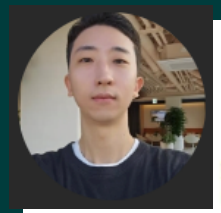
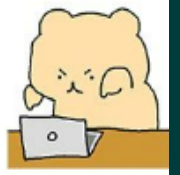
AI 생활 폐기물 분류

TEAM 1조 SHREGI조

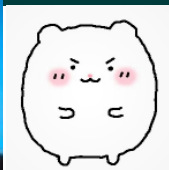
서중원, 윤종모, 서지원, 조현형, 조진우, 김희정

SHREGI Team Members


서중원 (팀장)
데이터 수집
데이터 전처리
학습 데이터셋 구축

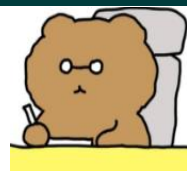
윤종모
데이터 수집
데이터 전처리
모델 학습

서지원
데이터 수집
데이터 전처리
학습 데이터셋 구축

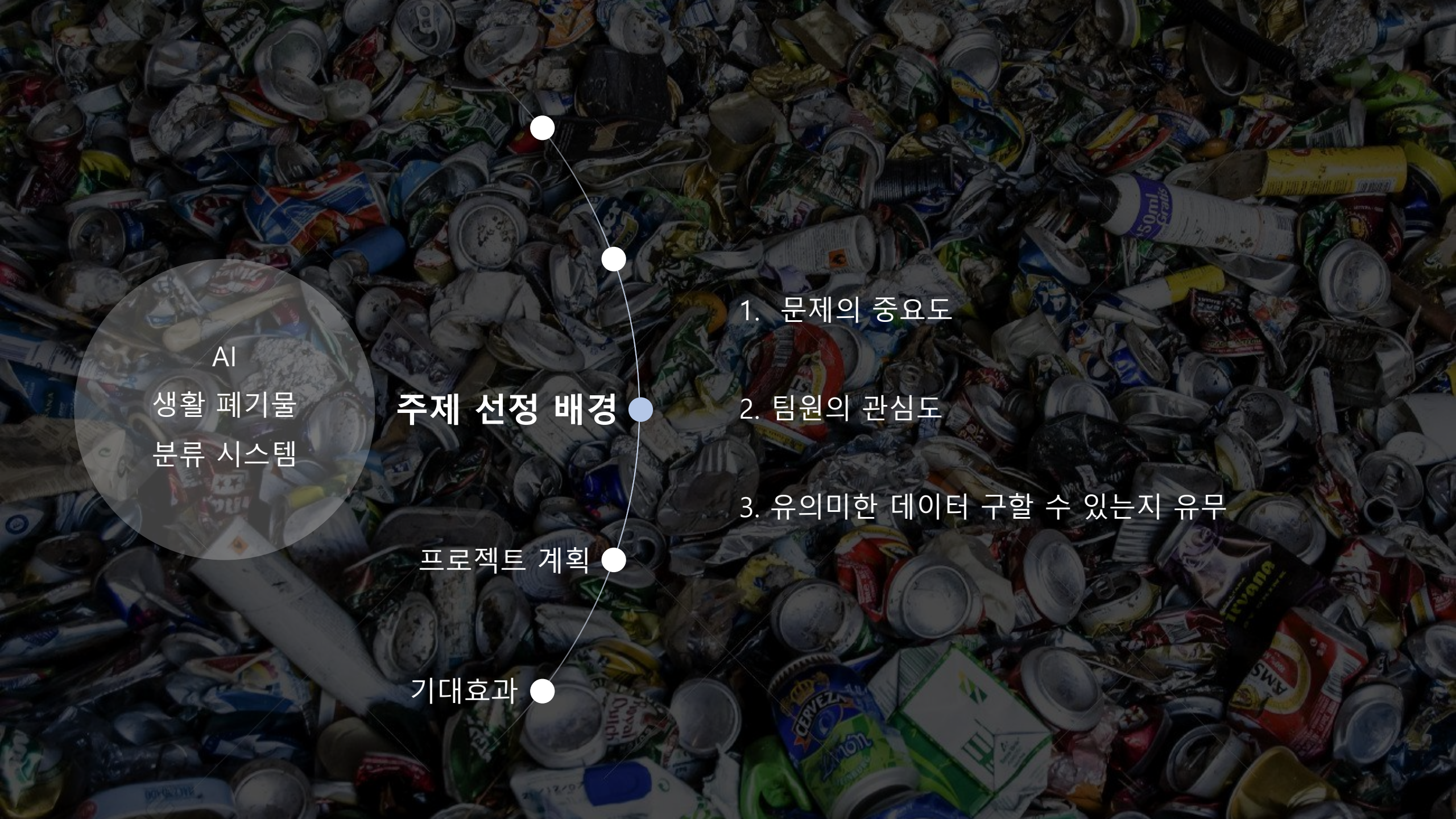

김희정
데이터 수집
데이터 전처리
웹 구현


조진우
데이터 수집
데이터 전처리
웹 데모 구현

조현형
데이터 수집
데이터 전처리
모델 학습

Contents



AI
생활 폐기물
분류 시스템

주제 선정 배경

1. 문제의 중요도
2. 팀원의 관심도
3. 유의미한 데이터 구할 수 있는지 유무

프로젝트 계획

기대효과

AI
생활 폐기물
분류 시스템

주제 선정 배경

프로젝트 계획

기대효과

인천시 "2025년부터 수도권매립지에 서울·경기 쓰레기 못 받는다" 선언
서울·경기 강력 반발에도 '발생지 처리' 내세우는 인천시 입장 강경
매립지 포화로 전국 곳곳서 '쓰레기 대란' 우려…"더 늦기 전에 대책 서둘러야"


인천 서구에 있는 수도권 매립지의 쓰레기. 경기일보DB

VIEW 5,848 | 2020.10.25 11:02

"물량 감당 안된다"...재활용 폐기물 늘어 발동동

AI
생활 폐기물
분류 시스템

주제 선정 배경

프로젝트 계획

기대효과

[Object Detection]

딥러닝 모델을 활용한 이미지 분류 시스템 설계

데이터 수집, 가공
(AI-Hub, GitHub)

모델 학습
(Yolov8m & Yolov8s 모델 사용)

프로토타입 및 시스템 구현
(Streamlit)

실제 환경에서의 테스트 및 개선


주제 선정 배경
프로젝트 계획
기대효과
AI
생활 폐기물
분류 시스템
수도권 쓰레기 문제 심각성 알림
재활용률을 높여 탄소발자국 저감
쓰레기 처리(매립, 소각) 비용 절감
도시환경 지속 가능성에 기여

