AI 기반 경구약제 객체 탐지

스마트폰 촬영 이미지에서 알약 Class + Bounding Box 동시 탐지

데이터 : Kaggle (AI Hub 원천) · 클래스 : 56 종
학습 데이터 : 원본 369 장 → 증강 포함 2358 장
최종 점수 : YOLOv8+WBF 0.82211

프로젝트 개요

문제 정의

스마트폰으로 촬영된 알약 이미지에서
알약의 종류 (Class) 와 위치 (Bounding Box) 를
동시에 탐지하는 객체탐지 모델 개발

실무 목표 (서비스 흐름)

- 사진 1 장 촬영 → 약명 자동 인식
- 전문의약품 여부
- 효능 / 효과 · 연령 · 복용법
- 병용금기 성분 / 주의사항 안내

엔드투엔드 파이프라인

촬영 이미지

↓

객체탐지 모델

↓

클래스 ID

↓

서비스 정보 매핑

↓

결과 카드

프로젝트 목표 (3 가지)

Objectives

1) 데이터 구조 파악 / 정리

Kaggle(AI Hub 원천) 구조를 분석하고
학습 가능한 단일 COCO JSON 으로 병합

2) End-to-End 파이프라인

전처리 → 데이터셋 → 학습 → 추론 →
제출 파일 생성까지 한 번에 실행

3) Top30 서비스 매핑

class id 를 약
이름 / 제조사 / 성분 등
서비스 정보와 연결

데이터셋 준비 : JSON 병합

문제 : 라벨 (JSON) 이 수백 개로 분산

- 첫 번째 병목 : 라벨 (JSON) 이 폴더 / 파일 단위로 분산 → 학습 불가

해결 : glob 재귀 탐색으로 전체 json 수집 → images/annotations/categories 병합
중복 방지 : seen_image_ids / seen_category_ids 적용
결과 : 단일 COCO(train.json) 생성 → 이후 모든 모델에서 재사용

증강 최종 데이터 (요약)

이미지 2358 장 · 박스 8455 개 · 클래스 56 종

원본 369 장 → 증강 2358 장

핵심 코드 (병합)

```
# 분산 JSON(폴더/파일 단위) → 단일 COCO JSON으로 병합
json_files = glob.glob(os.path.join(ANNOTATION_ROOT, '**', '*.json'), recursive=True)

merged = {'images': [], 'annotations': [], 'categories': []}
seen_img, seen_cat = set(), set()

for p in tqdm(json_files):
    d = json.load(open(p, 'r', encoding='utf-8'))

    for img in d.get('images', []):
        if img['id'] not in seen_img:
            merged['images'].append(img); seen_img.add(img['id'])

    merged['annotations'].extend(d.get('annotations', []))

    for cat in d.get('categories', []):
        if cat['id'] not in seen_cat:
            merged['categories'].append(cat); seen_cat.add(cat['id'])

json.dump(merged, open('./data/train.json','w',encoding='utf-8'), ensure_ascii=False)
```

분산 JSON → 단일 COCO(train.json)

데이터 이해 & 서비스 매핑

pill_info_map 구축

categories 에 포함된 제조사 / 성분 정보를 활용해
class_id → (약 이름 / 회사 / 성분) 으로 매핑

→ 탐지 결과를 “서비스 카드”로 연결하는 기반

Top30 상비약 후보 추출

- annotations 기반 클래스별 샘플 수 집계
- count 내림차순 정렬 → 상위 30 개 저장
- 결과물 : top30_pills_service.csv

클래스 분포 (상위 10) · 증강 후

그래프 1. 상위 10 개 클래스의 annotation(박스) 개수 (증강 후 데이터 기준)

데이터 불균형 해결

Offline augmentation

문제 : 클래스당 샘플 수 부족

원본 기준 : 최소 3 장 , 대부분이 50 장 미만
→ 학습 붕괴 위험이 큼

50 장 미만 클래스 : 55 개 → 0 개

증강 설계 포인트

- bbox 동반 변환 필요 → Albumentations bbox_params
- 경고 / 호환 이슈 (SafeRotate value) 처리
- ID 충돌 방지 : 200000 번대부터 새 ID 부여
- 결과 : 클래스 최소 개수 50 이상 달성

클래스 샘플 수 요약 (Before vs After)

그래프 2. 증강으로 최소 샘플 50 장 확보 및 50 장 미만 클래스 0 개 달성
원본 : 369 장 기준 / 증강 후 : 2,358 장

사용자 편의성 : “ 약 봉지 촬영” 대응

왜 필요했나 ?

