덕성여자대학교 & 해성여자고등학교

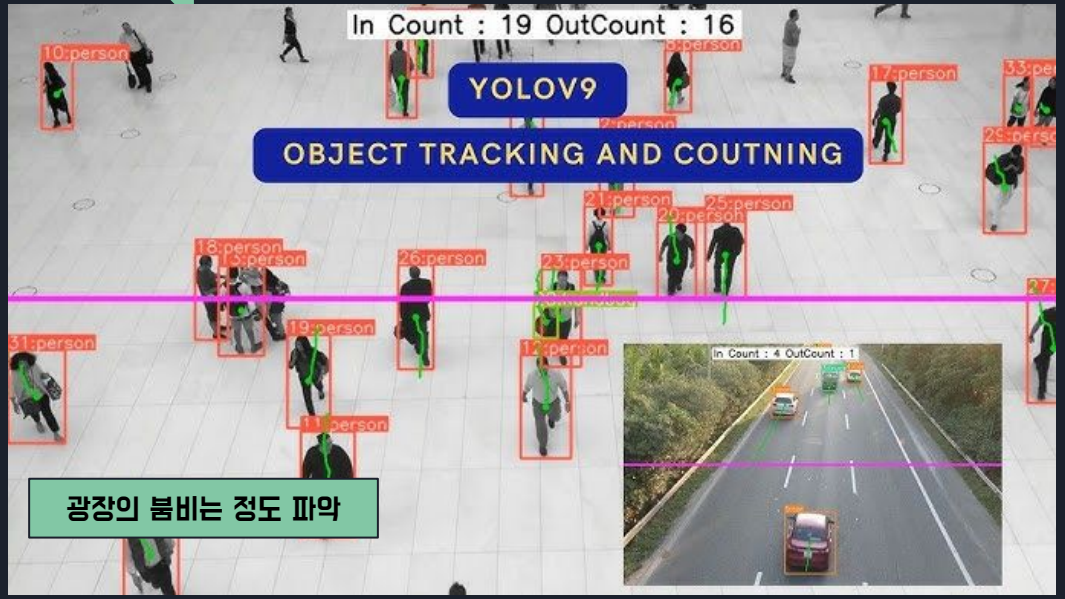
2025학년도 꿈이룸 창의융합인재 5주차

사례탐구로 이해하고 구현하는 인공지능 문제해결

하 윤 종 (hayunjong83@gmail.com)

우리가 배운 것은 무엇인가?

객체 탐지(object detection) 기술의 이해와 활용



하지만, 가장 중요한 것은 우리가 주변에서 느끼는 불편함과 문제점을 인공지능 기술로 해결할 수 있음을 배우는 것이다.

문제 해결을 위한 탐구 과정에서 고려해 볼 수 있는 몇가지 사항들

첫째. 실제 자신이 "관찰"한 사실을 바탕으로 해결해야 할 문제점을 생각해내고 제시하였는가?

둘째. 위의 문제점을 해결하기 위한 방법으로 제시된 공학적 해결책이
문제점과 실제적 연관성이 있으며, 그 이유는 타당한가?

셋째. 문제의 해결책으로 제시된 방법이 갖는 장점은 무엇인가?
그리고 그 장점을 근거를 들어서 제시하였는가?

넷째. 탐구과정을 거치면서 느꼈던 어려운 점과 아쉬운 점을 구체적으로 잘 표현하였고,
보완할 방법에 대한 계획을 제시하였는가?

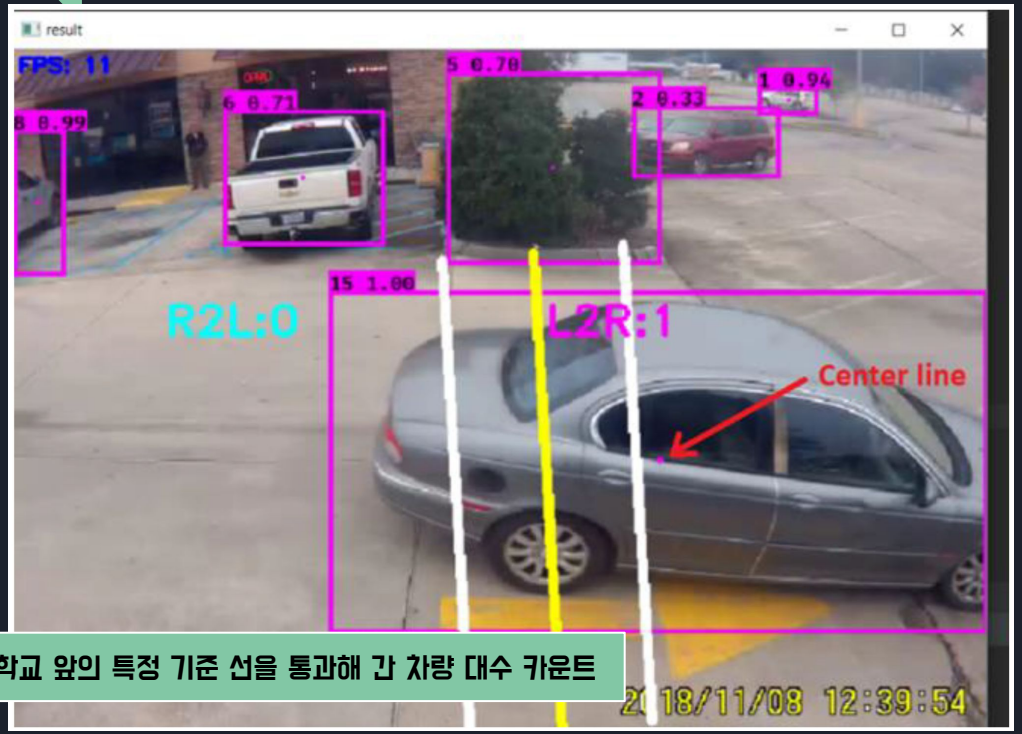
사례 연구 1. 아침 등교 시간 차량으로 붐비는 학교 앞 도로

단계 1) 문제점 발견 - (학교 앞 아침 9시 사진을 직접 찍어 제시하며)
학교 앞 도로는 등교 시간에 너무 붐비고 차가 많아서, 학생들은 늘 위험함을 느끼고 있어.

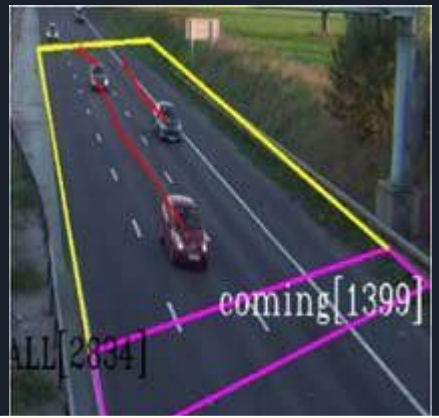
예시 사진 1 - 직접 찍은 아침 학교 앞 도로 사진

단계 2) 인공지능 기술을 해결책으로 제시

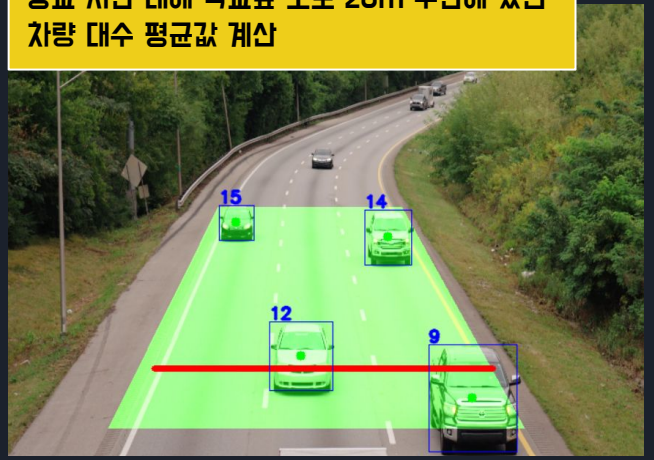
- 특정 시간 대에 몇 대의 차가 학교 앞 도로를 지나갔고 가장 붐비는 시간대가 등교시간이 맞았는지를 확인하기 위해서 **객체 탐지 모델인 YOLO를 사용**하려고 해