Contents

기간	구분	활동	비고
10.15 ~ 10.20 (6 Days)	**사전기획**	▪ 프로젝트 기획 및 주제선정 ▪ 데이터 리서치 ▪ 기획안 작성	-
10.21 ~ 10.23 (3 Days)	**데이터 수집**	▪ 필요 데이터 수집	▪ AI Hub
10.23 ~ 11.03 (12 Days)	**데이터 전처리**	▪ 데이터 가공 및 정제	▪ 클라우드에 데이터 공유 ▪ 폴더 디렉토리 통일 ▪ 폴더 및 파일명 영문 변환
10.30 ~ 11.13 (15 Days)	**모델 학습 및 데모 프로그램 제작**	▪ 모델 선정 ▪ 학습을 위한 데이터셋 구축 ▪ 데모 프로그램 제작	▪ YOLOv8 ▪ Streamlit
11.10 (1 Days)	**시각화**	▪ 결과 분석 및 시각화	-
11.06 ~ 11.13 (6 Days)	**시스템 구현**	▪ 시스템 구현 및 업데이트	▪ Streamlit
프로젝트 총 기간 ▶ 10.15 ~ 11.14 (4주)			

Contents

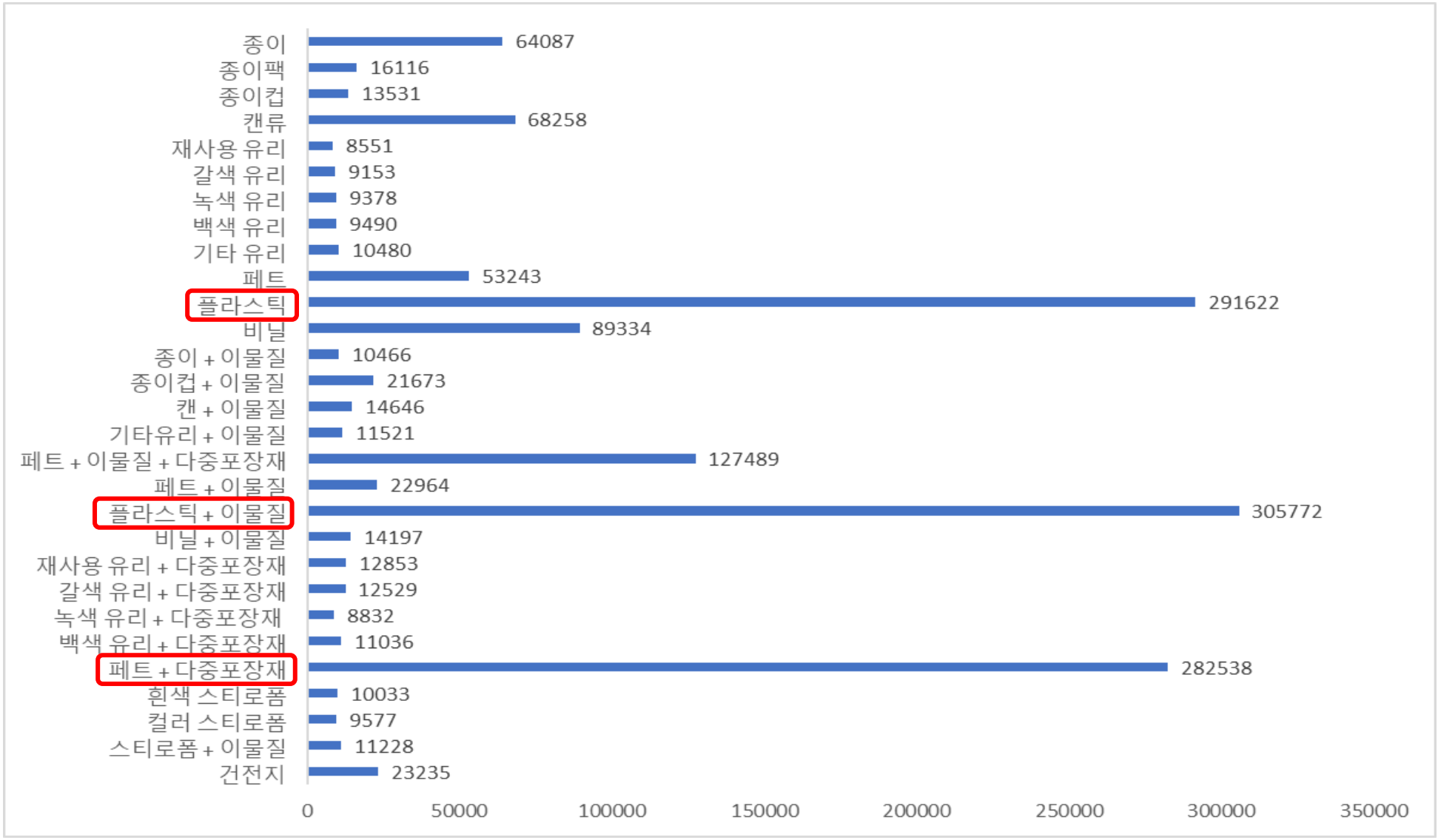
03-01 [Object Detection] 데이터 수집

데이터 출처 [AI Hub – 생활폐기물 데이터 활용·환류]