실사용에서는 알약을 비닐 / 약 봉지에 담아 촬영하는 경우가 많아
회전 · 반전 · 조명 변화 · 흐림 · 난반사에 강건해야 함

적용한 증강 (설명)

- SafeRotate(20): 봉지 안에서 굴러다니는 상황
- Rotate(180): 모든 각도 회전
- H/V Flip: 촬영 방향 / 앞뒤 모호함
- Brightness/Contrast: 조명 변화
- GaussianBlur: 비닐로 인한 뿌연 효과
- GaussNoise: 노이즈 추가 (강건성)
- CLAHE: 비닐 난반사 / 국소 대비 변화

코드 스냅샷

```
# 사용자 편의성: ‘약 봉지 촬영’에서도 탐지 가능하도록 증강/전처리
train_transform = A.Compose([
    A.Resize(Config.IMG_SIZE, Config.IMG_SIZE),

    # (1) 봉지 속: 무작위 방향 / 굴러다님
    A.SafeRotate(limit=20, p=0.5),
    A.Rotate(limit=180, p=0.7),

    # (2) 촬영 방향: 상/하/좌/우 뒤집힘
    A.HorizontalFlip(p=0.5),
    A.VerticalFlip(p=0.5),

    # (3) 비닐 특성: 흐림/난반사/노이즈
    A.GaussianBlur(blur_limit=(3, 5), p=0.3),
    A.GaussNoise(var_limit=(10, 50), p=0.3),
    A.CLAHE(clip_limit=4.0, p=0.3),

    # (4) 조명 변화: 밝기/대비 변화
    A.RandomBrightnessContrast(brightness_limit=0.2, contrast_limit=0.2, p=0.5),

    A.Normalize(mean=(0.485, 0.456, 0.406), std=(0.229, 0.224, 0.225)),
    ToTensorV2()
], bbox_params=A.BboxParams(format='coco', label_fields=['category_ids']))
```

Albumentations + bbox_params(format="coco")

데이터셋 / 로더 & 파이프라인 안정화

PillDataset 핵심 처리

- COCO bbox (x,y,w,h) → (x1,y1,x2,y2) 변환
- 화면 밖 bbox clipping
- 라벨을 학습용으로 1 부터 재매핑
- 이미지 로드 실패 / 빈 박스 샘플 자동 스킵

목표 : “ 돌아가는 파이프라인”을 팀 공통 자산으로

실험이 바뀌어도 전처리 / 데이터 선택 / 제출 생성이 흔들리지 않게 구성

Smart Select (코드)

```
# (Smart Select) 증강 > 정제 > 원본 순으로 자동 선택
AUGMENTED_PATH = './data/train_final_augmented.json'
CLEAN_PATH     = './data/train_final.json'
RAW_PATH       = './data/train.json'

if os.path.exists(AUGMENTED_PATH):
    TRAIN_JSON_PATH = AUGMENTED_PATH
elif os.path.exists(CLEAN_PATH):
    TRAIN_JSON_PATH = CLEAN_PATH
elif os.path.exists(RAW_PATH):
    TRAIN_JSON_PATH = RAW_PATH
else:
    raise FileNotFoundError('학습용 JSON 파일이 없습니다.')
```

팀원 환경이 달라도 “가장 좋은 데이터” 자동 선택

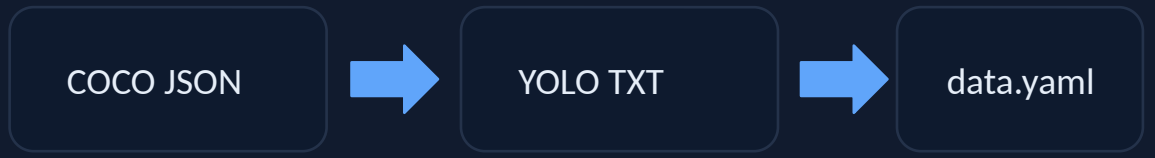
피벗 : YOLOv8 도입

왜 YOLOv8 인가 ?

- 학습 / 추론 속도 빠름 → 튜닝 사이클 짧음
- 실시간 탐지 서비스에 현실적
- 기간 내 성능 확보에 유리

핵심 작업 : COCO → YOLO 포맷 변환

- categories 정렬 후 id_map 생성 (원본 ID → 0~55)
- bbox (x,y,w,h) → (cx,cy,w,h) 정규화
- data.yaml 에 names(0~55) 등록



변환 코드 (요약)

```
# COCO(JSON) → YOLO(TXT) 변환: bbox (x,y,w,h) → (cx,cy,w,h) 정규화
cat_idx = id_map[ann['category_id']]
x, y, w_box, h_box = ann['bbox']  # COCO

x_center = (x + w_box/2) / img_w
y_center = (y + h_box/2) / img_h
w_norm   = w_box / img_w
h_norm   = h_box / img_h

f_out.write(f"{cat_idx} {x_center:.6f} {y_center:.6f} {w_norm:.6f} {h_norm:.6f}\n")
```