학교 앞의 특정 기준 선을 통과해 간 차량 대수 카운트



등교 시간 대에 학교앞 도로 20m 구간에 있던 차량 대수 평균값 계산



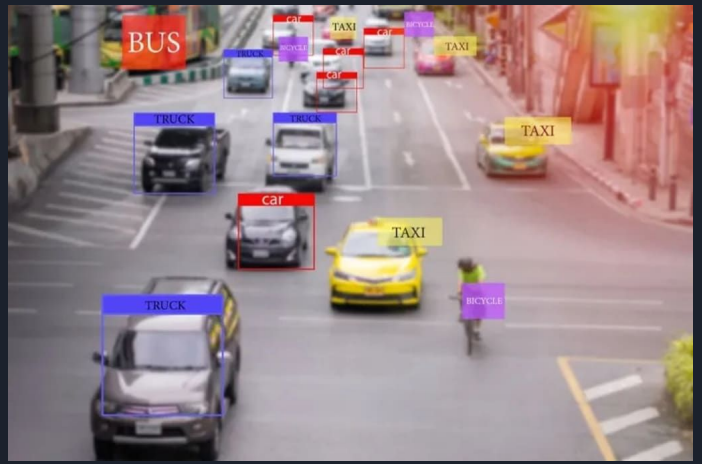
단계 3) 제시한 해결책이 타당한 이유

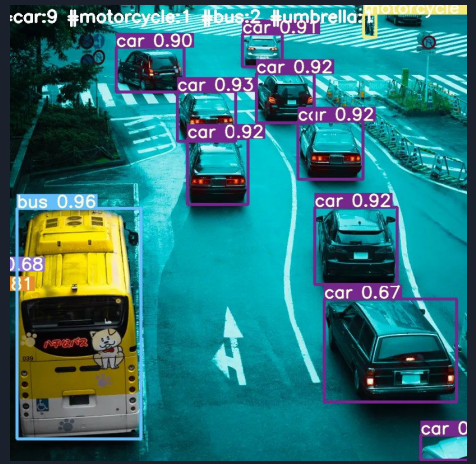
- 학교 앞 도로를 지나가는 차량 대수는 사람이 바로 파악하기 힘들 정도로 **불규칙적**이며
  오랜시간동안 반복해서 관찰하고 **기록**해야 한다.

- 이런 의미에서 인공지능인 객체 탐지 모델을 사용하여
  지나간 차량의 대수를 알아내는 것은 **효율적인 방법**이며.
  객체 탐지 분야의 빠른 발전 속도를 생각해볼 때 **자동화하기 쉽고**, **매우 정확한 결과**를 기대할 수 있다.

- 객체 탐지 모델의 학습에는 학습 데이터의 양이 매우 중요한 요소인데
  차량 데이터는 직접 구하기도 쉬웠고, 특히 차량 한 대도 놓치지 않을 정도로 성능 높은 모델이 많다.

- 방학 중 인공지능 캡스톤 과정을 통해서 그 개념을 이해하고 직접 내 손으로 구현해 보았다.