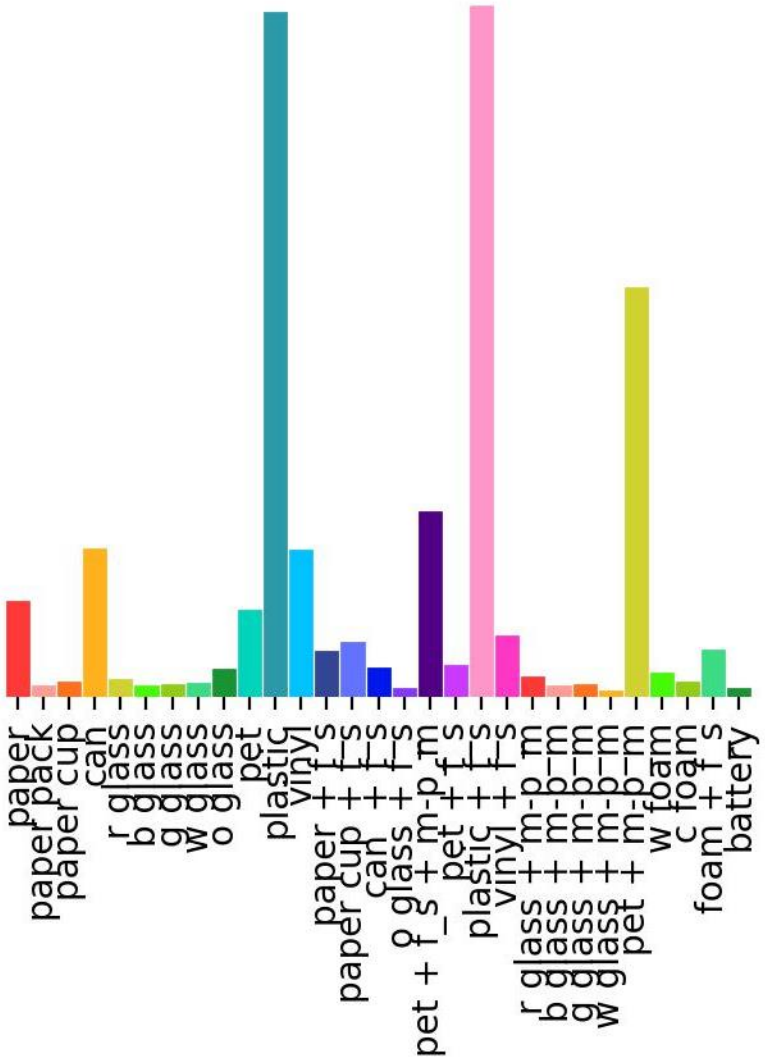
Data Info				
Image	Label		Number of Classes	29
Format	Format	Type		
jpg	json	bounding box		

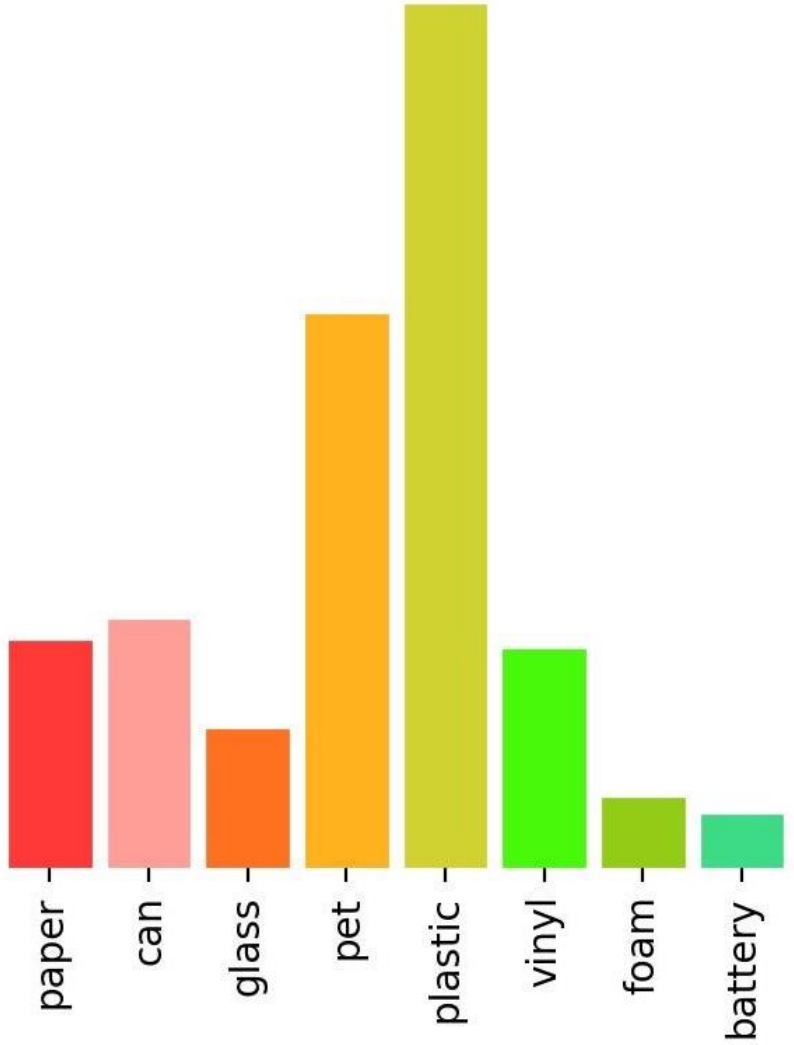

```
"objects": [
    {
        "id": "3f7e71c6-7e85-400e-a84b-5aa1f75b206d",
        "class_id": "3cb81173-41f1-4cc4-8415-dc3b33dc14b4",
        "tracking_id": 1,
        "class_name": "c_5_01",
        "annotation_type": "box",
        "annotation": {
            "coord": {
                "x": 390.4948878627418,
                "y": 0,
                "width": 118.42453864286335,
                "height": 104.52460175368465
            },
            "meta": {
                "z_index": 1,
                "visible": true,
                "alpha": 1,
                "color": "#76C923"
            }
        },
        "properties": []
    },
```

03-01 [Object Detection] 데이터 분포



03-02 [Object Detection] 데이터 전처리 - 클래스 통합





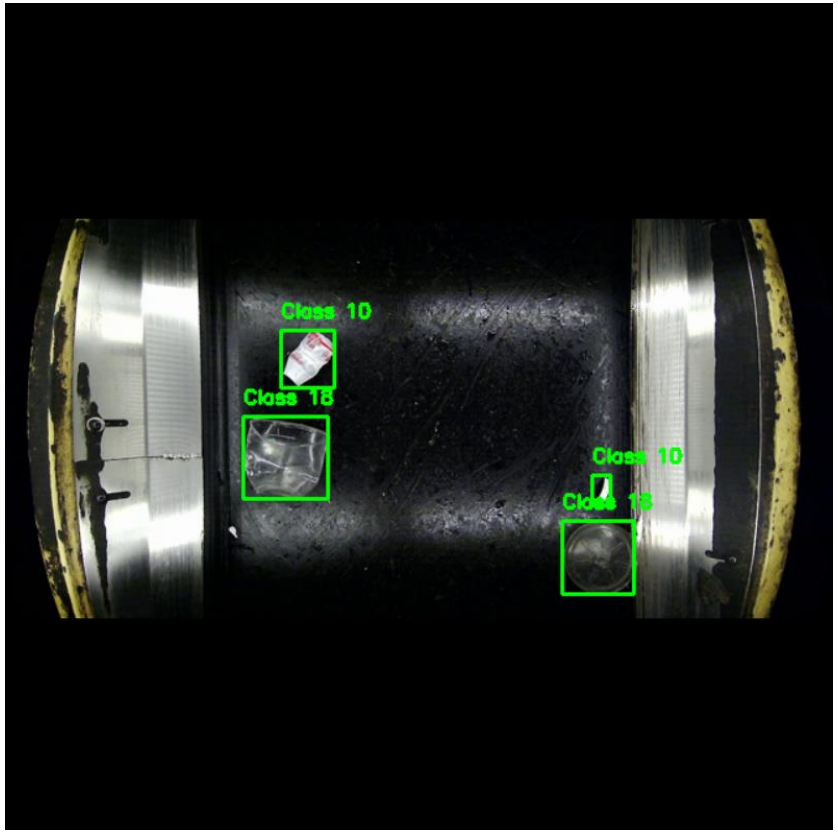
03-02 [Object Detection] 데이터 전처리

```python
original_width, original_height = 1847, 883
target_size = 640
x_scale = original_width / target_size
padding_size = (target_size - (original_height / x_scale)) / 2
```

