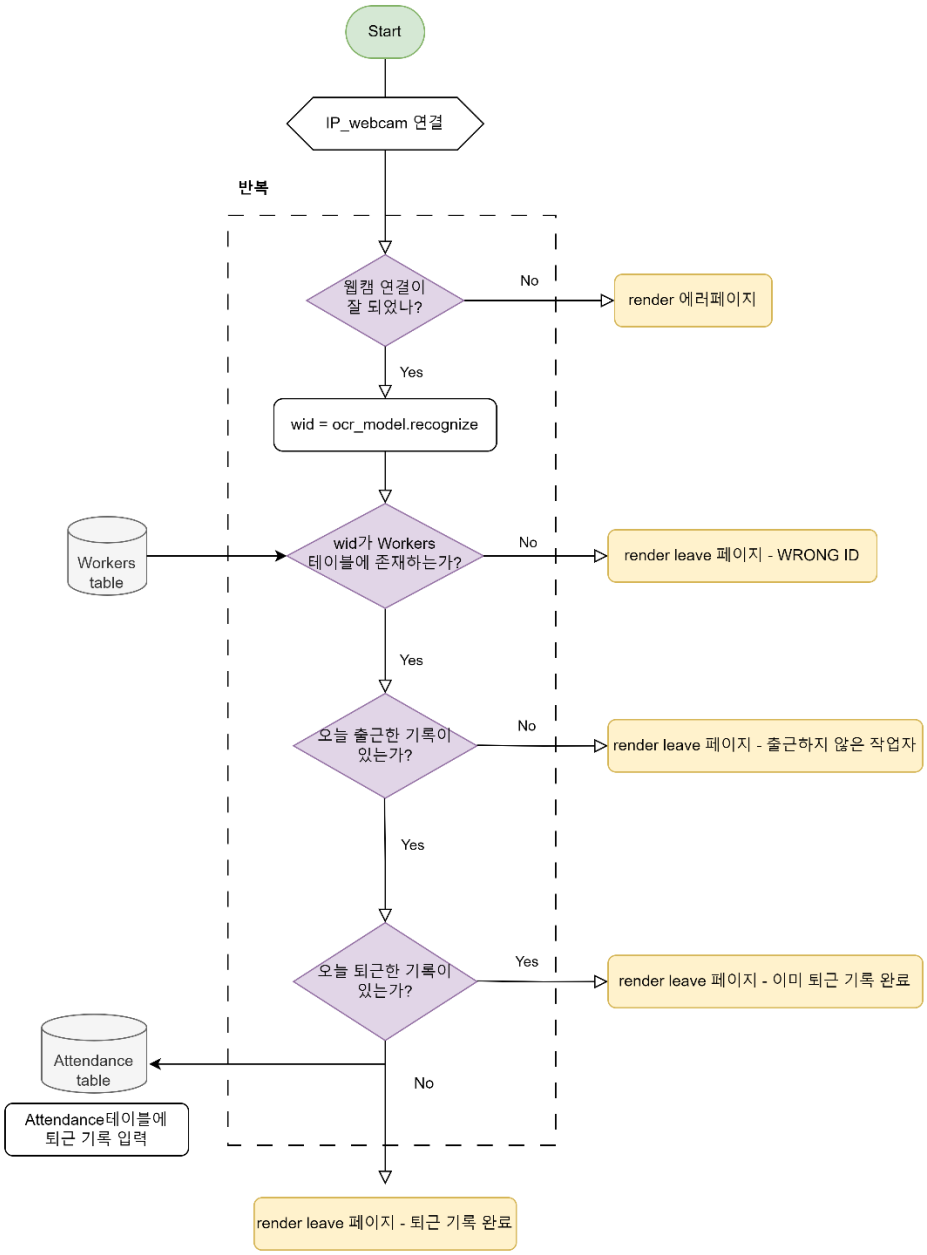
2) IpCam.cam_2() : yolov8n + mediapipe 보호구 착용 확인
[input] image path
[output] 보호구 착용 결과 dictionary 형태로 반환 {"wc" : warning_code, "wr" : wear}

2. check_leave() : 퇴근 입력
[input] image path(연결 캠 url 주소)
[output] render attend/leave.html

- 출근 확인 알고리즘



- 퇴근 확인 알고리즘


Start
IP_webcam 연결
반복
웹캠 연결이 잘 되었나?
No
render 에러페이지
Yes
wid = ocr_model.recognize
Workers table
wid가 Workers 테이블에 존재하는가?
No
render leave 페이지 - WRONG ID
Yes
오늘 출근한 기록이 있는가?
No
render leave 페이지 - 출근하지 않은 작업자
Yes
오늘 퇴근한 기록이 있는가?
Yes
render leave 페이지 - 이미 퇴근 기록 완료
Attendance table
No
Attendance테이블에 퇴근 기록 입력
render leave 페이지 - 퇴근 기록 완료

