[최종보고서 - 2팀]

▼ 0. 목차

▼ 1. 과제 개요 및 목표

- **프로젝트명:** [AI07] 경구약제 이미지 객체 검출(Object Detection) 프로젝트
- **기간: 2026**.01.29 ~ 2026.02.23 (총 13일)
- **구성:** 팀장(김예주), 팀원(김정우, 박윤민, 임운하, 최지훈) (총 5인)
- **역할:**
    - PM: 김예주
    - Data Engineer: 최지훈, 박윤민
    - Model Architect: 임운하(Faster R-CNN), 김정우(Yolo v8)
    - Experimentation Lead: 김예주, 최지훈
- **목표:**
    - Object Detection 모델 구축
    - 안정적인 추론–후처리–제출 파이프라인 구축
    - Kaggle 평가 지표(mAP) 기준 성능 개선 달성

▼ 2. 작업 환경 및 협업 환경 설정

1) 작업 및 협업 환경 통일

- Python 3.13과 conda 환경 기준으로 통일함
- GPU 사용은 선택 사항으로 두며, GPU 없는 환경에서도 실행 가능하도록 통일함

2) Github 협업 체계 구축

- Branch 관리 및 PR 사용 등 협업 체계 구축함

3) 코드 모듈화 및 재사용성 확보

▼ 기능별 코드 분리를 통한 유지보수 효율성 및 실험 재현성 향상 시킴

- 모듈화 구조

```
medicine/
├── configs/                    # 설정 관리
│   ├── paths.yml               # 데이터 경로 (상대경로)
│   └── load_paths.py           # yml → 절대경로 변환
├── imports.py                  # 공통 라이브러리 import 모음
│
├── preprocessing/              # 데이터 전처리
│   ├── coco_data_integration.py
│   ├── yolo_coco_preprocessing.py
│   ├── yolo_converter.py
│   └── data_inventory_check.py
│
├── dataset/                    # 데이터셋 클래스
│   └── faster_rcnn/rcnn_dataset.py
│
├── model/                      # 모델별 학습 & 추론
│   ├── faster_rcnn/
│   │   ├── rcnn_model.py       # 모델 생성
│   │   ├── rcnn_train.py       # 학습
│   │   └── rcnn_inference.py   # 추론 → CSV
│   └── yolov8s/
│       ├── yolov8s_model.py    # 학습
│       └── yolo_inference.py   # 추론 → CSV
│
├── runner/                     # 실행 진입점 (파이프라인)
│   ├── main_rcnn.py            # RCNN 학습/추론 통합 실행
│   └── main_yolo.py            # YOLO 학습/추론 통합 실행
│
├── outputs/                    # 학습된 모델 저장
└── submit/                     # 제출용 CSV 저장
```

- 모듈화 핵심 설계

1. **경로 중앙 관리**
   - `paths.yml`에 경로 모음
   - `load_paths.py`에서 yml → 절대경로 변환함
   - 모든 모듈에서 import하여 사용함
   - 데이터 소스 경로는 `paths.yml`로 중앙 관리함
   - 출력 경로(모델 저장, CSV 등)는 각 모듈에서 직접 관리함
   - 향후 개선 시 출력 경로도 yml에 통합 가능함
2. **공통 import 분리**
   - `imports.py`에 공용 라이브러리 모음
   - 각 파일에서 `from imports import *`로 사용함
3. **모델별 분리**
   - 학습(train), 추론(inference), 모델 정의(model) 파일 분리함
4. **runner로 통합 실행**
   - 하이퍼파라미터를 runner에서 설정함

- 학습→추론 파이프라인 구축

- **전체 실험 흐름**

1. 1단계: 데이터 전처리 (preprocessing/)

   [원본 COCO 데이터 → 통합 → 전처리 → 학습] 가능한 형태로 가공함

2. 2단계: 데이터 증강 — 희귀 클래스 오버샘플링

   notebooks/이미지 합성.ipynb

3. 3단계: 모델 학습 (실험마다 반복)

   실험 이름, epoch, lr, 증강 사용 여부 등 하이퍼파라미터 수정 후 runner 실행함

   매 epoch마다 mAP 측정, best 성능 갱신 시 모델 자동 저장함

   - RCNN → outputs/rcnn/모델명.pth
   - YOLO → outputs/yolo/yolov8_pill_날짜_시간/weights/best.pt