Target size / Scale / Padding size

```python
for obj in objects:
    annotation = obj.get('annotation', {})
    coord = annotation.get('coord', {})
    x = int(coord.get('x', 0.0) / x_scale)
    y = int(coord.get('y', 0.0) / x_scale) + padding_size
    width = int(coord.get('width', 0.0) / x_scale)
    height = int(coord.get('height', 0.0) / x_scale)
    class_id = obj.get('class_name', "Unknown")

    center_x = (x + width / 2) / target_size
    center_y = (y + height / 2) / target_size
    normalized_width = width / target_size
    normalized_height = height / target_size

    # Write YOLO-formatted data to the .txt file
    yolo_txt_file.write(f"{class_id} {center_x} {center_y} "
                        f"{normalized_width} {normalized_height}\n")
```

Bounding Box 좌표/ scale 변환 / txt파일 생성




```
18 0.34140625 0.5476824749593936 0.1015625 0.0984375
18 0.71640625 0.6656512249593936 0.0859375 0.0875
10 0.36796875 0.4281512249593936 0.0640625 0.06875
10 0.71875 0.5867449749593936 0.021875 0.0359375
```

이미지에 존재하는 객체의
Class id와 bbox 정보가 저장된 txt파일

03-02 [Object Detection] 데이터 전처리 - 리사이즈, 패딩

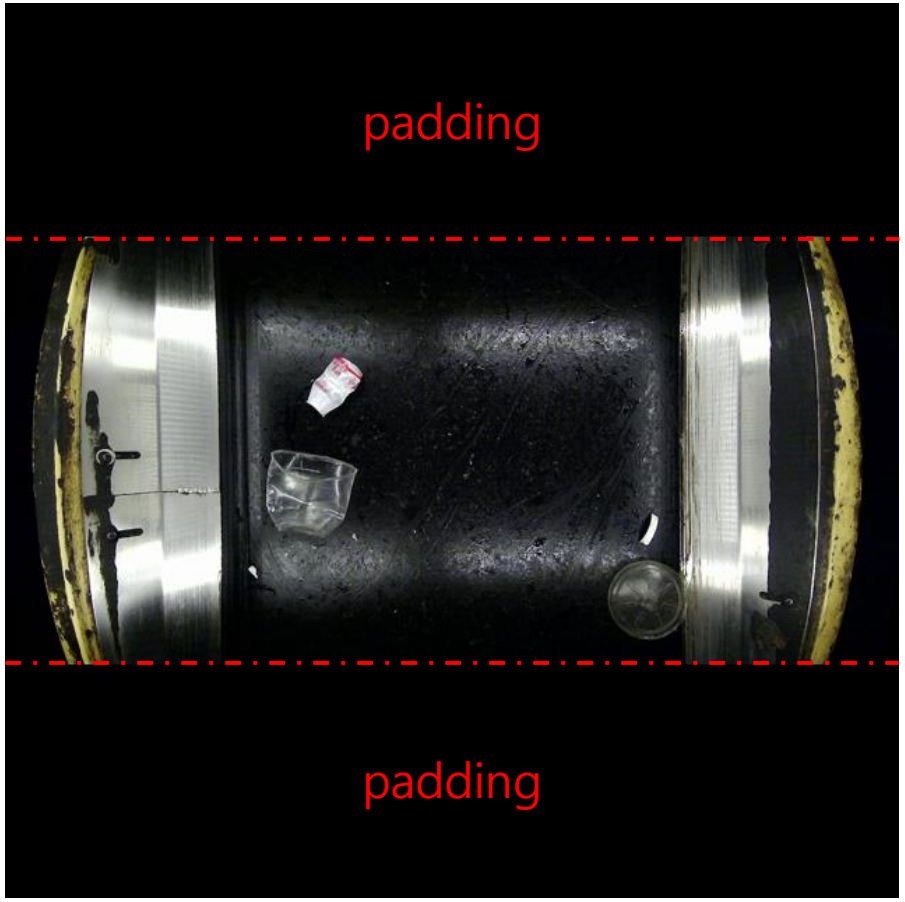
```python
from PIL import Image
import os

# JPG path & save PNG path
jpg_dir = './data/Training/image/TS_A2'
png_dir = './data_trash/train/image/A2'

target_size = (640, 640)

if not os.path.exists(png_dir):
    os.makedirs(png_dir)

for jpg_file in os.listdir(jpg_dir):
    if jpg_file.endswith(".jpg"):
        img = Image.open(os.path.join(jpg_dir, jpg_file))
        img.thumbnail(target_size)

        new_img = Image.new("RGB", target_size, (0, 0, 0))
        left = (target_size[0] - img.width) // 2
        top = (target_size[1] - img.height) // 2
        new_img.paste(img, (left, top))

        png_file = os.path.splitext(jpg_file)[0] + ".png"
        new_img.save(os.path.join(png_dir, png_file), "PNG")

print("Conversion and resizing with padding completed.")
```

resize

padding

PNG 변환

원본 JPG 이미지



학습 네트워크의 input 형태에 알맞게
정사각형 Resize / Padding / png 변환

03-03 [Object Detection] YOLOv8 선정

모델 성능비교 출처: https://learnopencv.com/ultralytics-yolov8/

YOLO 알고리즘

2016년에 처음 등장한 객체탐지 알고리즘으로, 최근 YOLOv8 버전이 공개
One-Stage 검출기로서 객체의 위치를 인식하고 클래스 분류를 하나의 과정으로 수행
기존 객체탐지 알고리즘과 비교하여 정확도는 거의 대등하나 연산속도가 월등하여 실시간 객체탐지에 유리

Object Detection Performance Comparison (YOLOv8 vs YOLOv5)