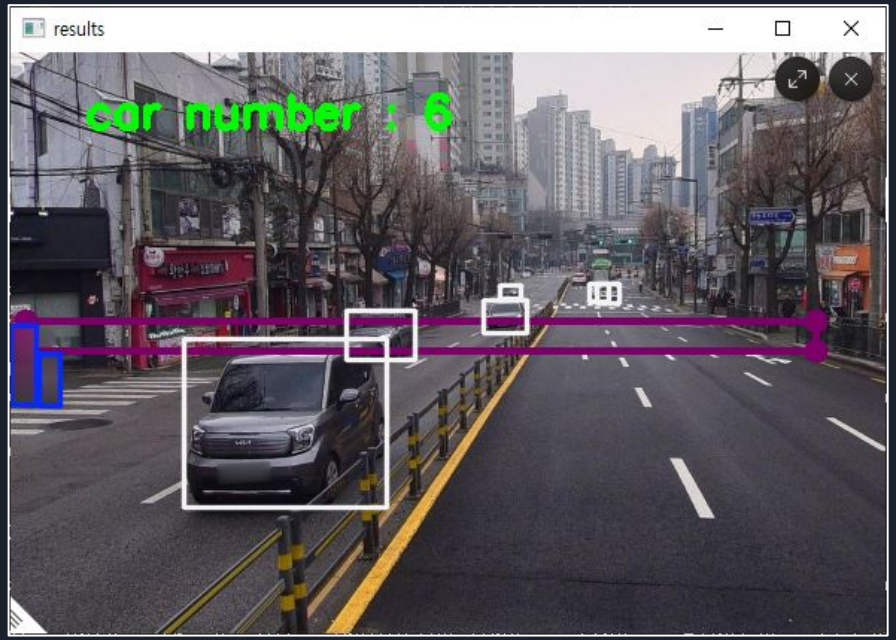
단계 4) 탐구 과정 수행 결과 토의

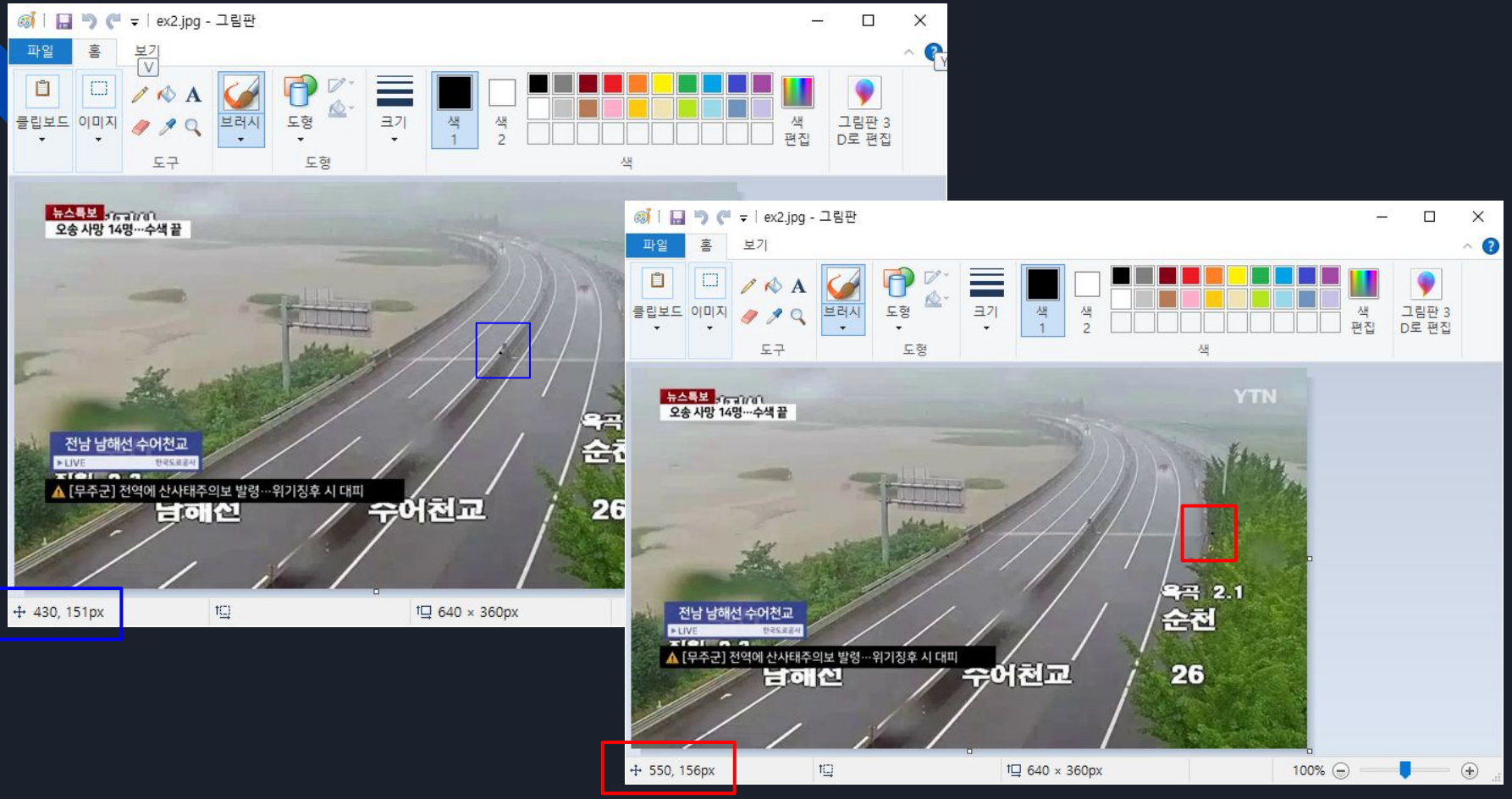
- 실제 7일 (2025년 *월 *일부터 *월 *일까지)간 등교시간 차량 이동 건수는 **건으로
  오전 시간의 차량 이동 건수인 **건에 비하여 **배 높다는 것을 알 수 있었다.

- 이를 통해, 아침 등교시간에 학교 앞 도로의 차량 이동 건수가 많아서
  학생들이 위험하게 느낄 수 있다고 제시한 문제점이 옳았다는 것을 탐구과정을 통해 알 수 있었다.

- 아쉬운 점 :
  우리가 가진 노트북을 학교 앞 도로에 직접 설치하고 객체 탐지 모델을 실행했어야해서
  (충전 문제 와 설치상의 안전 문제로 인하여) 더 많은 시간을 관찰할 수 없었다는 점이 아쉬워.
  더 많은 시간을 분석할 수 있다면 더 정확한 수치를 구하여
  설득력을 높일 수 있었다고 생각한다.

- 실제로 학교에서 배운 과정이 실생활의 문제점을 해결하는 방법이 된다는 것을
  경험할 수 있어서 뿌듯하고 보람찬 시간이었다고 생각한다.


results
car number : 6


ex2.jpg - 그림판
파일
홈
보기
클립보드
이미지
브러시
도형
크기
색 1
색 2
색 편집
그림판 3D로 편집
도구
도형
색
뉴스특보
오송 사망 14명…수색 끝
전남 남해선 수어천교
LIVE
[무주군] 전역에 산사태주의보 발령…위기징후 시 대피
남해선
수어천교
YTN
옥곡 2.1
순천
26
430, 151px
640 × 360px
550, 156px
100%


Ultralytics Solutions
Car: IN 0 OUT 2
뉴스특보
예천 산사태 8명 못 찾아
53 car 0.45
36 car 0.54
전남 남해선 수어천교
LIVE
한국도로공사
[금산군] 전역에 산사태경보 상향 발령…우려 지역 대피
남해선
수어천교
26
옥곡 2.1
순천

참고코드 1

도로 앞 이미지에 대해서
차량 대수를 탐지하고.
지나가는 사람에 대해
블러링을 수행하는 코드

ex1_1.py

day4 > ex1_1.py > ...

```python
from ultralytics import solutions
import cv2

# 주어진 경로의 이미지를 읽는다.
image_path = "ex1.jpg"
img = cv2.imread(image_path)

# 이미지 안에서 자동차(2번 클래스)의 개수만 표현한다.
counter = solutions.ObjectCounter(
    show=False,         # OpenCV로 직접 화면에 띄우기 위해 False
    model="yolo11s.pt",
    classes = [2],
    show_conf = False,
    show_labels = False
)

# 자동자의 개수를 센다.
results = counter(img)

# 파악한 객체의 개수는 results.total_tracks로 알 수 있다.
cv2.putText(results.plot_im, f"car number : {results.total_tracks}",
            (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 3)

# 결과이미지에서 사람(0번 클래스)은 블러 처리한다.
blurrer = solutions.ObjectBlurrer(
    show =False,         # OpenCV로 직접 화면에 띄우기 위해 False
    model="yolo11s.pt",
    classes = [0],
    show_conf = False,
    show_labels = False,
)

# 이번에서는 자동차 개수를 센 결과이미지를 입력한다.
results2 = blurrer(results.plot_im)

# 결과 이미지(results.plot_im)를 화면에 띄운다.
cv2.imshow("results", results2.plot_im)

cv2.waitKey(0)    # 바로 사라지지 않게 대기한다.
cv2.destroyAllWindows()
```

- 참고 코드 : ex1_1.py
- 사용한 이미지 : ex1.jpg

참고코드 2

동영상을 읽어서
첫 이미지 저장 코드

ex1_2.py

day4 > ex1_2.py > ...

```python
import cv2

# 비디오를 열어서 첫번째 이미지만 저장한다.
cap = cv2.VideoCapture("ex2.mp4")
if cap.isOpened():
    ret, img = cap.read()
    cv2.imwrite("ex2.jpg", img)

cap.release()
```

- 참고 코드 : ex1_2.py
- 사용한 동영상 : ex2.mp4
- 결과 이미지 : ex2.jpg

참고코드 3

동영상에서
설정된 영역(=직선)을
지나가는 차량대수 확인

ex1_3.py M

day4 > ex1_3.py > ...

```python
from ultralytics import solutions
import cv2

video_path = "ex2.mp4"
cap = cv2.VideoCapture(video_path)
# 영상정보를 얻는다.
fps = cap.get(cv2.CAP_PROP_FPS)             # 초당 재생속도(fps)
w = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))  # 영상의 가로길이(너비)
h = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT)) # 영상의 세로길이(높이)

counter = solutions.ObjectCounter(
    show=True,
    region=[(430, 150), (550, 150)],     # 관찰할 영역을 지정한다.
    model = "yolo11l.pt",
    # classes = [2],
)

while cap.isOpened():
    ret, img = cap.read()
    if ret:
        results = counter(img)
        # cv2.imshow("video", results.plot_im)

        if cv2.waitKey(30) == ord('q'):
            break

    else:
        break

cap.release()
cv2.destroyAllWindows()
```

- 참고 코드 : ex1_3.py
- 사용한 영상 : ex2.mp4

사례 연구 2. 교복 착용 여부 자동 확인 시스템

1. 문제 발견

- 학교 복도에서 찍은 사진을 관찰해보니, 일부 학생들이 체육복 상태로 수업에 들어오거나 교복을 입지 않은 경우가 있었다.
- 선생님이 매번 일일이 확인하기 어렵고, 빠르게 지나가는 학생들을 육안으로 구별하기 쉽지 않다.