2) 위험구역 확인 알고리즘

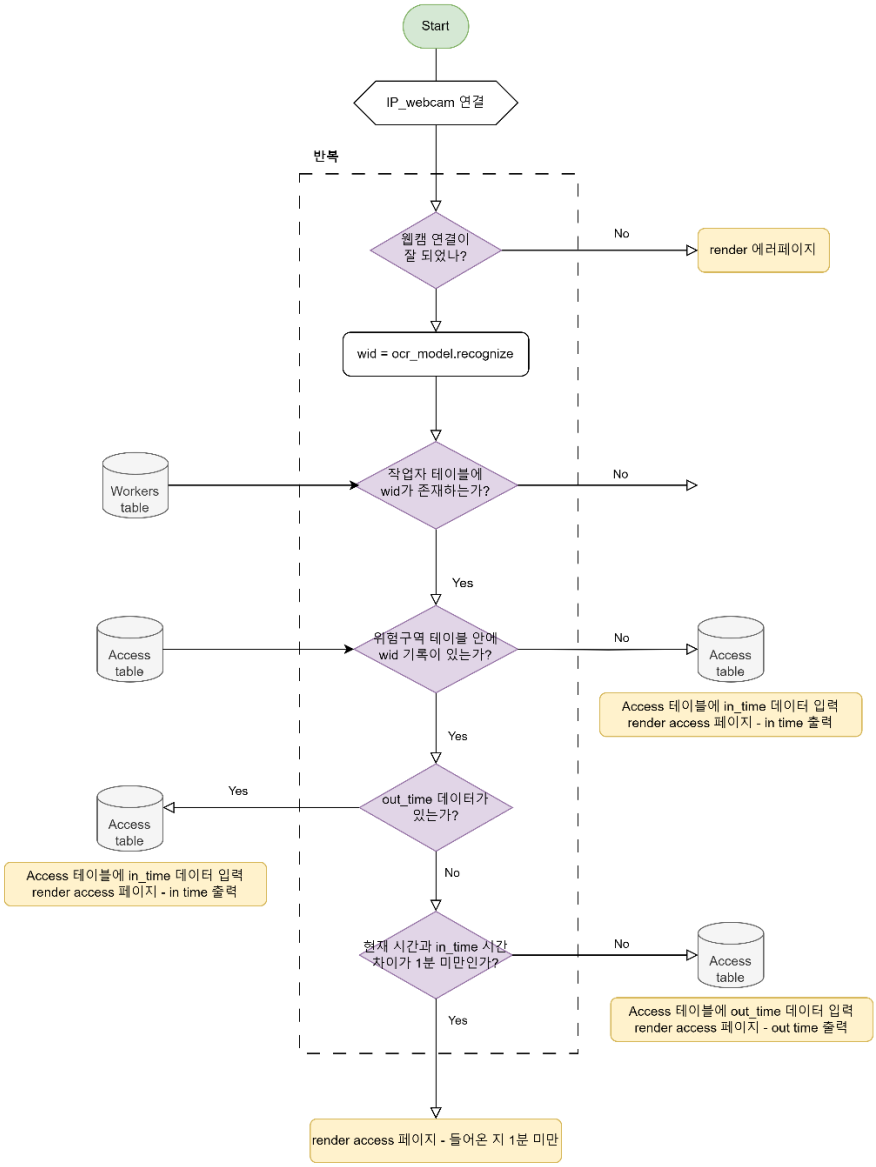
access app

1. access_inout() : 작업구역 출/입 확인
[input] image path(연결 캠 url 주소)
[output] render access/workArea.html

2. check_and_save_to_slack() : 위험 지역 기준 인원 미달일 경우 slack 테이블에 데이터 저장
[input] check_nums() result
[output]