Model Size	YOLOv5	YOLOv8	Difference
Nano	28	37.3	+33.21%
Small	37.4	44.9	+20.05%
Medium	45.4	50.2	+10.57%
Large	49	52.9	+7.96%
Xtra Large	50.7	53.9	+6.31%

*Image Size = 640

**작은 객체 탐지와 겹침 객체 처리에 한계가 있지만,
실시간 영상에 적합한 빠른 연산속도를 갖는 YOLOv8s 선택**

03-04 [Object Detection] YOLOv8 모델 학습 - 결과

학습 회차별 데이터셋 정보

학습 회차별 성능지표

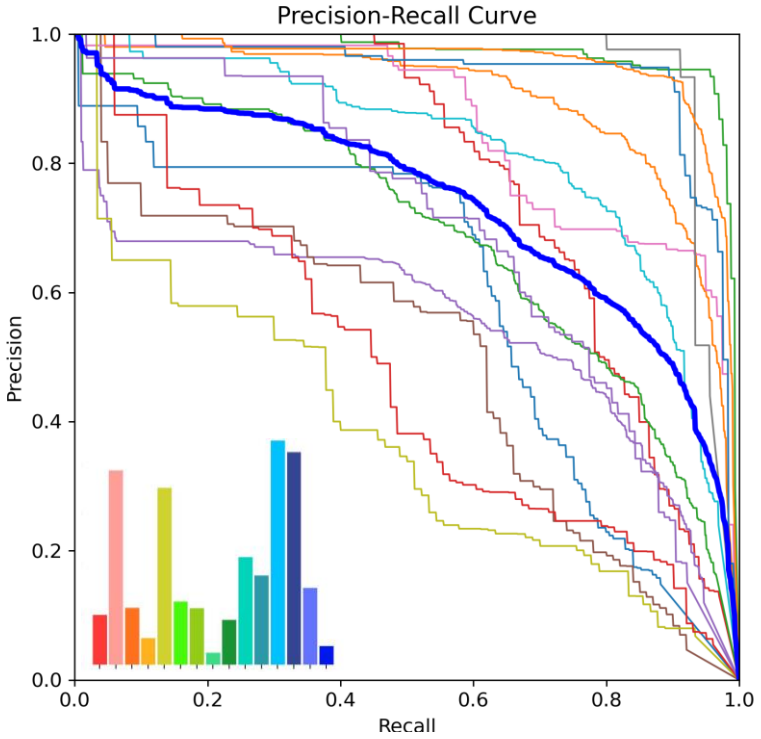
Train	Model	Image Size	Dataset Volume	Classes	Note	Precision (1.00 at)	Recall (at 0.00)	F1 Score	mAP@50
1st	yolov8m	640	9,000	15	최초 데이터셋.	1.00	0.95	0.74	0.791
2nd	yolov8m	640	9,000	15	부족한 클래스(비닐+이물질)의 데이터 보충. | cf. 1st	1.00	0.93	0.70	0.735
3rd	yolov8m	640	9,000	11	데이터셋 및 클래스 재구성.	0.991	0.96	0.81	0.868
4th	yolov8s	640	9,000	11	모델 변경 yolov8m → yolov8s | cf. 3rd	0.993	0.95	0.80	0.867
5th	yolov8s	640	9,000	8	재활용 물체만 객체로 가정하여 클래스 구성.	0.984	0.96	0.84	0.888
6th	yolov8s	640	38,000	8	데이터셋 규모를 크게 증가. | cf.5th	0.979	0.99	**0.91**	**0.957**
7th	yolov8s	640	38,000	16	클래스 분화 8 → 16 (이물질, 병의 색상) | cf. 6th	0.988	0.96	0.79	0.841
8th	yolov8s	1280	9,000	11	픽셀 증량 640 → 1280 | cf. 4th	0.993	0.97	0.82	0.882
9th	yolov8s	1280	9,000	8	픽셀 증량 640 → 1280 | cf. 5th	0.992	0.97	0.86	0.911

* 100 Epochs 및 Best.pt 기준.

학습 회차별 비교

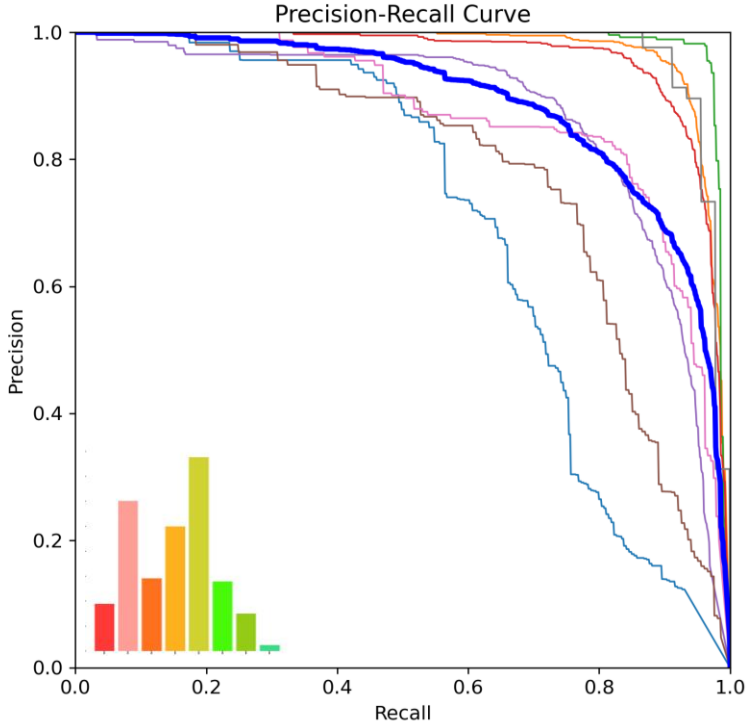
- 1st vs 2nd : 작은 데이터셋, 동일 해상도, 부족한 라벨의 데이터 보충에 따른 차이(라벨 15개), 모델 m.
- 3rd vs 4th : 작은 데이터셋, 동일 해상도, 라벨 11, 모델 m->s 따른 성능 차이. 크지 않음을 확인. 이후 모델 s 채택.
- 4th vs 8th : 작은 데이터셋, 해상도 차이, 라벨 11, 모델 s
- 5차 vs 9차 : 작은 데이터셋, 해상도 차이, 라벨 8, 모델 s
- 5차 vs 6차 : 데이터셋 규모 차이, 동일 해상도, 라벨 8, 모델 s
- 6차 vs 7차 : 큰 데이터셋, 동일 해상도, 라벨 차이(8vs16), 모델 s