모델	실행 방법	하이퍼파라미터 위치
Faster R-CNN	`python runner/main_rcnn.py`	`runner/main_rcnn.py 상단`
YOLOv8s	`python runner/main_yolo.py`	`model/yolov8s/yolov8s_model.py`

4. 4단계: 추론 & CSV 생성

   outputs에서 최신 best 모델 자동 탐색함 (또는 경로 직접 지정)

   테스트 이미지 추론 → submit 폴더에 CSV 자동 생성함

모델	실행 방법	주요 설정
Faster R-CNN	`python model/faster_rcnn/rcnn_inference.py`	`MODEL_PATH, CONF_THRESHOLD`
YOLOv8s	`python model/yolov8s/yolo_inference.py`	`model_path, conf`

▼ 3. EDA 및 데이터 전처리

1) EDA 결과

1. 데이터 현황

- **Train 이미지:** 232장
- **Train annotation:** 763개
- **Test 이미지:** 824장

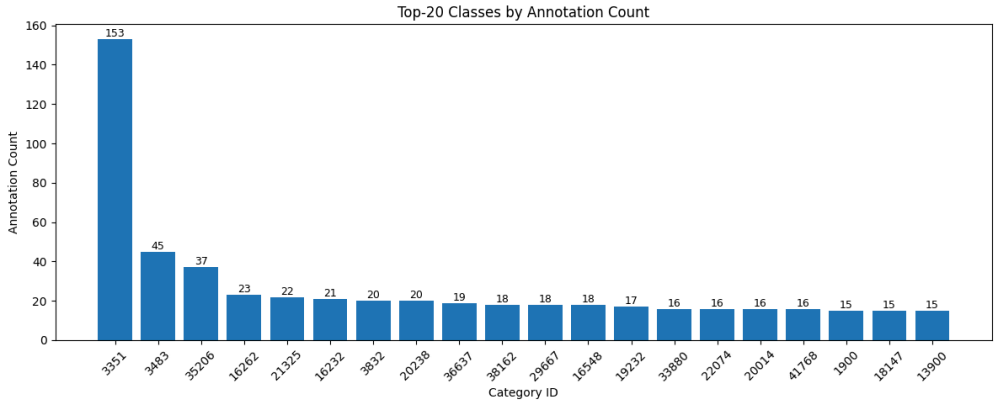
Train 이미지 1장당 평균 **3.29개의 객체**가 존재하며, COCO 형식 JSON으로 제공됨.

제출 CSV 파일은 `image_id` 기준 평균 2~4개의 row가 생성되는 구조임.

이미지당 객체 수	이미지 수
2개	7
3개	151
4개	74

2. 클래스 분포 분석

- 총 클래스 수: **56개**
- 분포 형태: **Long-tail 구조 (데이터 불균형 존재)**

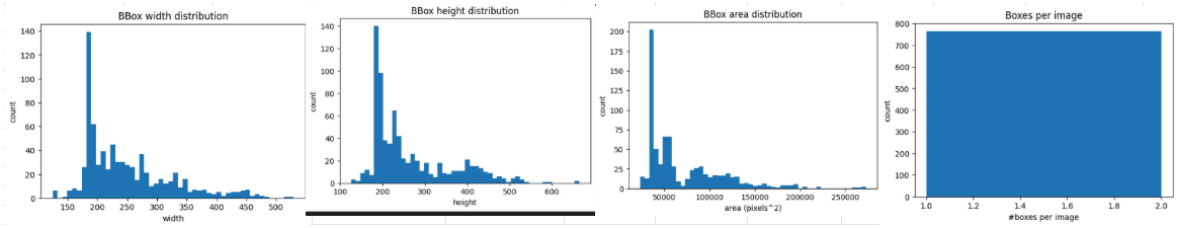


- 가장 많은 클래스:
  - 일양하이트린정 2mg (153)
  - 기넥신에프정(은행엽엑스)(수출용) (45)
  - 아토젯정 10/40mg (37)
- 희귀 클래스(≤10회 등장):
  - **29개 (전체의 51.8%)**
  - 최소 3회 등장 클래스: 17종

  → 전체 클래스의 절반 이상이 희귀 클래스에 해당하여 **모델의 일반화 및 소수 클래스 성능 저하 가능성 존재**

---

3. Bounding Box 구조



- COCO format 기반
- 일부 bbox 좌표 오류 존재 (총 763개 중 5개 확인)
- 좌표는 `[x, y, width, height]` 형식

---

4. Category ID 불연속성

- min: 1900
- max: 41768

→ ID 값이 연속적이지 않으며, 모델 학습 전 **index 재매핑 필요성 존재**

---

5. Camera_Ia 기반 데이터 다양성

- 촬영 조건: 70 / 75 / 90

camera_Ia 값 변화는 조명 조건 차이를 의미하며 이는 **자연스러운 데이터 확장 효과**를 제공함.