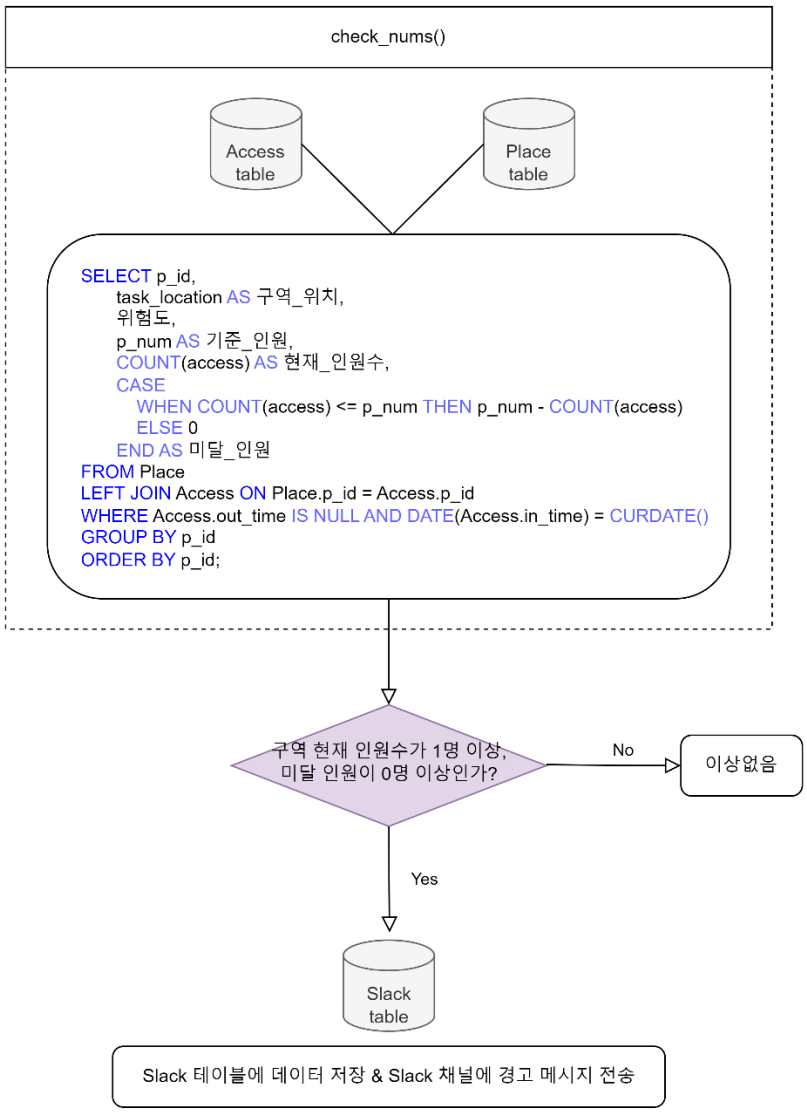
1) check_nums() : 위험 구역 인원 파악
[input]
[output] return 구역별 현재 인원, 미달 인원 정보

- 위험구역 출입 확인 알고리즘



- 위험구역 인원수 확인 알고리즘

runapscheduler - 1분마다 실행



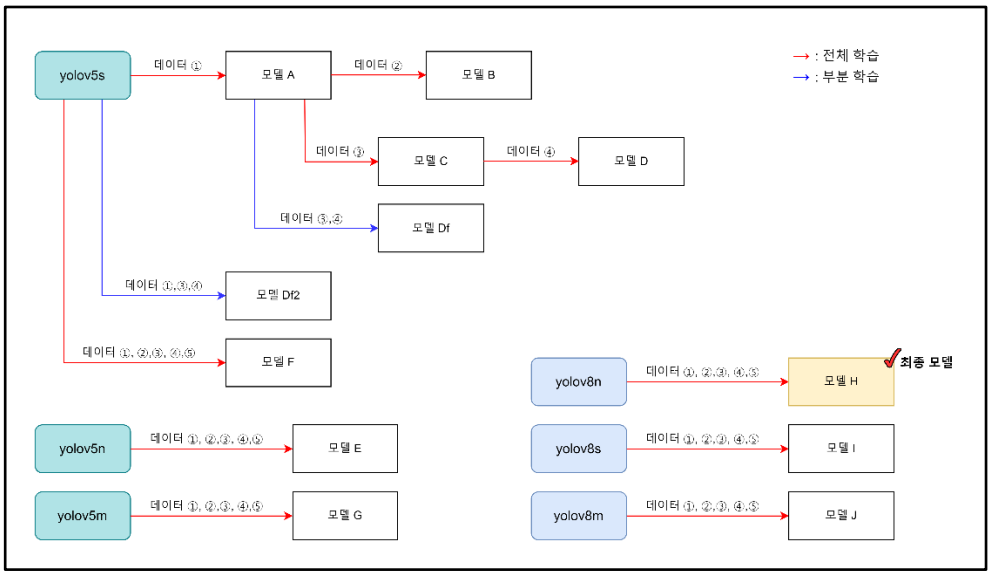
3. 모델 학습 및 검증

가. 객체 탐지 모델 학습

객체 탐지를 위한 모델로 YOLO를 선택하여 학습을 시켰다. 모든 모델은 아래의 학습 속성을 동일하게 적용하여 학습을 진행하였다.

```
--img 640 --batch 16 --epochs 200 –cache
```

모델의 발전 과정은 아래의 순서도와 같다.



모델 학습 순서도

1) 데이터에 따른 모델 비교

모든 학습 속성을 같게 하고, 데이터의 종류를 다르게 하여 학습을 진행하였다.

모델 명	학습한 데이터	class	weights
모델 A	데이터①	helmet, goggle, vest	yolov5s.pt
모델 B	데이터①, ②	helmet, goggle, vest	모델 A
모델 C	데이터①, ③	glasses, helmet, goggle, vest	모델 A

2) 동결에 따른 모델 비교

데이터를 더 효율적으로 학습시키기 위해, fully connected layer를 제외한 layer를 모두 동결한 채로 학습을 진행하였다. 아래 모델들 데이터의 class는 glasses, hat, helmet, goggle, vest로 모두 같다. 다만 각각 동결한 후 학습한 데이터가 다르다.

