AI-Hub 알약 이미지 데이터를 활용한 객체 탐지 시스템 개발

알약 이미지 객체 탐지 파이프라인 구축 프로젝트

6팀

한경섭
박상진
오현서
오현석

목차

프로젝트 개요 및 목표

프로젝트 목표

- AI-Hub 알약 이미지 데이터를 활용한 객체 탐지 시스템 개발
- 한 이미지 내 최대 4개의 알약에 대한 위치(Bounding Box) 예측
- 알약 종류(Class) 동시 예측 파이프라인 구축

데이터 특성

- AI-Hub에서 제공하는 알약 이미지 데이터셋 활용
- 초기 651장 이미지, 1,001개 JSON 어노테이션 분석
- 다양한 알약 클래스와 형태 포함

접근 방식

- 객체 탐지(Object Detection) 모델 기반 접근
- 데이터 구조 및 어노테이션 품질 우선 개선
- 단계적 성능 향상 전략 수립

팀 구성 및 역할

팀 구성

- 총 4인으로 구성된 프로젝트 팀
- 각 팀원의 전문성을 고려한 역할 배분

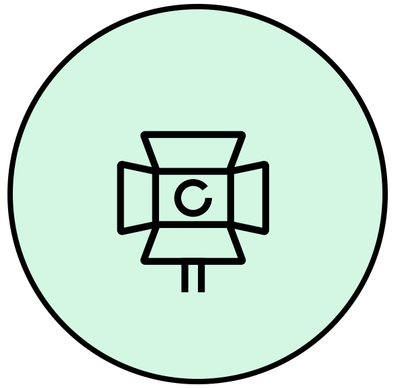
역할 분담

- PM: 프로젝트 관리 및 일정 조율
- 데이터 엔지니어: 데이터 분석 및 전처리
- 파이프라인/베이스라인 엔지니어: 모델 구축
- 최적화 엔지니어: 성능 개선 및 튜닝

작업 방식

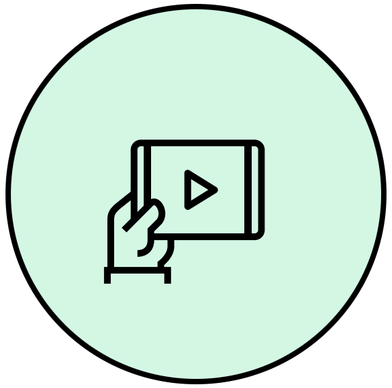
- 애자일 방식 채택으로 빠른 피드백과 개선
- 데이터와 구조 문제를 우선적으로 해결하는 접근법
- 단계별 성능 향상에 집중