→ 별도의 강한 색상 증강 없이도 **조명 변화에 대한 모델 강건성 확보에 기여**

---

6. 클래스 구성 전략

- Test 데이터(40종)가 Train 데이터(56종)와 거의 일치함 확인
- 별도 클래스 축소 없이 **전체 56종 학습 진행**

→ 실제 평가 환경과의 일관성 유지

---

7. 시각적 분포 분석

Train 데이터 수는 적은 편이나,

- 알약은 제조 공정상 형태·색상이 정형화됨
- 배경 및 촬영 환경이 비교적 통제됨
- 클래스 내 변동성(Low Intra-class Variation)이 낮음

따라서 모델이 학습해야 할 변동 요인이 제한적이며, 소량 데이터로도 비교적 높은 정확도 달성 가능하다고 판단.

이 문제는 일반 객체 탐지와 달리 알약의 **각인, 미세 곡률, 색상 조합 등 도메인 특화 시각 패턴 학습**이 핵심임.

---

2) 데이터 전처리

1. Annotation 보완

Train 데이터가 제한적이므로, 데이터 삭제 대신 **오류 bbox 수정 방식**을 선택함.

총 6개 bbox 좌표 수정:

- K-003351-016262-018357.json / 75
  → K-018357 / bbox: 581, 633, 285, 295
- K-003351-019232-029667.json / 70
  → K-019232 / bbox: 534, 218, 210, 309
- K-003351-020014-020238.json / 90
  → K-003351 / bbox: 397, 256, 166, 165
- K-003351-020238-031863.json / 70
  → K-020238 / bbox: 598, 306, 196, 192
- K-003351-029667-031863.json / 70
  → K-003351 / bbox: 382, 865, 167, 173
- K-001900-016548-019607-033009.json / 70
  → K-016548 / bbox: 94, 862, 242, 242

→ 소량 데이터 환경에서 데이터 보존을 최우선 전략으로 설정

---

2. Train / Validation Split

- Image 단위 split 적용
- Seed 고정 → 실험 재현성 확보
- YOLO 및 앙상블 실험에만 적용
  ※ Faster R-CNN 모델은 train/val split을 별도로 적용하지 않고 전체 Train 데이터를 활용하여 학습 진행함.

---

3. 중복 박스 제거

모델이 동일 객체에 대해 다수의 bbox를 예측하는 현상 발생.

이를 해결하기 위해:

- **NMS (Non-Maximum Suppression)** 적용
- Faster R-CNN 모델에 우선 적용

→ False Positive 감소

→ mAP@[0.75:0.95] 구간에서 성능 안정화

▼ 4. 모델 설명

▼ 1) Faster R-CNN

a. 선정 배경 및 설계 개요

- 수업 시간에 배운 모델임
- 따로 라이브러리를 설치할 필요 없음: torchvision 내 import로 로딩 용이함
- 대표적인 2-stage 모델임

b. 모델 성능 결과

- 하단 이슈로 성능 측정에 어려움을 겪음

c. 진행 중 발생 이슈

- Kaggle score 0점 발생함
- 원인1: 이미지 증강 과정에서 size 조정이 잘못됨.
- 원인2: evaluation 과정에서 테스트 데이터를 정규화시키지 않고 모델에 넣음
- validation dataset 분리하지 않고 진행해 Kaggle 점수로만 평가함

▼ 2) Yolo v8 (baseline)

a. 선정 배경 및 설계 개요

- 1-stage Detector 아키텍처적 우위 있음
- 미세 객체 탐지 성능 우수함
- 통합 최적화 구조임

b. 모델 성능 결과(base line)