모델 명	학습한 데이터	동결한 후 학습한 데이터	weights
모델 D	데이터①, ③, ④	-	모델 C
모델 Df	데이터①, ③, ④	데이터③, ④	모델 A
모델 Df2	데이터①, ③, ④	데이터①, ③, ④	yolov5s.pt

3) 가중치에 따른 YOLO v5 모델 비교

YOLO v5의 다양한 가중치 크기 중, 어떤 게 우리 프로젝트에 적합할 지를 알아보기 위한 실험을 진행하였다. 가중치 속성만을 다르게 한 뒤, 지금까지의 모든 데이터를 전체학습 시켜보았다.

모델 명	학습한 데이터	class	weights
모델 E	데이터①~⑤	glasses, hat, helmet, goggle, vest	yolov5n.pt
모델 F	데이터①~⑤	glasses, hat, helmet, goggle, vest	yolov5s.pt
모델 G	데이터①~⑤	glasses, hat, helmet, goggle, vest	yolov5m.pt

4) 가중치에 따른 YOLO v8 모델 비교

YOLO v5의 결과와 비교하기위해, YOLO v8로도 여러 가중치별로 학습을 진행했다. YOLO v5와 같이 가중치 속성만을 다르게 한 뒤, 지금까지의 모든 데이터를 전체학습 시켰다.

모델 명	학습한 데이터	class	weights
모델 H	데이터①~⑤	glasses, hat, helmet, goggle, vest	yolov8n.pt
모델 I	데이터①~⑤	glasses, hat, helmet, goggle, vest	yolov8s.pt
모델 J	데이터①~⑤	glasses, hat, helmet, goggle, vest	yolov8m.pt

나. 모델 검증

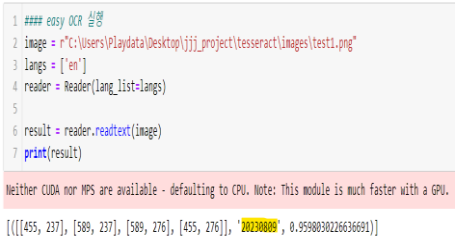
1) 글자 인식 모델

- Tesseract OCR

실시간 영상 테스트를 진행하였다. 이미지 전처리 과정(grayscale 변환, 노이즈 제거, 밝기 및 대비 조절)이 있기에 텍스트를 얼추 찾아냈지만, 텍스트가 멀리 떨어진 경우 성능이 많이 떨어지는 모습을 보였다. 이에 더 성능이 좋은 모델을 찾아보았다.

- Easy OCR

Tesseract OCR 실행 결과



Easy OCR 실행 결과

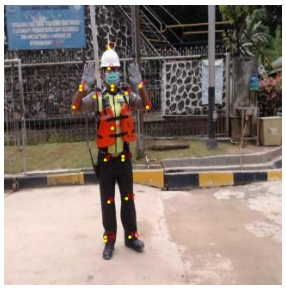
실시간 영상 테스트를 진행한 결과, 어느정도 거리가 멀어져도 텍스트를 잘 찾아냈다. 위의 사진을 보면 '20230809'란 글씨를 탐지할 때, Tesseract OCR은 '20230803'으로 인식하였다. 하지만 Easy OCR은 '20230809'를 정확하게 찾아내는 것을 확인할 수 있다. 이렇듯 Easy OCR은 Tesseract가 잘 찾지 못하던 텍스트도 찾아내는 등 높은 정확도를 보였다. 하지만 모델의 크기가 커서 실시간 화면 송출이 지연되는 현상이 발생하였다.

- Paddle OCR

Paddle OCR은 모델 내부에서 이미지 크기 조정, 노이즈 제거, 이진화 작업 등을 자동으로 수행한다. 따라서 전처리 과정을 삭제하였다. 실시간 영상 테스트를 진행한 결과, 최대 약2m 떨어진 거리에서도 텍스트 탐지가 가능하였다. 처리속도는 0초대로 매우 빨랐고 Easy OCR만큼이나 높은 정확성을 보였다.

2) 자세 추정 모델

동일한 이미지로 세 모델을 각각 실행한 결과는 아래와 같다.

		
Openpose 실행 결과	Cascade Classifier 실행 결과	MediaPipe 실행 결과

- Openpose

인체 포즈 추정 신경망 구조인 protofile 에 대하여 두 가지를 활용해보았다. 우선, 8 개 스테이지로 구성된 네트워크 아키텍처를 적용하였다. 이 구조는 한 개 이미지당 포즈를 추정하는 데에 약 20 초가 소요되었다. 다음으로 4 개의 스테이지로 구성된 네트워크 아키텍처를 적용하였다. 전체적인 속도는 8 초 가까이 빨라졌고, 일부 정확도는 낮아진 모습을 볼 수 있었다.