초기 베이스라인 구축



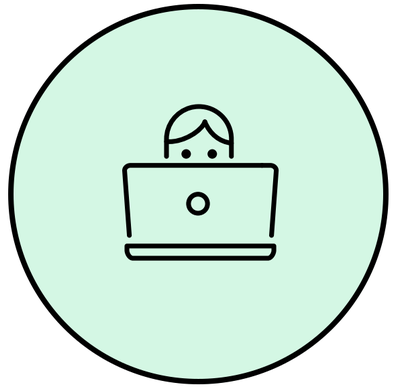
데이터 분석

- 651장의 이미지 확보
- 1,001개의 JSON 어노테이션 분석
- 데이터 구조 및 형식 파악
- 클래스 분포 확인



문제점 발견

- 어노테이션과 이미지 매칭 문제
- 232장만 정상 학습 가능
- 동일 이미지에 중복 어노테이션
- 데이터 구조적 불일치



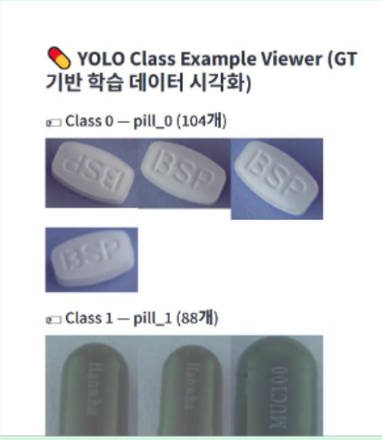
초기 모델 학습

- YOLOv8 기반 모델 선택
- 제한된 데이터로 초기 학습
- 초기 Kaggle 점수 0.49 획득
- 개선 방향 설정

수동 라벨링 작업

Step 1: 기준 데이터 준비

라벨이 있는 데이터 클래스별 시각화



Step 2: 클래스 매핑

샘플과 비교하여 클래스 결정

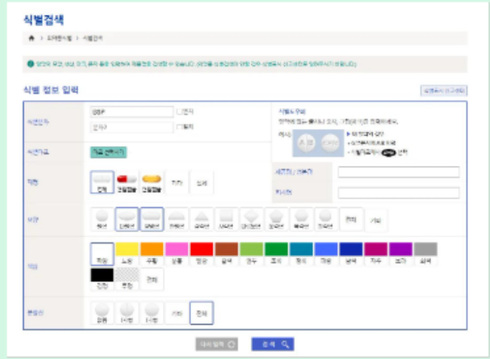
Step 3: Bounding Box 생성

알약 영역을 직접 확인해 박스 생성

YOLO txt 방식으로 저장

Step 4: 신규 클래스 처리

약학 정보원과 AI-Hub Json 사용



모델 예측 후 클래스가 일치하면 저장

예측이 틀릴 경우 수동 라벨링

Step 5: 라벨링 후 재학습

submission13.csv

Complete · 박상진qkrtkdwls · 11d ago

0.82158

AI-Hub 추가 데이터 활용

클래스 해당 알약만 선별적 활용

- 전체 AI-Hub 데이터 중 필요한 알약 클래스만 선별
- 클래스 불일치 데이터는 학습에서 제외

노이즈 최소화 위한 image-level 선별

- 이미지 단위로 데이터 품질 평가 및 선별 적용
- 노이즈가 많은 이미지는 학습에서 제외하여 모델 안정성 확보

학습 데이터 확보 및 성능

- 총 1,187장의 고품질 학습 데이터 확보
- Kaggle 점수 0.83으로 제한적 성능 향상 확인

Crop 전략 적용

▲원본 이미지

데이터 확장

- 알약 개별 crop 적용
- 약 24,000장으로 데이터셋 확장
- 다양한 알약 패턴 학습 가능

▲crop한 이미지

성능 향상

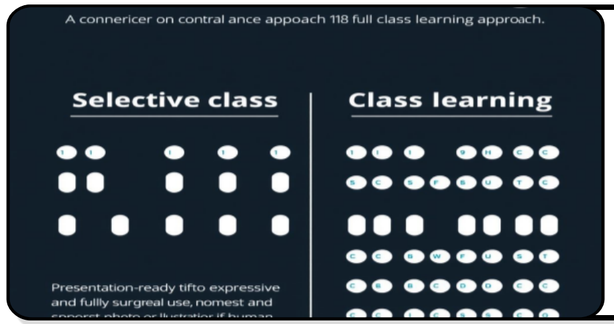
- Kaggle 점수 0.857까지 향상
- 데이터량 증가로 인한 성능 개선
- 일부 클래스 인식 정확도 향상

▲객체 위치 인식 성능 저하

한계점 발견

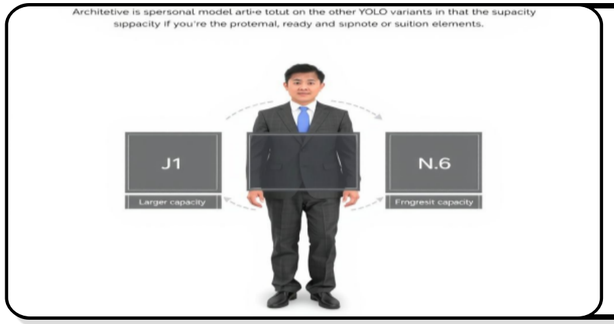
- 데이터량 증가만으로는 한계 확인
- 모델이 중앙 객체 패턴에 과적합
- 위치 일반화 성능 감소 문제 발생