2. 해결책 제시

- YOLO 객체 탐지 모델을 사용해서 "사람" 객체를 감지한 뒤, 이미지 분류기를 추가하여 "교복" 착용 여부를 자동으로 판단해볼 수 있다.

3. 해결책의 타당성

- 객체 탐지로 사람을 찾고, 교복 착용 여부는 이미지 분류(전이 학습 활용 가능)로 판단할 수 있어 조합이 가능하다.

- 방학 중 프로젝트에서 이미지에 대해 라벨링하고 간단한 분류 모델을 학습한 경험이 있었기 때문에, 이를 실제 상황에 응용할 수 있다고 판단했다.

4. 수행 결과 및 토의

- 복도에서 촬영한 100장의 사진 중에서 87건을 정확히 분류했다.

- 라벨링 작업이 시간이 오래 걸려 데이터 수가 부족했던 점이 아쉬웠다.

- 다음에는 자동 라벨링 도구나 더 다양한 교복 이미지로 학습시켜 성능을 높이고 싶다.

사례 연구 3. 길고양이 출현 지역 자동 파악 시스템

1. 문제 발견

- 학교 주변에 길고양이가 자주 출몰하지만, 어디에서 자주 나타나는지 학생들이 주관적으로만 이야기하고 있어 정확한 정보가 부족하다.
- 급식 쓰레기를 버리는 장소 근처에서 고양이를 본 경험이 많았다.

2. 해결책 제시

- YOLO를 이용해 방학 중에 학교 주변을 촬영한 CCTV 영상을 분석하고, 고양이 객체의 출현 위치를 기록한다.
- 출현 빈도가 높은 장소를 시각화해 고양이에게 안전한 장소를 마련하거나 쓰레기통 정비 계획을 제안할 수 있다.

3. 해결책의 타당성

- YOLO는 고양이 객체를 인식할 수 있으며, 위치 정보도 함께 얻을 수 있어 heatmap 형태로 분석이 가능하다.
- 사람이 하루종일 관찰하는 것보다 효율적이고, 동물의 출몰을 기록으로 남길 수 있다.

4. 수행 결과 및 토의

- 오전 시간보다 저녁 시간대 고양이 출현 빈도가 약 3배 높았다.
- 쓰레기통 근처에서 70% 이상의 고양이가 포착되었고, 이 지역을 개선할 필요성이 확인되었다.
- 아쉬운 점은 밤 시간 영상의 품질이 낮아 탐지 정확도가 떨어졌다는 점이다.

사례 연구 4. 킥보드 무단 주차 문제 탐지

1. 문제 발견

- 학교 앞 인도에 전동 킥보드가 무질서하게 세워져서 학생들의 보행을 방해하고, 교통사고 위험이 높아지고 있다.

2. 해결책 제시

- YOLO로 촬영된 이미지에서 전동 킥보드를 탐지하고, '보행자 통행로'에 얼마나 침범하고 있는지를 확인한다.
- 탐지된 객체의 위치를 기준으로 보행 공간 침해 여부를 시각화해본다.

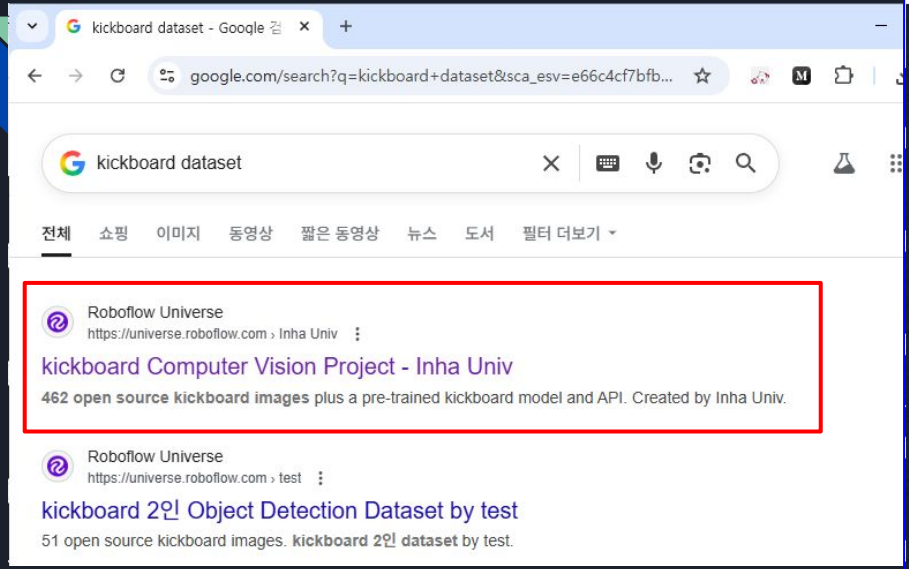
3. 해결책의 타당성

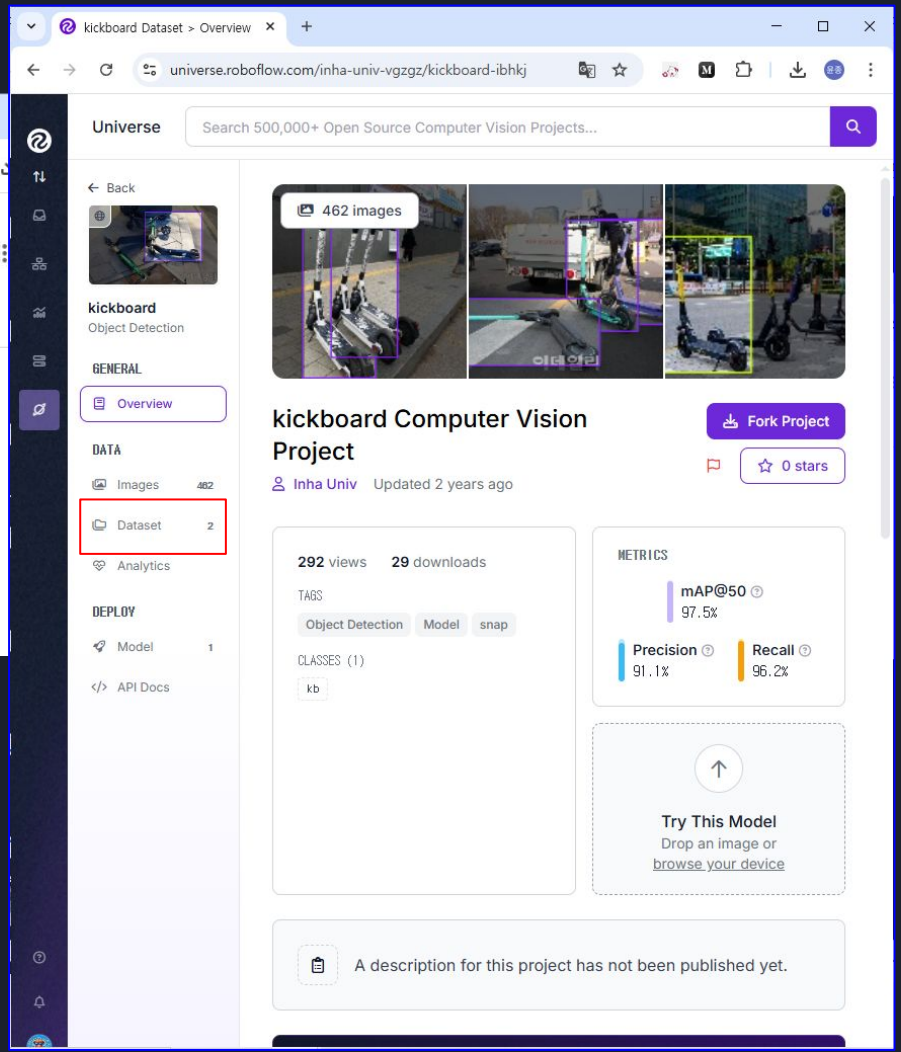
- YOLO는 킥보드 객체도 감지 가능하며, 여러 시간대 사진을 비교해 무단 주차 패턴을 분석할 수 있다.
- 시간대별, 위치별 데이터로 해결책(가이드라인 설치, 지도 배포 등) 제안에 신뢰를 더할 수 있다.