- Cascade Classifier

정면, 전신, 상반신을 탐지하는 코드를 각각 실험해보았다. 그 결과, 전신 탐지는 어느정도 정확했지만, 정면과 전신 탐지에서 아주 낮은 정확성을 보였다. 속도는 매우 빨랐지만 이미지의 배경에서 엉뚱한 것을 탐지하는 모습을 자주 보였다.

- MediaPipe

이 모델은 하이퍼파라미터를 다양하게 조절하면서 실험을 진행했다. Model complexity를 2에서 1로 변경한 결과, 탐지 시간이 약 0.2초 감소했다. 시간은 감소했지만, 정확도는 눈에 띄는 변화가 없었다. Enable segmentation을 True에서 False로 변경한 결과, 탐지 소요시간이 약 0.05초 감소했다. 이 역시 시간은 감소했지만, 탐지된 점들의 위치는 변화가 없었다.

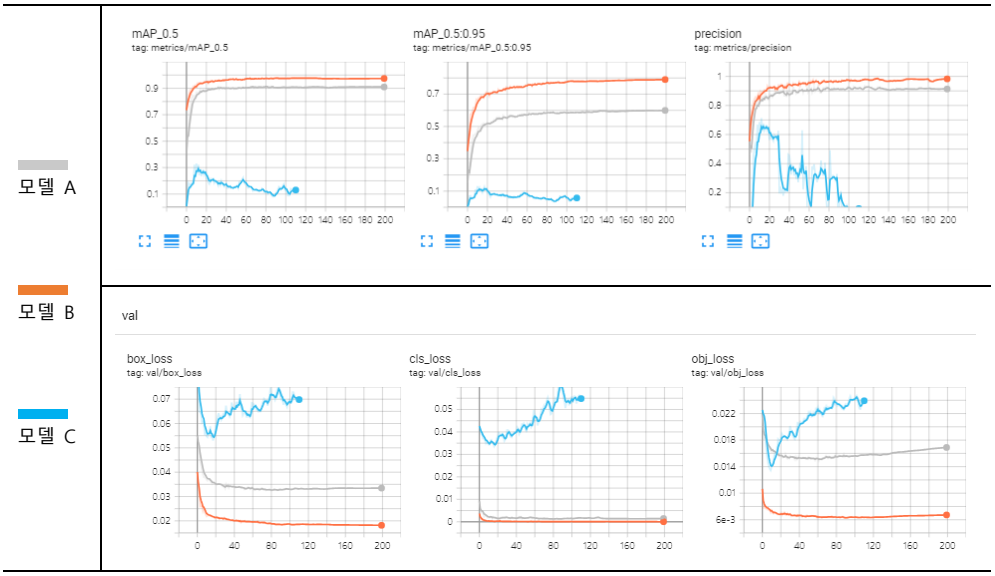
3) 객체 탐지 모델

모델 검증을 위해 다양한 성능평가 지표를 활용하여 모델을 비교하였다. 여러 항목들 중, 우리가 중점적으로 사용한 지표는 아래 5 개 항목이다.

mAP_0.5	객체 탐지 모델에서 사용되는 IoU (Intersection over Union) 임계값이 0.5 일 때의 mAP 값
mAP_0.5:0.95	IoU 임계값을 0.5 에서 0.05 씩 증가시키면서 0.95 까지 변화시킨 경우에 대한 mAP 값
box_ loss	예측된 바운딩 박스와 실제 바운딩 박스 사이의 차이를 측정하는 손실값
cls_loss	객체 탐지 모델에서 개별 클래스에 대한 분류 오차를 측정하는 손실값
obj_loss	주어진 그리드 셀 내에 객체가 있는지 여부를 판단하는데 사용되는 손실값