- baseline 성능 결과
  - Kaggle Score : 0.4117
  - **낮은 평균 신뢰도 (Mean Score: 0.26)**
  - **유의미한 검출 부족**
  - **최저 점수 빈발**
  - **클래스 편향성 (Class Bias)**
  - **위치 부정확성 (Localization Error)**

c. 진행 중 발생 이슈

- **데이터 증강(Augmentation) 부재**
- **환경 변화 취약성**

▼ 5. 성능 개선 실험

▼ 1) Faster R-CNN

a. 이미지 증강

- RandomResizedCrop/Resize
  - 이미지 크기 랜덤 조정
  - 초기 Kaggle값이 제대로 나오지 않은 문제의 원인
- RandomHorizontal/VerticalFlip
  - 상하/좌우 반전
- RandomApply
  - p값 기준 무작위 적용하는 함수
- RandomAffine
  - 회전 각도 범위, 이미지 크기 대비 이동 비율, 확대/축소, shear 조절
- ColorJitter
  - 밝기/대비/채도/색조 변화
- RandomPhotometricDistort
  - ColorJitter보다 강하고 공격적인 색·조명 증강
- Normalize
  - 정규화 mean=(0.485, 0.456, 0.406), std=(0.229, 0.224, 0.225) **ImageNet** 데이터셋의 통계치로 설정

- p값을 0.2~0.5로 설정하고 각 세부 파라미터는 0.1 ~ 0.15 사이 너무 과하지 않게 조정함
- 결론: 기하, 색감 등 증강이 오히려 정확도를 떨어트려 정규화를 빼곤 전부 제외함

b. 모델 구조 변경

1. RPN(Region Proposal Network) 하이퍼파라미터 설정
   a. anchor generator 값 설정(더 작은 박스 검출)
   b. aspect ratio 설정(bbox 가로세로 비율)
   - 모두 모델 성능 저하 결과였음
     - 학습 과정에서 bbox 크기가 주요 문제가 아니었기 때문으로 판단함
2. Backbone 고정
   a. ResNet50+FPN 제외 학습 진행함
   - 성능이 심하게 저하됨
     - 구조까지 전부 학습했을 때의 성능이 가장 좋음

c. Fine-tuning 전략 (Hyperparameter Optimization)

▼ 모델 수렴 속도 향상 및 최적 성능 도출을 위해 **Optimizer**, **Learning Rate Scheduler** 주요 파라미터 조정하며 실험 진행함

1. Optimizer 설정:
   a. SGD (Stochastic Gradient Descent)
   • Learning Rate (lr): 0.01 설정 시 손실 값 하락 가장 안정적이었음
   • Momentum: 0.9 유지함 → 관성으로 로컬 미니마(Local Minima) 탈출에 효과적이며, 기본값이 검증 과정에서도 성능 가장 우수했음

• Weight Decay: 0.0005로 하향 조정함 → 페널티(규제) 낮추는 것이 학습 효율 및 정교한 특징 추출에 도움 됨 확인함

b. Adam

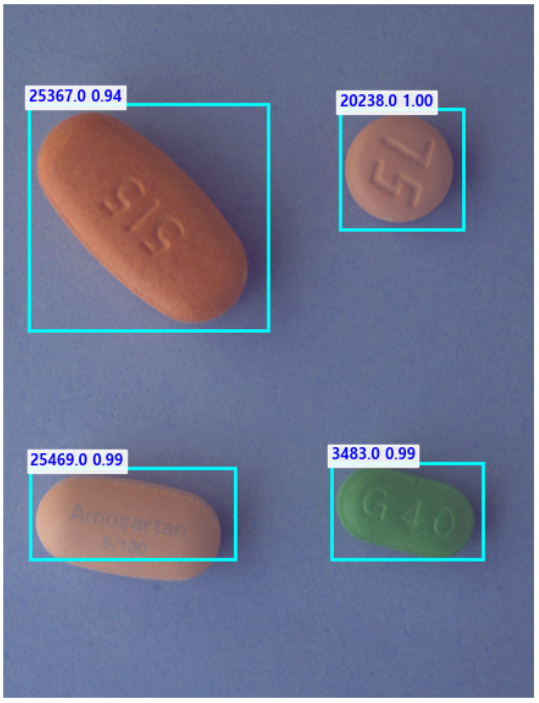
- lr = 1e-4 SGD보다 낮은 lr로 학습시작 필요함에 따라 낮춤
- betas=(0.9, 0.999) 추천값 적용
- weight_decay=5e-5 SGD와 동일

2. Learning Rate Scheduler 설정: StepLR
   • Step Size: 12 (Epoch)로 설정함 → 11회와 12회 간 유의미한 성능 차이 없었으나, 학습 전반부 안정성 고려하여 12회 채택함. learning loss가 보통 10~12 epoch 사이까지 원만하게 하락하는 걸 관찰함.
   • Gamma : 0.5로 설정함