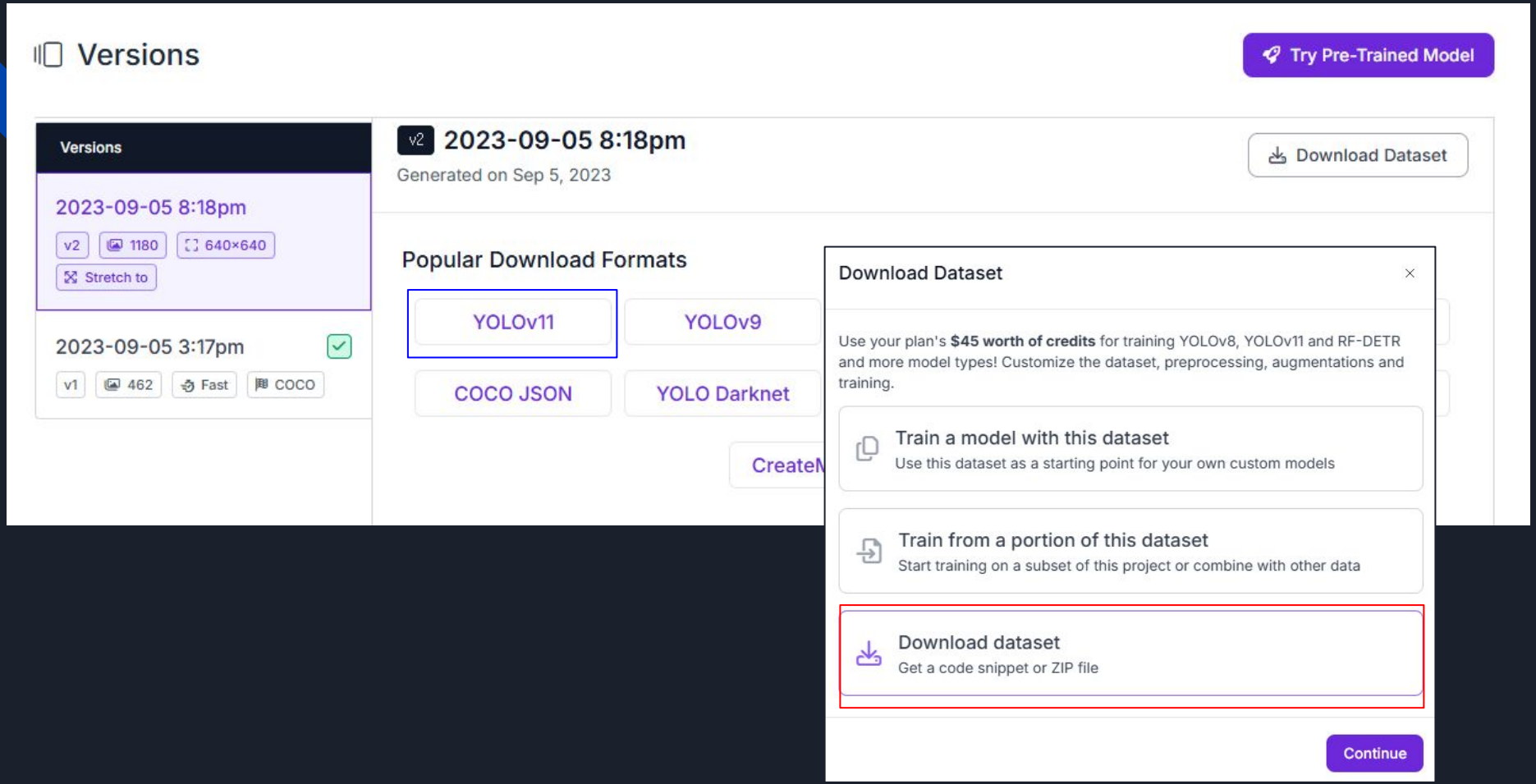
4. 수행 결과 및 토의

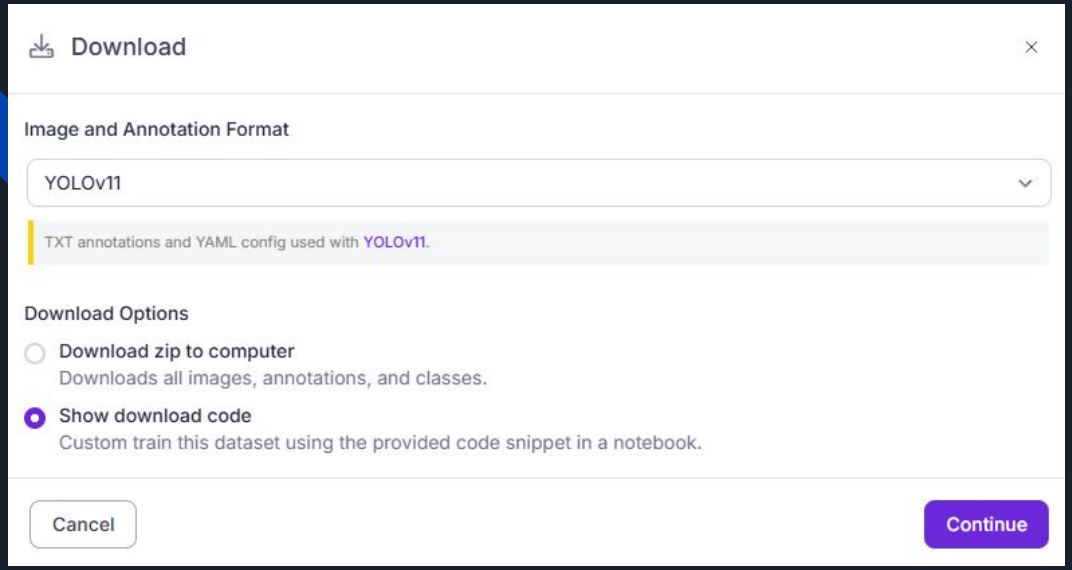
- 오후 6~8시에 킥보드가 보도 위에 무질서하게 있는 비율이 가장 높았다.
- 탐지 정확도는 높았지만, 도로/보도 구분을 직접 그려야 해서 자동화에 한계가 있었다.
- 다음에는 Segmentation 기술을 활용해 보행로를 자동으로 인식하도록 도전하고 싶다.