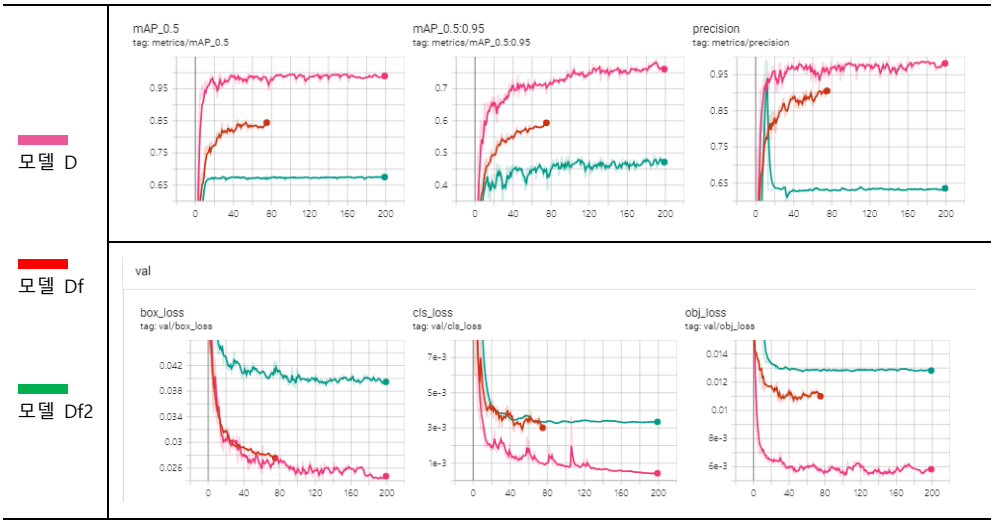
- 데이터에 따른 모델 비교

모델 명	mAP_0.5	mAP_0.5:0.95	box_loss	cls_loss	obj_loss
모델 A	0.907	0.5735	0.03257	1.6415e-3	0.01512
모델 B	0.9769	0.7726	0.0183	7.5898e-5	6.2801e-3
모델 C	0.2707	0.1078	0.05432	0.03432	0.01437



- 동결에 따른 모델 비교

모델 명	mAP_0.5	mAP_0.5:0.95	box_loss	cls_loss	obj_loss
모델 D	0.995	0.7498	0.02558	7.835e-4	5.3983e-3
모델 Df	0.8424	0.5667	0.02855	2.7885e-3	0.01049
모델 Df2	0.675	0.4814	0.0389	3.3699e-3	0.01289



- 가중치에 따른 YOLO v5 모델 비교

모델 명	mAP_0.5	mAP_0.5:0.95	box_loss	cls_loss	obj_loss
모델 E	0.84668	0.4701	0.03677	0.015868	0.007164
모델 F	0.88072	0.56119	0.034425	0.017019	0.006913
모델 G	0.88096	0.56587	0.033838	0.17975	0.007468

- 가중치에 따른 YOLO v8 모델 비교

모델 명	mAP_0.5	mAP_0.5:0.95	box_loss	cls_loss	obj_loss
모델 H	0.86445	0.55052	1.2747	0.763	1.182
모델 I	0.85204	0.55814	1.278	0.71297	1.1985
모델 J	0.87218	0.59274	1.2532	0.68381	1.2583

4. 결과 분석

가. 글자 인식 모델

모델 명	정확도	최대 거리(궁서체 72pt 기준)	실시간성
Tesseract OCR	낮음	20cm	높음
Easy OCR	높음	50cm	낮음
Paddle OCR	높음	2m	높음

제일 처음 선택했던 Tesseract OCR 모델은 이미지를 전처리했음에도 불구하고, 정확도가 낮고 속도도 느렸다. 그 후에 선택한 Easy OCR모델은 정확도가 높았지만 텍스트 거리가 멀어지면 잘 탐지하지 못했다. 실시간 영상에 적용했을 때는 모델이 무거워 영상 송출이 지연되는 현상이 발생했다. 마지막으로 테스트해보았던 Paddle OCR은 이미지 전처리를 따로 하지 않았음에도 비교적 먼 거리까지 텍스트 인식이 가능했고, 실시간성이 높았다. 따라서 프로젝트에 적용할 모델로 Paddle OCR을 선택하였다.

나. 자세 추정 모델

모델 명	정확도	한 개 이미지 탐지 속도	실시간성
Openpose	높음	12.425초	낮음
Cascade Classifier	낮음	0.373초	높음
MediaPipe	높음	0.281초	높음

Openpose는 detection 후 포즈 좌표를 찾기 때문에, 속도가 느리고 정확도가 높았다. 이 점을 보완하기 위해 가벼운 모델인 Cascade Classifier를 이용하였다. Cascade Classifier는 딥러닝이 적용된 기술이 아니기 때문에 속도는 빠르나 정확도가 낮은 편이었다. 엉뚱한 영역을 탐지하는 경우가 많아서 최종 선택에서 제외하였다. 반면에 MediaPipe는 detection 과정을 생략하고 바로 포즈 좌표를 찾으므로 속도가 빨랐다. 또한 딥러닝 모델이므로 높은 정확도를 보였다. 따라서 프로젝트에 이용할 모델로 MediaPipe를 선정하였다.

다. 객체 탐지 모델

우리 프로젝트의 목표를 달성하기 위해 가장 중요한 부분은 정확성과 실시간성이다. 따라서 결과를 분석할 때 이 두 가지를 주요 평가지표로 삼았다. 안전조끼와 헬멧은 모든 모델이 잘 탐지했으므로 정확성을 판단하기 위한 근거로 고글, 안경, 모자 탐지 유무를 실험했다. 실시간성은 한 개 이미지 탐지 속도로 판단하였다. 모든 모델은 10번 이상의 실험을 거쳤고, 그 실험 결과의 최빈값과 평균값은 아래와 같다.