   ◦ 실험 결과: 감마 값을 너무 작게 설정(예: 0.1 이하)할 경우, 학습률 급감으로 후반부 미세 조정 거의 불가 현상 발생함. 반대로 0.5보다 클 시에도 학습 효율이 안 나옴
   ◦ 결론: 감마 값을 0.5로 유지하여 학습률 변화 폭 확보 시, 모델이 전역 최적점(Global Optimum)으로 더 효과적으로 학습 진행하는 것으로 나타남

d. 앙상블 (Faster R-CNN + Yolo v8)

- 두 모델의 결과값 csv파일을 불러와 앙상블을 진행함
  - WBF(Weighted Boxes Fusion)
    - ensemble_boxes 라이브러리 활용함
    - 설정한 Threshord 값 이상일 시 좌표의 가중평균을 내는 방식임
    - Kaggle 점수 큰 폭으로 0.2 가까이 하락함
    - 시각화 결과 오히려 박스가 잘리는 문제가 발생함



- NMS(Non-Maximum Suppression)
  - torchvision.ops의 NMS 활용함
  - 겹치는 박스 중에서 가장 score 높은 것만 남기고 나머지는 버리는 방식임
  - 결과: Kaggle score 0.95983로 기존 결과값에서 소폭 상승함 기존 결과값: (FasterRCNN: 0.95117, YOLO: 0.94536)

e. Threshord 값 조정

- 0.25 최종채택
  - 0.03: Kaggle score 미세하게 하락함 -0.002
  - 0.5: Kaggle score 크게 하락함 - 0.05

f. 전처리 반영: 데이터 증강

- 최종 Kaggle 점수: 0.96292
- 이후 앙상블은 오히려 결과가 좋지 않아 제외함

▼ 2) Yolo v8 성능 개선 실험 1차(argument 적용)

- Kaggle Score : 0.94126
- 사용 모델 : YOLOv8s
- 성능개선 1차
  - Argumentation 적용: Mosaic : 1.0 / Translate : 0.1 / Scale : 0.5 / FlipLR: 0.5
- 성능 결과
  - **신뢰도 향상 (baseline 모델 대비 176% 향상)**
  - **미검출 클래스 존재**
  - **중복 검출 발생**

    - **특정 클래스 혼동 및 오분류**
- 원인 분석
    - **Train Image 데이터 부족**
    - **클래스 불균형**
- 개선 방향 및 실험 전략
    - **희귀 클래스 추출 및 복제**
    - **증강 기법 적용**

▼ 2.1) Yolo v8 성능 개선 실험 2차(모델 변경, 데이터 불균형 해소)

- Kaggel Scroe : 0.96195
- 사용 모델 : YOLOv8m (모델 변경)
- 성능개선 2차
    - 모델 변경 : YOLOv8s → YOLOv8m
    - Train Image 증강
        - 희귀 클래스 복제
        - 복제 class 증강 적용
- 성능 결과
    - 신뢰도 개선(Basline 대비 199% 상승함)
    - 고득점 검출 비율 증가
    - 학습 검증 지표 우수
    - 클래스별 편차 발생
- 원인 분석
    - 모델 파라미터 증대
    - 증강을 통한 일반화 성공
    - 최적 임계값의 유연화
- 개선 방향 및 실험 전략
    - 하이퍼파라미터 고도화 및 학습 환경 최적화
    - 후처리 최적화

▼ 2.2) Yolo v8 성능 개선 실험 2-1차(데이터 불균형 해소, 하이퍼 파라미터 설정)

- Kaggle Score : 0.97548
- 사용 모델 : YOLOv8m
- 성능개선 2-1차
    - 모델 학습시 Imgsz 변경 : 640 -> 1024
    - 모델 학습시 batch_size 변경 : 16 -> 8
    - 모델 학습시 Epochs 변경 : 30 -> 50
- 성능 결과
    - mAP50-95 상승

        - 신뢰도 양극화
            - 학습 데이터 포함 클래스는 0.98~0.99의 고득점을 유지하나, 특정 클래스는 0.03~0.4 수준의 저득점 기록
        - 중복 박스(Multiple Bounding Boxes) 발생
    - 원인 분석
        - 고해상도 학습의 효과
        - 학습 심화 및 최적화 (batch: 8, epoch: 50)
        - Threshold의 역설
        - Test Image data안에 Train/val 학습시 포함 되지않은 Class 발견
    - 개선 방향 및 실험 전략
        - 임계값 조정