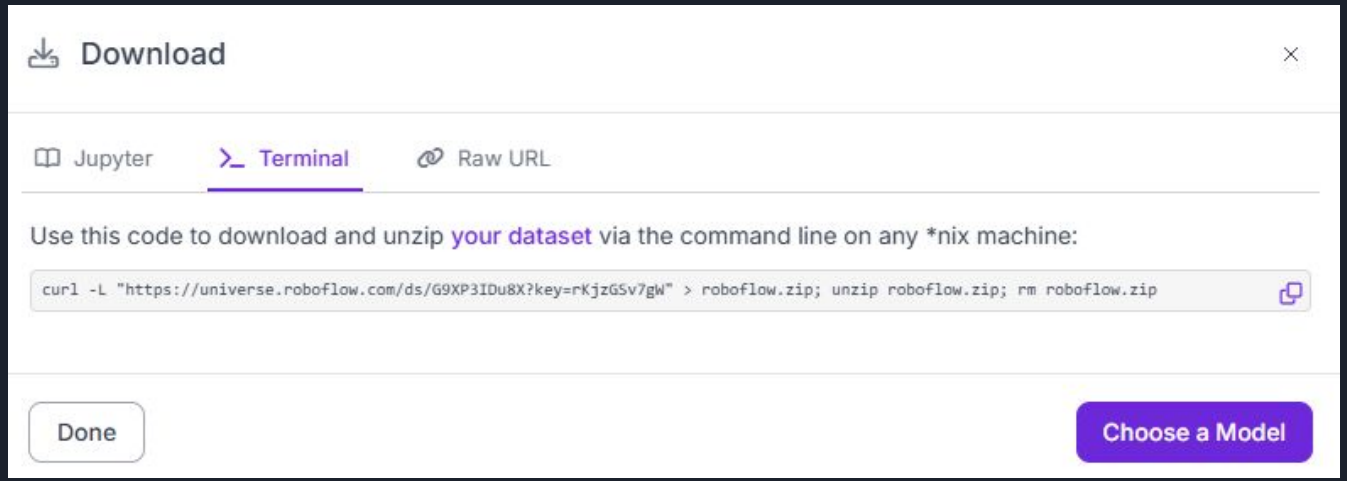
참고 1 킥보드 데이터셋을 찾는 방법 예시






Versions
Try Pre-Trained Model
Versions
2023-09-05 8:18pm
v2
1180
640×640
Stretch to
2023-09-05 3:17pm
v1
462
Fast
COCO
v2 2023-09-05 8:18pm
Generated on Sep 5, 2023
Download Dataset
Popular Download Formats
YOLOv11
YOLOv9
COCO JSON
YOLO Darknet
Download Dataset
Use your plan's $45 worth of credits for training YOLOv8, YOLOv11 and RF-DETR and more model types! Customize the dataset, preprocessing, augmentations and training.
Train a model with this dataset
Use this dataset as a starting point for your own custom models
Train from a portion of this dataset
Start training on a subset of this project or combine with other data
Download dataset
Get a code snippet or ZIP file
Continue

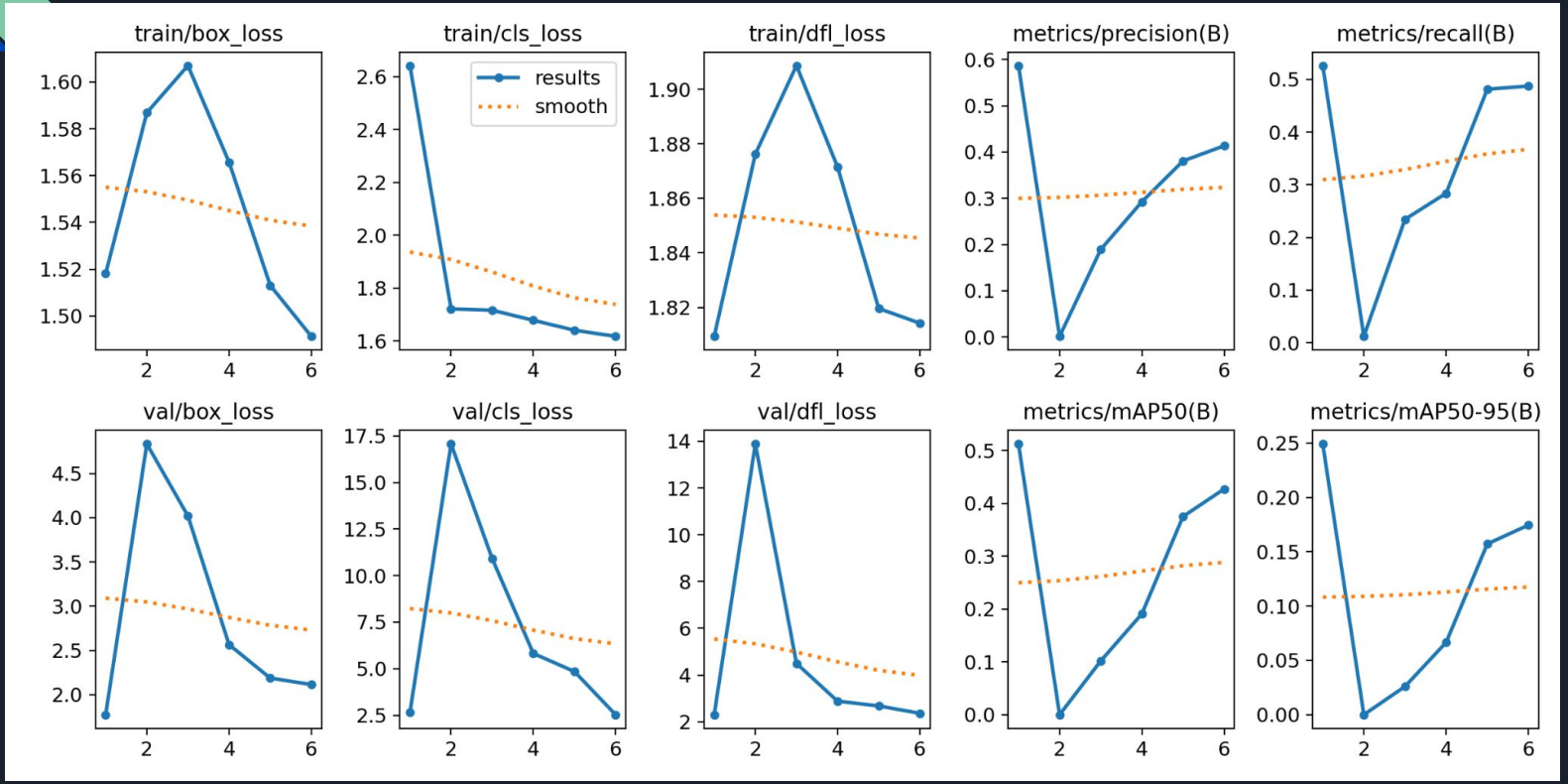

Download
Image and Annotation Format
YOLOv11
TXT annotations and YAML config used with YOLOv11.
Download Options
Download zip to computer
Downloads all images, annotations, and classes.
Show download code
Custom train this dataset using the provided code snippet in a notebook.
Cancel
Continue