1) 데이터에 따른 모델 비교

모델 명	지표	고글	안경	모자	한 개 이미지 탐지 속도	실시간성
모델 A	좋음	X	X	X	0.791초	보통
모델 B	좋음	X	X	X	0.779초	보통
모델 C	나쁨	O	X	X	0.798초	보통

헬멧이나 조끼에 비해 작은 고글을 잘 탐지하지 못하는 모습을 보여 고글을 탐지해내는 것을 목표로 다양한 데이터를 추가하여 학습을 진행했다. 직접 만든 이미지 데이터까지 추가하자 고글을 탐지해냈다. 하지만 안경과 모자를 각각 고글과 헬멧으로 혼동하는 모습을 보여 이 점을 추후 보완하고자 하였다.

2) 동결에 따른 모델 비교

모델 명	지표	고글	안경	모자	한 개 이미지 탐지 속도	실시간성
모델 D	좋음	O	O	X	0.801초	보통
모델 Df	좋음	O	X	X	0.792초	보통
모델 Df2	나쁨	X	X	X	0.830초	보통

모델 C에서 학습한 데이터에 안경과 모자 데이터를 추가하여 학습을 진행했다. 그 결과 안경과 모자 데이터를 전체학습한 모델 D의 성능은 향상되었다. 여기서 지표와 성능을 더 끌어올리기 위해 동결을 적용하여 학습을 진행했다. 추가 학습하는 데이터의 수가 적기에 동결을 적용하면 지표와 성능이 더 좋아질 것이라 예상했지만, 동결을 적용해 학습한 데이터 수가 늘어날수록 지표와 성능이 모두 하락하는 결과를 얻었다. 따라서 동결하면 성능이 크게 저하되므로 동결을 하지 않고 전체학습을 시키기로 결론 내렸다.

3) 가중치에 따른 YOLOv5 모델 비교

모델 명	지표	고글	안경	모자	한 개 이미지 탐지 속도	실시간성
모델 E	좋음	O	O	O	0.365초	높음
모델 F	좋음	O	O	O	0.782초	보통
모델 G	좋음	O	O	O	1.989초	낮음

위의 여섯 개 모델을 통해 얻은 최적의 조건을 공통으로 적용하고, 직접 제작한 데이터를 추가하여 학습을 진행했다. YOLOv5 모델의 가중치만을 변경한 채로 학습시킨 결과는 위와 같다. 모델 E(nano)의 경우, 비교적 가벼운 모델임에도 불구하고 학습시킨 데이터가 다양하고 많았기에 지표와 성능 그리고 속도 모두 매우 훌륭했다. 모델 F(small)는 모델 E와 비슷한 성능이지만 속도는 2배가 되었다. 모델 G(middle)는 속도가 모델 E의 4배이며 성능에서 큰 차이를 보이지는 않았다. 따라서 성능과 실시간성이 모두 뛰어난 모델 E를 최종 모델로 염두해두기로 하였다.

4) 가중치에 따른 YOLOv8 모델 비교

모델 명	지표	고글	안경	모자	한 개 이미지 판단 속도	실시간성
모델 H	좋음	O	O	O	0.263초	높음
모델 I	좋음	O	O	O	0.566초	보통
모델 J	좋음	O	O	O	1.271초	낮음

위에서 YOLO v5 모델을 비교한 것과 동일한 조건으로 최신 모델인 YOLO v8 모델에 대한 실험을 진행했다. 이 최신 모델은 v5 모델보다 속도는 빠르고 정확성은 떨어질 것이라 예상했다. 하지만 실험 결과 YOLO v8 모델은 YOLO v5 모델과 비슷한 정확성을 보였다. 또한 속도는 v5 모델보다 유의미하게 빠른 모습을 보였다. 따라서 객체 탐지 모델은 YOLO v8 n 모델을 학습한 모델 H로 선정하였다.

5. 결론

가. 기능별 모델 활용

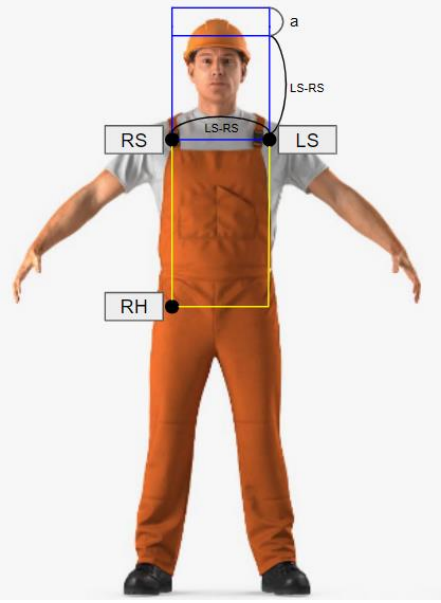
1) 출근 관리 기능

이 기능에서는 글자 인식 모델을 활용하였다. 순서는 다음과 같다. 먼저, 웹캠에 이미지가 들어오면 Paddle OCR 모델이 작업자 번호를 판별한다. 모델이 인식한 번호가 DB의 작업자 테이블에 존재한다면, 오늘 출근 기록이 있는지 확인하는 알고리즘을 거쳐 DB에 출근 기록을 한다. 출근 기록을 한 뒤에 보호구 착용 관리 알고리즘이 실행된다.