▼ 2.3) Yolo v8 성능 개선 실험 2-2차(중첩 바운딩 박스 해소)

- Kaggle Score : 0.96719
- 사용 모델 : YOLOv8m
- 성능개선 2-2차
    - Confidence : 0.03 → 0.25
- 성능 결과
    - 평균 신뢰도 및 점수 변동
    - 학습 성능 유지
    - 중복 박스 및 노이즈 제거
    - 미학습 객체 필터링

▼ 3) Yolo 성능 개선 실험 비교 시각화

1.Yolo v8 성능 개선 실험 1차(arugument 적용)

Yolo v8 성능 개선 실험 2-2차(중첩 바운딩 박스 해소)

Yolo v8 성능 개선 실험 2-1차(데이터 불균형 해소 및 하이퍼 파라미터 설정)

Yolo v8 성능 개선 실험 2-2차(중첩 바운딩 박스 해소)

▼ 6. 최종 평가

▼ a. 성과 및 인사이트

- Faster R-CNN 모델
  - raw 데이터 품질과 EDA·전처리의 중요성을 실험적으로 확인함 (Garbage in, garbage out)
  - 모델 성능 개선 방법이 배운 것 이상으로 다양하였음(앙상블, 데이터 증강, 후처리 등)
  - train/test 데이터의 배경과 촬영 구도가 유사하여 데이터 분포가 단조로웠고, 이에 따라 일반적인 이미지 증강의 성능 개선 효과는 제한적이었음
    - 배경 변화에 따른 알약 색상 인지 차이가 발생할 가능성은 낮았음
  - 하이퍼파라미터 조절에 대한 이론적 근거가 제한적이여, 성능 개선은 실험적 탐색에 의존하는 경향이 큼
- Yolo 모델
  - 성능 향상은 모델 구조 변경보다 데이터 확장을 통한 클래스 분포 보정 시점에서 가장 크게 발생함.
    - 희귀 클래스 이미지를 회전·반전 방식으로 확장한 이후, 모델은 색상·형태 중심 단순 특징 대신 각인(Imprint)·질감 기반 특징을 학습하며 Bounding Box 정밀도와 클래스 구분 성능 동시에 개선됨.
  - 알약 객체가 작아 입력 이미지 해상도 확대(예: 1024) 시 각인, 경계선 등 미세 특징 추출이 더 효과적으로 이루어짐
  - 데이터 규모가 작아 작은 batch size가 유리할 것으로 예상했으나, 실제로는 batch size를 크게 설정했을 때 더 안정적인 성능을 보임
- 종합 평가
  - 데이터셋 규모가 작아 모델 간 성능 차이가 뚜렷하게 나타나지는 않았으며, Faster R-CNN과 YOLO 간 정량적 성능 차이는 크게 체감되지 않음
    - 다만, YOLO는 모델이 경량이고 학습·추론 파이프라인이 간결하여 실험 반복 및 활용 측면에서 장점이 있음

    - 데이터 중심 접근(Data-Centric AI)의 중요성
        - 모델 구조 변경보다 데이터 정제 및 클래스 분포 보정이 성능에 더 직접적인 영향을 미침 (특히 소규모 데이터셋에서는 모델 복잡도보다 데이터 품질이 우선적 요인으로 작용함)

▼ b. 향후 개선방안

- 데이터 전처리:
    - 알약은 앞·뒷면이 서로 다른 시각적 특징을 가지므로, 향후 양면 이미지를 모두 반영한 학습 데이터 구성을 통해 분류 및 검출 성능 향상을 도모
- Faster RCNN:
    - validation dataset 구성
    - seed 값 고정
    - 다양한 앙상블 기법에 대한 추가 실험 수행 (ex. 클래스 예측 단계에서의 앙상블 적용, 모델별 weight 조절 기반 앙상블 적용을 통해 예측 안정성 및 성능 향상 검토)
    - Optuna 기반 다양한 실험 수행
- Yolo:
    - 더 높은 버전의 Yolo 모델로 backbone 변경
    - Yolo 모델끼리의 앙상블 실험

(부록) YOLOv8 모델 최종 보고서 원본 docs url

https://docs.google.com/document/d/14MNHtKdSQDovdixws9mDFXpTn97chrMTzO_r2BRy6IY/edit?tab=t.0