03-04 [Object Detection] YOLOv8 모델 학습 - 결과 분석

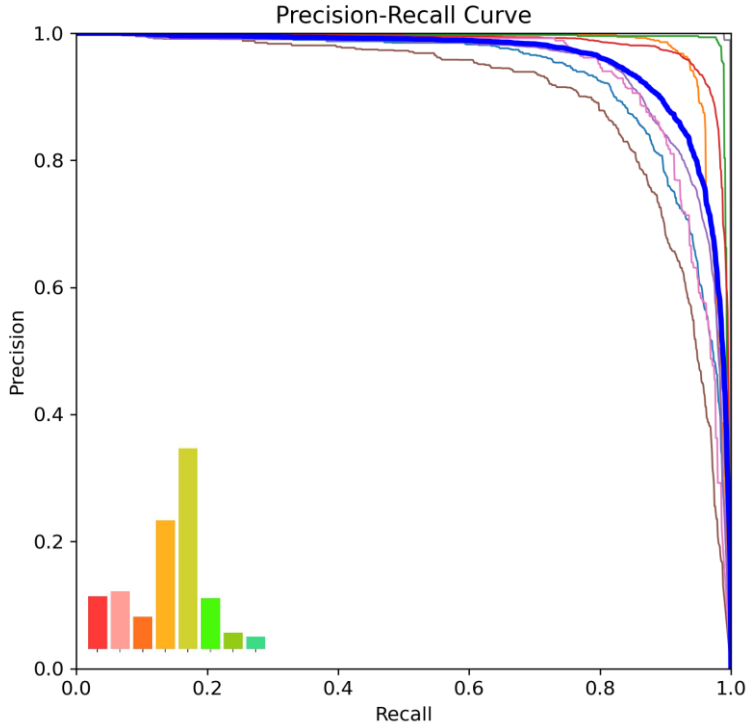


paper 0.593
can 0.896
glass 0.971
pet 0.781
plastic 0.551
vinyl 0.491
foam 0.856
battery 0.951

paper + f_s 0.370
can + f_s 0.815
o glass + f_s 0.937
pet + f_s + m-p_m 0.954
plastic + f_s 0.687
vinyl + f_s 0.473
foam + f_s 0.694
all classes 0.735 mAP@0.5



paper 0.700
can 0.970
glass 0.983
pet 0.959
plastic 0.869
vinyl 0.783
foam 0.870
battery 0.971
all classes 0.888 mAP@0.5



paper 0.931
can 0.976
glass 0.991
pet 0.982
plastic 0.945
vinyl 0.901
foam 0.939
battery 0.995
all classes 0.957 mAP@0.5

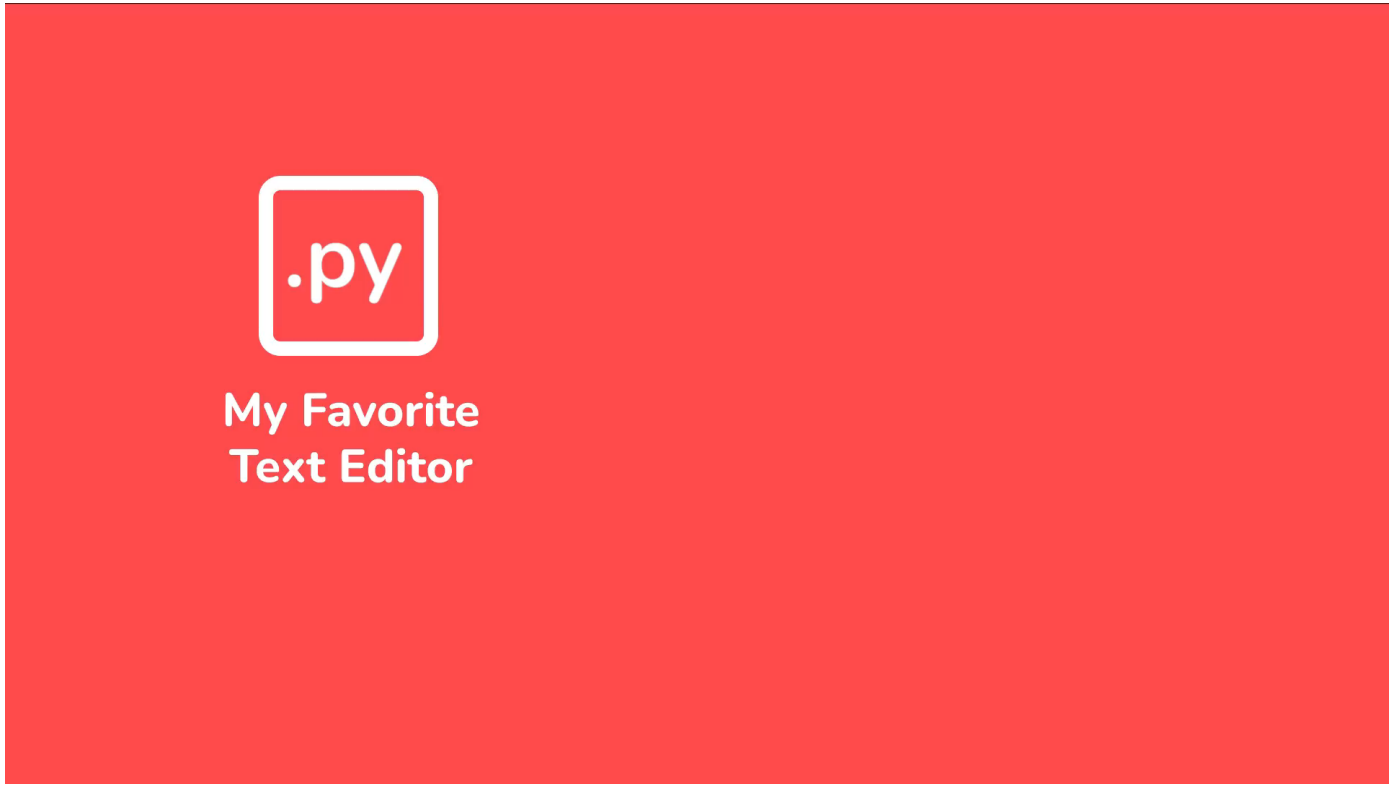
03-05 [Object Detection] Streamlit

Streamlit 소개 :