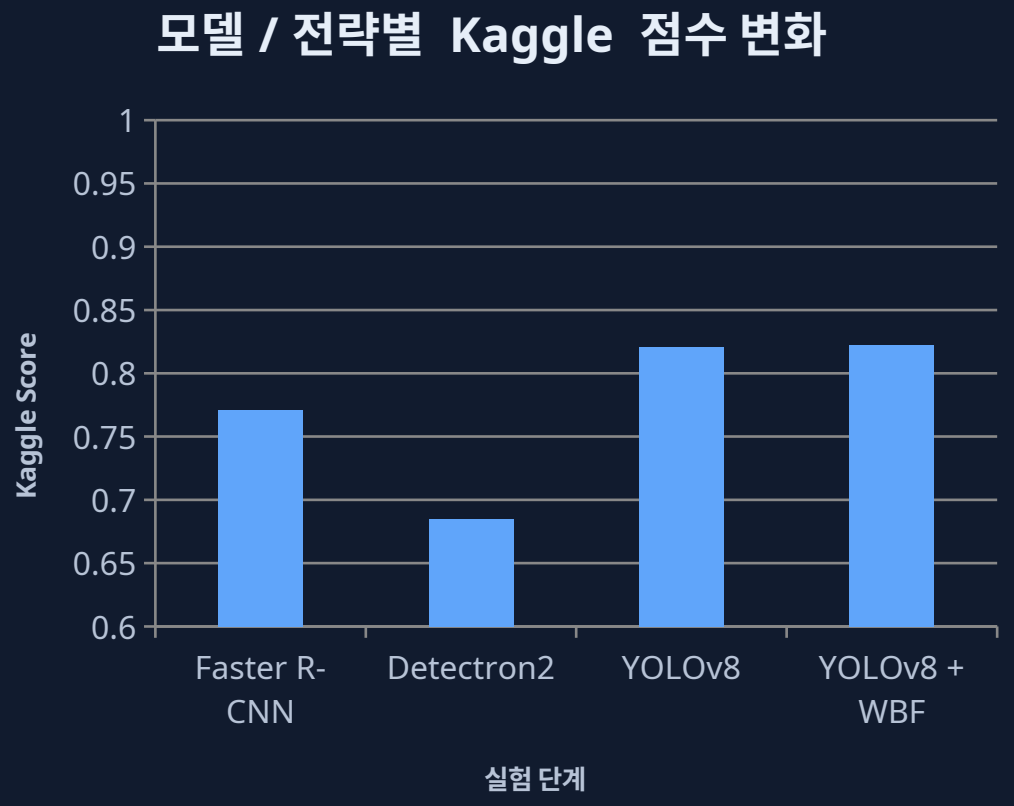
COCO bbox → YOLO bbox 정규화

모델 실험 타임라인

실험 흐름

- • Faster R-CNN: 파이프라인 검증용 베이스라인 (0.7712)
- Detectron2: 과다 예측 (박스 난사) → threshold/NMS 튜닝 부담 ↑ (0.68491)
- YOLOv8: 속도 / 튜닝 효율로 피벗 → 점수 · 실용성 동시 확보 (0.82073)
- TTA: 이득 미미 / 하락 (0.82061) → 단독 채택 X
- WBF: 예측 결합 + label +1 보정 → 최종 제출 (0.82211)

Kaggle 점수 비교



그래프 3. 베이스라인→피벗 (YOLO)→ 앙상블 (WBF) 로 점수 개선 (0.7712 → 0.82211)

0.7712 → 0.82211 (피벗 + 앙상블)

최종 제출 : WBF 앙상블 + 라벨 보정

WBF(Weighted Boxes Fusion) 전략

같은 모델에서 augment=False / augment=True 두 예측을 생성하고
박스 정규화 후 WBF 로 결합

실수 포인트 (중요): 라벨 +1 보정

YOLO 라벨 : 0~55 → Kaggle 제출 : 1~56

최종 결과

- YOLOv8 기본 : 0.82073
- TTA 단독 : 0.82061 (소폭 하락)
- WBF 앙상블 : 0.82211 (최종)

핵심 코드 (WBF +1)

```
# WBF 앙상블 후: YOLO 라벨(0~55) → Kaggle 제출(1~56) 보정
boxes, scores, labels = weighted_boxes_fusion(
    boxes_list, scores_list, labels_list,
    weights=[2, 1], iou_thr=0.55, skip_box_thr=0.01
)

for box, score, label in zip(boxes, scores, labels):
    # 중요: +1 보정
    category_id = int(label) + 1
    results.append({'category_id': category_id, 'score': float(score), 'bbox': box})
```

category_id = int(label) + 1

산출물 & 재현 방법

핵심 산출물

파일

- train_final_augmented.json (증강 포함 라벨)
- yolov8_best.pt (최종 모델 가중치)
- submission_FINAL_1BASED.csv / submission_ensemble_WBF.csv
- yolo_data_config.yaml (names 0~55)

재현 흐름

- 1) 데이터 병합 → train.json 생성
- 2) 불균형 / 봉지 촬영 대응 증강 + 전처리
- 3) COCO→YOLO 변환 후 학습
- 4) 추론 (TTA) + WBF 앙상블
- 5) label +1 보정 후 제출 파일 생성