Download
Jupyter
Terminal
Raw URL
Use this code to download and unzip your dataset via the command line on any *nix machine:
curl -L "https://universe.roboflow.com/ds/G9XP3IDu8X?key=rKjzG5v7gW" > roboflow.zip; unzip roboflow.zip; rm roboflow.zip
Done
Choose a Model

참고 2

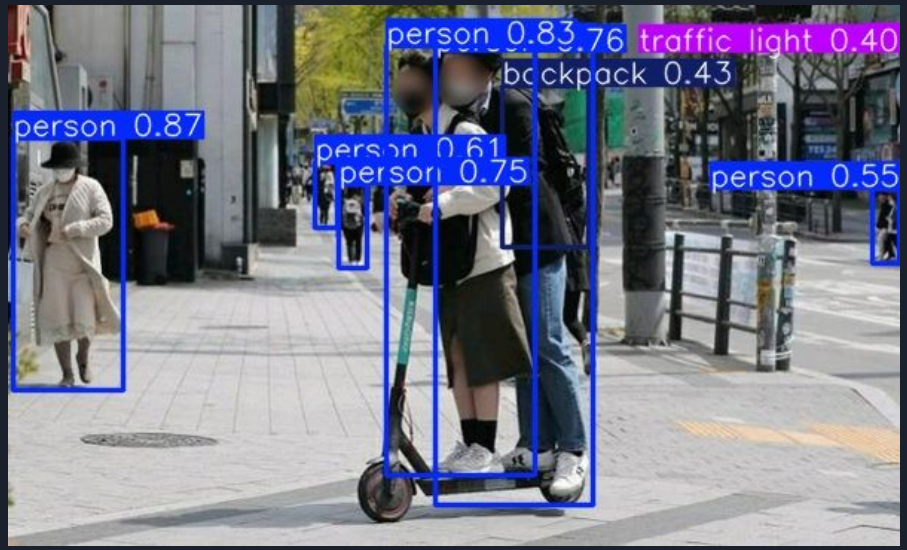
확보한 데이터셋을 이용하여, YOLOv11 학습시키기

- 참고 코랩 : 공유링크
- 학습된 모델 파일 : kb_yolo.pt
- 학습 그래프 이미지 : results.png



- 킥보드 객체탐지 직접(=커스텀) 학습 전체 코드 : 링크
  참고)
  학습 데이터만 받아 사용할 때에는 다음의 명령어를 사용한다.

```
curl -L "https://universe.roboflow.com/ds/G9XP3IDu8X?key=rKjzGSv7gW" > roboflow.zip; unzip roboflow.zip; rm roboflow.zip
```



참고 3

사전훈련된 YOLO모델과 직접 학습시킨 모델의 결과 비교

- 참고 코드 : ex2_1.py
- 사용한 이미지 : ex3.jpg
- 결과 이미지 1 : kickboard_result1.jpg
- 결과 이미지 2 : kickboard_result2.jpg
- 결과 이미지 3 : kickboard_result3.jpg

ex2_1.py

day4 > ex2_1.py > ...

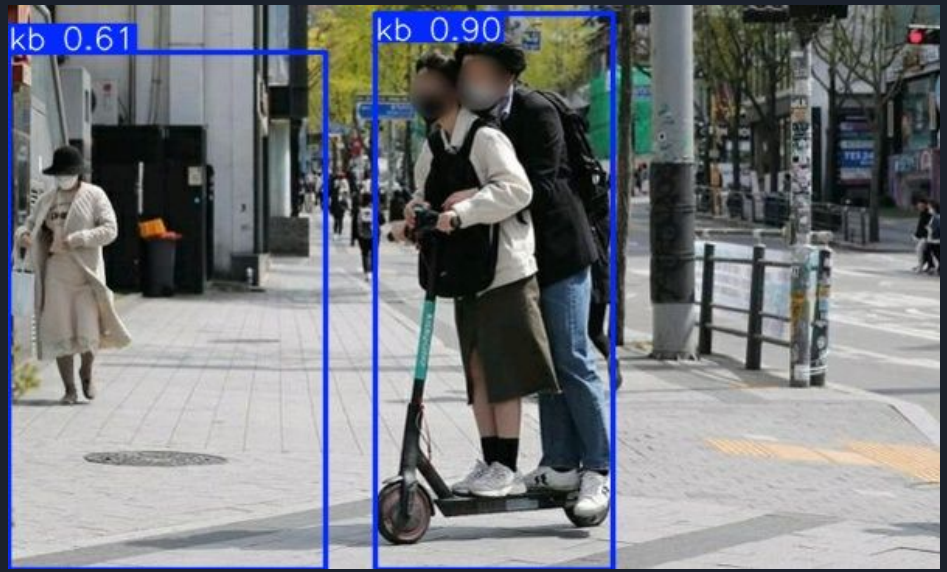
```python
from ultralytics import YOLO
import cv2

image_path = "ex3.jpg"
model_1 = YOLO("yolo11s.pt")
model_2 = YOLO("kb_yolo.pt")

results_1 = model_1(image_path)
results_2 = model_2(image_path)
results_3 = model_2(image_path, conf=0.8)

cv2.imshow("results", results_3[0].plot())
cv2.waitKey(0)
cv2.destroyAllWindows()

results_1[0].save("kickboard_result1.jpg")
results_2[0].save("kickboard_result2.jpg")
results_3[0].save("kickboard_result3.jpg")
```



참고 4

보고서 작성을 위한 마크다운 예시

참고 파일링크 : README.md

💡 여름방학 해성여자고등학교 캡스톤 탐구 프로젝트

📌 프로젝트 개요

- **프로젝트 기간**: 2025년 7월 22일 ~ 2025년 7월 25일
- **참여 학생**: 해성여자고등학교 여름방학 특별 탐구 활동

🎯 문제 인식

학교 정문 앞에 **킥보드가 무질서하게 많이 놓여 있어** 통행에 위험을 느끼는 일이 많았습니다.
이 문제를 인공지능 기술로 탐지하고 해결 방안을 모색하기 위해,
YOLO 객체 탐지 모델을 활용해 킥보드의 수를 세는 시스템을 만들어 보았습니다.

📂 학습 데이터 및 YOLO 모델

- 🔗 **데이터셋 출처 (Roboflow)**:
  https://universe.roboflow.com/your-workspace/kickboard-detection
- 📘 **YOLOv8 학습 코드 (Google Colab)**:
  https://colab.research.google.com/drive/your_colab_link

💬 추가 설명

YOLOv8을 사용하여 직접 킥보드 탐지 모델을 학습하였습니다.
이미지 전처리, Bounding Box 정제, mAP 성능 확인 등의 과정을 거쳤고,
소형 모델을 활용해 비교적 빠르고 정확하게 킥보드를 탐지할 수 있도록 설정하였습니다.

🖼️ YOLO 탐지 결과 비교

일반 YOLO 모델	학습한 킥보드 전용 YOLO 모델
일반 YOLO 결과	학습 모델 결과

🧪 실행 방법

아래 코드를 사용하여 원하는 이미지에서 킥보드를 탐지할 수 있습니다:

사례 연구 5. 급식 잔반량 분석

1. 문제 발견

- 매일 급식 후 남은 잔반량이 많은데, 어떤 반이 유독 많이 남기는지, 어떤 메뉴일 때 잔반이 많은지를 선생님이나 급식실이 파악하기 어렵다.