Streamlit

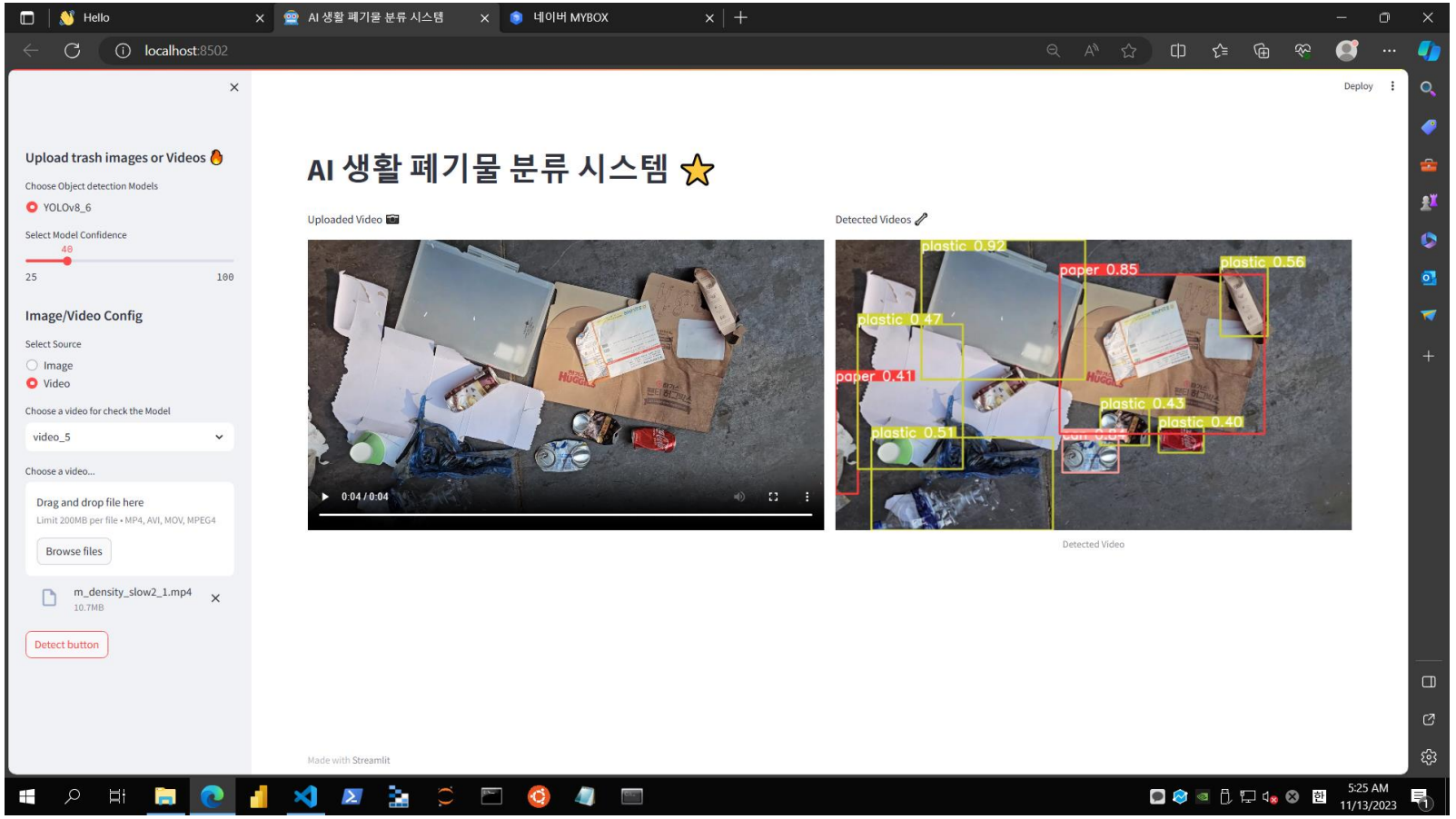
streamlit.io

https://wastedetection-ms3.streamlit.app/



03-05 [Object Detection] Streamlit 시연

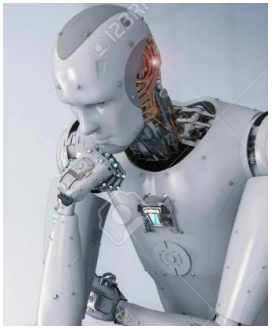
Streamlit 사용 시연 :