2) 위험구역 관리 기능

이 기능에서는 출근 관리 기능과 비슷한 구조로 글자 인식 모델을 적용하였다. 구조는 다음과 같다. 위험구역에 어떤 작업자가 출입하는 지를 모델이 판별하면 그 결과를 DB에 저장한다. 이렇게 저장된 테이블에서 현재 인원수를 세는 SQL 질의문을 scheduler를 통해 1분마다 실행시킨다. 이 작업으로 위험구역에 기준 인원수가 충족되었는지 확인할 수 있다.

3) 보호구 착용 관리 기능



해당 기능에서는 자세 추정 모델과 객체 탐지 모델을 사용하였다. 그 구조는 다음과 같다. 이미지가 들어오면, 먼저 자세 추정 모델인 MediaPipe가 실행된다. MediaPipe는 이미지 속 작업자의 Leftshoulder, Rightshoulder, Righthip 좌표들을 산출한다. 이 좌표를 통해서 머리 부분과 몸통 부분 총 2개의 가이드 박스를 만들고, 그 가이드 박스의 좌표값을 return한다. MediaPipe의 연산이 끝나면 객체 탐지 모델인 YOLOv8 n가 실행된다.

객체 탐지 모델은 이미지에서 안전 보호구(헬멧, 고글, 안전조끼)를 탐지해낸다. 모델이 탐지한 객체 박스의 중앙점이 MediaPipe가 return한 가이드 박스 안쪽에 존재한다면, 보호구를 정위치에 올바르게 착용했다고 판단한다.

초기 모델을 사용할 때는 정확성을 보완하기 위해 보호구를 착용 여부 판단 알고리즘을 작업자당 5번 반복하였다. 이렇게 5개의 결과값을 얻어 최빈값을 최종 결과값으로 return했다. 하지만 모델 E부터는 정확성이 뛰어나 3번으로 반복횟수를 줄였다. 이를 통해 작업자당 전체 알고리즘 실행 시간을 2초대로 단축시켰다.

나. 제약사항

GPU 리소스가 부족하여 모델을 학습하는데 어려움을 겪었다. GOOGLE COLAB에서 약 26000장의 이미지를 학습시키는 것을 시도한 적이 있었는데, 컴퓨팅 시간 부족으로 항상 100epoch을 넘기지 못하고 끊기는 환경이 아쉬웠다.

다. 개선사항

현재 사용한 MediaPipe 모델은 이미지 내 한 명의 포즈만 탐지가 가능하여 두 명 이상의 사람이 화면으로 들어온다면 정확한 탐지가 불가능하다. 두 명 이상의 사람이 화면으로 들어왔을 시에도 탐지가 가능하게 하려면, MediaPipe 모델 대신 Movenet이라는 모델로 대체해야 한다. 또한 매 이미지마다 OCR을 측정하고 글자 위치의 좌표를 통해 자세 추정 모델과 객체 탐지 모델에게 가이드 박스 조건을 주어야 한다.

현재는 위험구역 출입 관리에 OCR 모델만을 활용하고 있다. 이후에 기회가 된다면, OCR로 출입기록을 함과 동시에 위험구역 전체를 웹캠으로 감독하며 실시간 image segmentation 모델을 통해 인원수를 측정하는 기능을 추가하고 싶다.

라. 기대효과 및 활용분야

2023년 8월 서울시 조사에 따르면, 서울 지역 사고사망재해의 43.5%가 안전 보호구 미착용에서 기인하였다. 우리 프로젝트가 실제 현장에서 활용된다면, 안전 보호구 미착용에 의한 사교율이 대폭 감소될 것으로 예상된다. 따라서 전체 산업현장 재해의 약 40%를 감소시킬 수 있을 것으로 기대된다.

고용노동부가 2021년에 발표한 통계에 의하면, 전체 안전조치 위반 사례의 약 80%는 소규모 사업장에서 적발되었다. 규모가 작을수록 관리자가 적어지기에 이런 결과가 나온 것으로 보인다. 우리 프로젝트의 관리 자동화 특징이 이런 문제를 해결할 수 있을 것이라 생각한다.

과학기술정보통신부가 연구실 안전 실태조사를 시행한 결과, 국내 연구실의 약 30%가 보호구를 미착용하는 것으로 나타났다. 우리 프로젝트는 공장 같은 산업현장에서 확장하여 다양한 연구실 및 실험실에서도 활용가능한 범용성을 지니고 있다.