2. 해결책 제시

- 급식 후 반별로 식판 사진을 찍고, YOLO로 남은 음식 종류(밥, 국, 반찬 등)를 탐지하여 남긴 양을 시각적으로 확인한다

3. 해결책의 타당성

- 사람마다 "많이 남겼다"는 기준이 다르기 때문에, 객체 탐지를 이용한 사진 기반 분석이 더 객관적이다.

- YOLO는 밥, 국, 김치, 기타 반찬 등을 라벨링해서 인식할 수 있고, 클래스별 탐지 수를 통해 양을 추정할 수 있다.

4. 수행 결과 및 토의

- 김치가 남은 비율이 가장 높고, 반찬은 메뉴에 따라 잔반 편차가 컸다.

- 식판 위치나 조명에 따라 인식률 차이가 생겨 조정이 필요했다.

- 반별 차이를 분석해 식생활 지도 자료로 활용할 수 있을 것 같다.

사례 연구 6. 복도 내 정체 시간대 분석

1. 문제 발견

- 쉬는 시간마다 복도가 매우 붐벼서 부딪힘 사고가 자주 발생한다는 이야기가 있다.
- 어떤 시간대가 특히 혼잡한지 교사나 학생 모두 체감만 있을 뿐, 실제 데이터는 없다.

2. 해결책 제시

- 쉬는 시간마다 복도 CCTV 영상을 수집하고, YOLO를 통해 탐지된 '사람' 객체의 수를 시간대별로 분석하여 혼잡도를 정량화한다.

3. 해결책의 타당성

- 사람이 육안으로 확인하기엔 너무 많은 인원이 한꺼번에 움직이고, 시간대별 비교도 어렵다.
- YOLO는 빠르고 정확하게 사람 수를 세어줄 수 있어, 혼잡도를 수치로 비교할 수 있다.

4. 수행 결과 및 토의

- 오전 10:40 쉬는 시간이 전체 시간 중 가장 많은 인원이 복도에 있었다.
- 일부 영상은 카메라 위치상 사각지대가 있어 정확한 수치 추정에 어려움이 있었다.
- 다음에는 여러 대의 카메라를 활용하거나, 센서와 함께 사용하는 방법도 고려해보고

사례 연구 7. 버스 승하차량 자동 파악 시스템

1. 문제 발견

- 통학 시간대에 학생들이 어떤 정류장에서 주로 타고 내리는지에 대한 데이터가 부족하다.
- 특히 늦게 오는 학생들이 어떤 노선을 이용하는지도 파악이 어려움

2. 해결책 제시

- 학교 근처 정류장에서 영상을 촬영하고, 버스와 사람 객체를 YOLO로 탐지하여 승하차 장면을 기록한다.
- 특정 시간대, 특정 노선에서의 이용 패턴을 파악할 수 있다

3. 해결책의 타당성

- 사람 수를 수작업으로 세는 것보다 YOLO 모델을 통해 자동으로 감지하고 시간과 위치를 기록하면 분석이 훨씬 쉬워진다.
- 방학 중 학습한 사람/버스 객체 탐지 기술을 활용하면 구현 가능하다.

4. 수행 결과 및 토의

- 오전 7:20~7:40 사이, OO 정류장에서 승차하는 학생이 가장 많았다.
- 인원 수 세기에는 성공했지만, '승차'와 '하차'를 구별하기 위해서는 더 많은 전후 영상 분석이 필요했다.
- 다음에는 객체의 방향 정보도 고려한 추론이 가능하도록 해보고 싶다.

사례 연구 8. 학교 주변 조류 출현 빈도 분석

1. 문제 발견

- 일부 교실 창문에 새가 부딪히는 사고가 종종 있었고, 특정 장소에 새가 자주 모여드는다는 학생들의 증언이 있었다.
- 조류 출현 지역을 알 수 있다면 안전 대책을 마련할 수 있다.

2. 해결책 제시

- 학교 외벽 주변에 카메라를 설치하고, 일정 시간마다 촬영한 사진에서 새 객체를 YOLO로 탐지하여 출현 위치를 기록한다.
- 출현 위치를 지도 형태로 시각화해 위험 지역을 파악한다.

3. 해결책의 타당성

- 사람의 관찰은 새가 날아다니는 시간과 장소를 모두 커버할 수 없다.
- YOLO는 새 객체 탐지가 가능하며, 시간대/장소별로 비교 분석이 용이하다.

4. 수행 결과 및 토의

- 주로 오전 6시~7시에 학교 동쪽 창문 근처에서 새가 가장 많이 출현함이 확인되었다.
- 낮은 해상도에서는 먼 거리의 작은 새는 잘 탐지되지 않아 정확도가 떨어졌고, 이를 보완하기 위한 렌즈 변경 또는 거리 측정이 필요했다.

사례 연구 9. 흡연 의심 구역 모니터링

1. 문제 발견

- 학교 담장 뒤쪽 골목이나 화장실 근처에 사람들이 오래 머무르는 경우가 많고, 그중 일부가 흡연 중이라는 민원이 있다.
- 흡연 장면을 직접적으로 촬영하기 어려운 경우가 많다.

2. 해결책 제시

- 특정 구역에서 사람 객체가 얼마나 자주, 얼마나 오래 머무르는지를 YOLO로 탐지하여 이상 행동 여부를 추정한다.
- 오래 머무르는 사람 수를 기록하여 이상 징후 감지를 시도한다.

3. 해결책의 타당성

- 흡연을 직접 감지하는 것은 어렵지만, 특정 장소에 장시간 머무르는 행동은 감시 가능하다.
- YOLO는 사람 객체를 시간대별로 정확히 탐지할 수 있고, 체류 시간 분석도 가능하다.

4. 수행 결과 및 토의

- 오후 4시~6시에 특정 화단 근처에 사람 객체가 지속적으로 감지됨을 확인함.
- 사람 수는 적었지만 평균 체류 시간이 높아, 해당 장소를 관리 대상으로 제안했다.
- 아쉬운 점은 영상에서 행동 자체(흡연 등)를 직접 확인할 수 없다는 점이었다.

사례 연구 10. 헬멧 미착용 탐지 시스템

1. 문제 발견

- 일부 학생들이 자전거나 전동 킥보드를 탈 때 헬멧을 착용하지 않아 안전사고 위험이 있다.
- 교문 앞에서 일일이 확인하기 어렵고, 규칙이 잘 지켜지는지도 파악되지 않고 있다.

2. 해결책 제시

- 등하교 시간대 교문 앞 영상을 촬영하고, YOLO를 이용하여 사람과 헬멧 객체를 동시에 탐지한다.
- 헬멧이 감지되지 않은 경우를 "미착용"으로 기록해 분석한다.

3. 해결책의 타당성

- 헬멧 착용 여부는 멀리서도 시각적으로 구분 가능하고, YOLO는 복합 객체 탐지가 가능하다.
- 탐지 결과를 근거로 교육 자료나 계도 활동 자료로 활용 가능하다..

4. 수행 결과 및 토의

- 킥보드 이용자의 약 30%가 헬멧을 착용하지 않은 것으로 확인되었다.
- 탐지 정확도는 비교적 높았지만, 모자와 헬멧이 혼동되는 경우가 있었고, 각도에 따라 탐지 누락이 발생했다.
- 다음에는 다양한 각도에서의 데이터로 보완하거나, 후처리 알고리즘을 개선하고 싶다.

Q & A

감사합니다