Contents

04-01 학습 결과의 한계

정답 라벨



모델이 예측한 라벨

고민

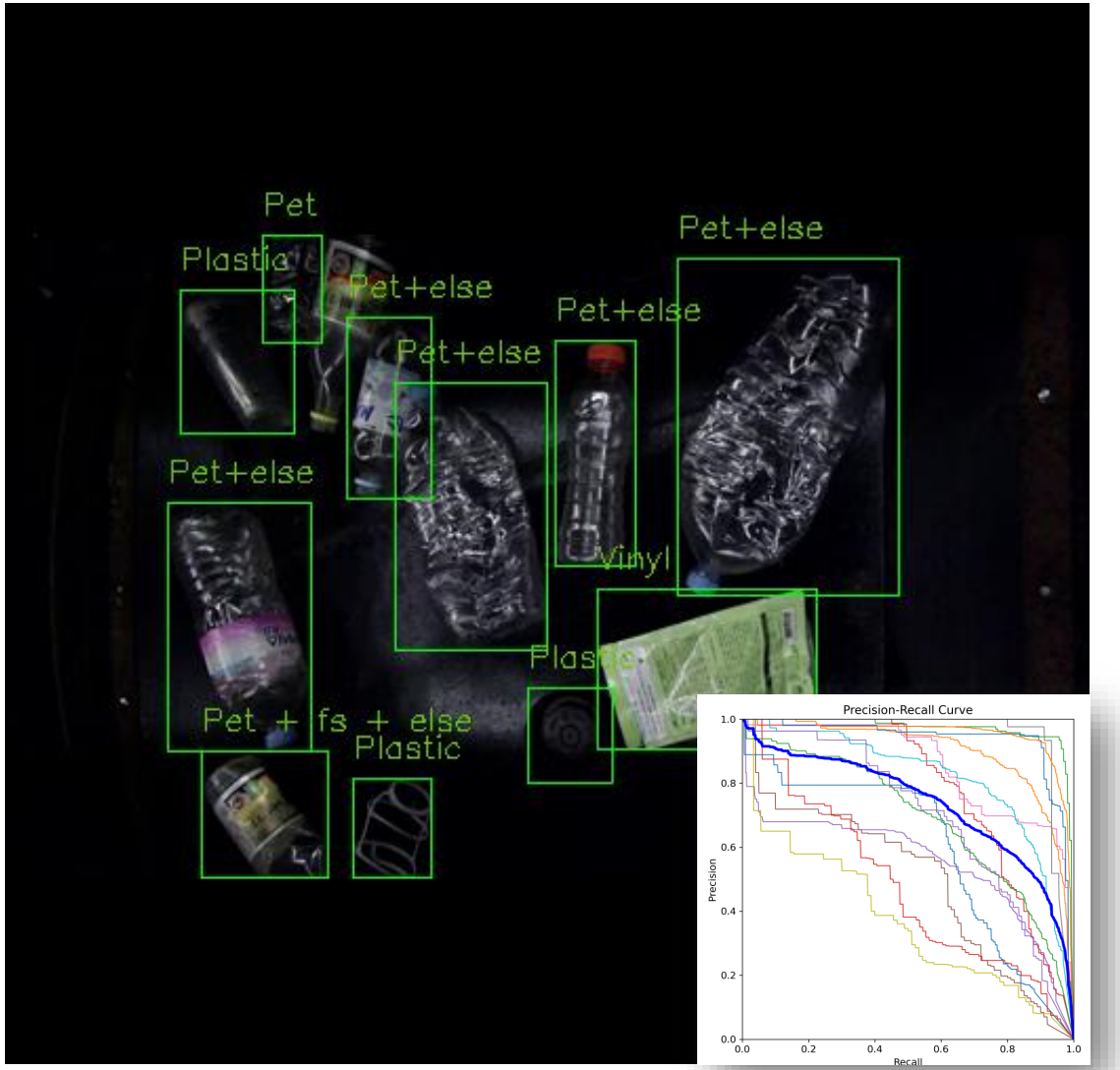
시도

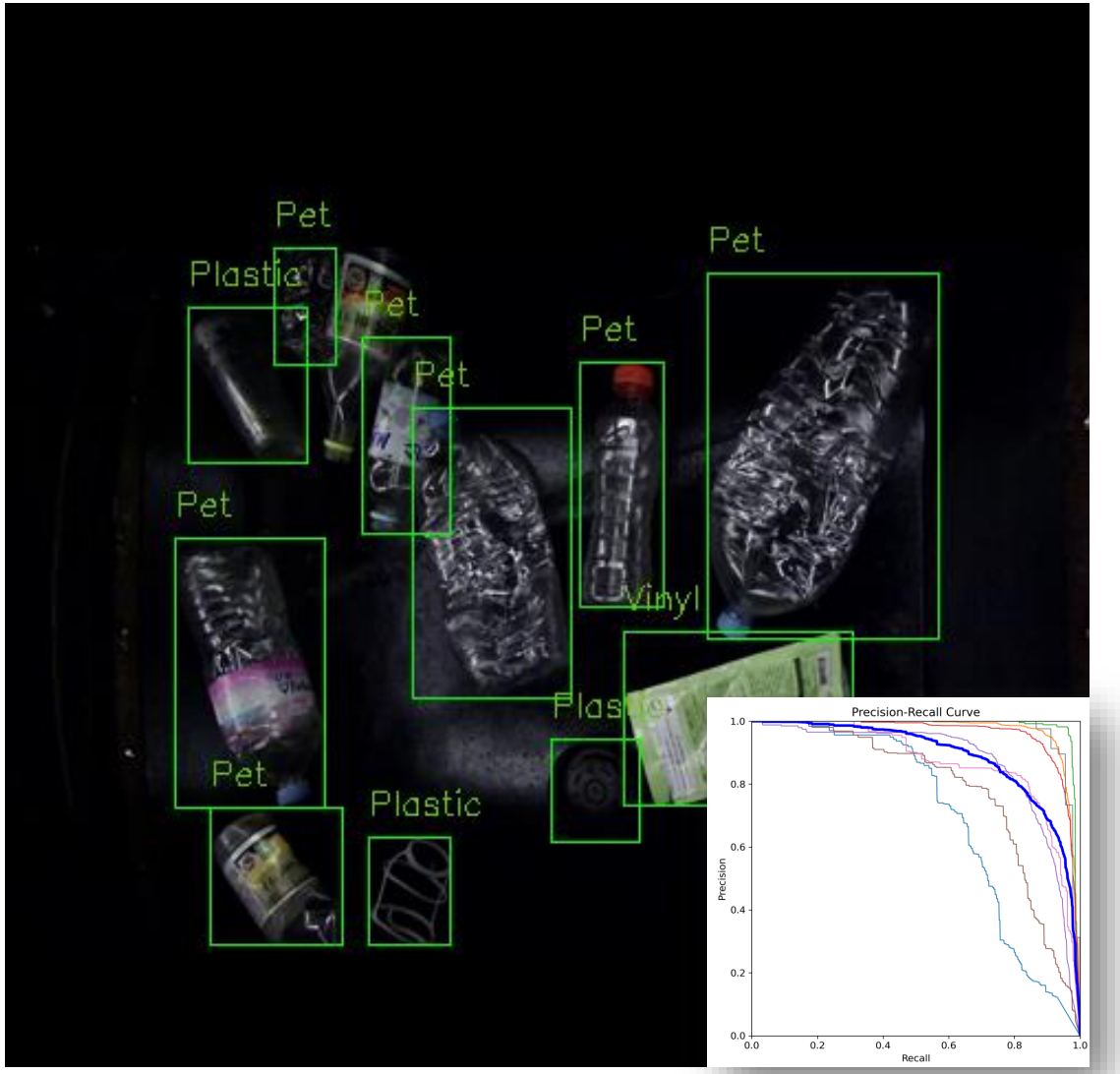
데이터수를 더 늘릴까?

클래스를 얼마나 줄일까?

학습시키는 **이미지 해상도**를 높일까?

04-02 - 성능 개선 과제


Pet
Plastic
Pet+else
Pet+else
Pet+else
Pet+else
Pet+else
Vinyl
Pet + fs + else
Plastic
Precision-Recall Curve
Precision
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


Pet
Plastic
Pet
Pet
Pet
Pet
Vinyl
Pet
Plastic
Precision-Recall Curve
Precision
Recall
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

프로젝트 자체 평가

한계점

- 겹치는 형태의 쓰레기는 분류하기에 어려움
- 데이터 수를 늘리고, 1280pixels의 고화질 이미지를 학습 시켰을 때,
  성능의 향상은 있었으나 유의미하게 개선하지는 못함
- 사람의 눈으로 빈 페트병이나, 비닐 등의 구분은 쉽게 가능하나,
  객체를 탐지할 때는 재질의 특성을 구분하기 어려워 혼동할 수 있어 아쉬웠음.

개선점

- 현장에서 활용될 수 있도록, 이물질 및 색을 이진 분류하는 모델을 추가 (Two-Stage 모델 사용)

Q & A


감사합니다