전체 클래스 학습 전략 전환



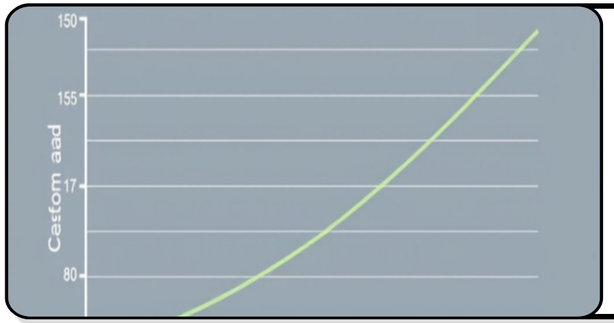
선별 학습에서 전체 학습으로

- 기존: 필요 클래스만 선별적 학습
- 변경: 전체 118개 클래스 모두 학습
- 데이터 구조 전면 개편으로 학습 안정성 확보



YOLOv8n -> YOLOv8L 대규모 모델 적용

- 더 큰 모델 아키텍처 채택
- 복잡한 알약 패턴 인식 능력 향상
- 대용량 데이터 처리에 최적화된 모델 구성



성능 극대화 달성

- 선별 학습이 항상 최적이 아님을 확인
- 전체 클래스 학습이 일반화 성능 향상에 기여

모델 선정

Faster R-CNN

✓ 높은 정밀도
✗ 느린 학습 속도

RT-DETR

✓ 안정적 위치 예측
✗ 초기 데이터에 약함

YOLOv8m

✓ 빠른 학습
✗ 낮은 정밀도

🏆 YOLOv8L

최종 선정 모델
안정적 성능

튜닝

기본 학습

Imgsz=896, SGD, Early Stopping -> 안정성 확보

Freeze

Backbone 일부 동결, lr=0.002, Imgsz=1024 -> 정밀도 향상

Unfreeze

전체 fine-tune, lr=0.001, Imgsz=1024 -> 세부 정보 학습

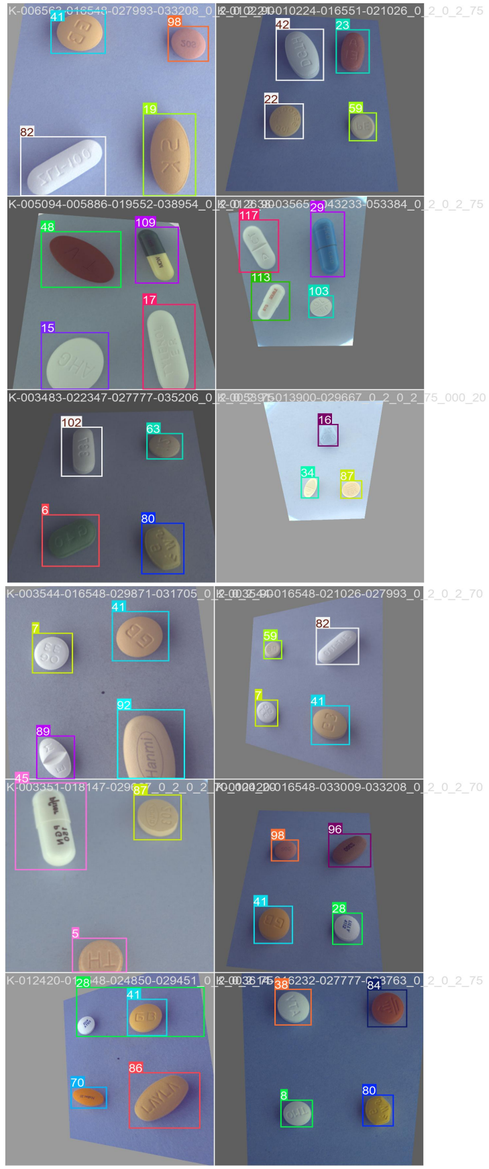
High-res

Imgsz=1152, lr=0.0007 -> 최적화

Noise

Imgsz=1152, lr=0.0005, 위치/스케일/erasing 증강

Noise 증강 적용 결과 및 한계



Noise 단계에서 성능 향상은 제한적이었으며, 일부 설정에서는 성능 저하가 발생

원인 ①

- 알약 데이터의 구조적 특성
- 색상·형태가 매우 유사
- 클래스 구분 단서(문자·각인)가 작고 흐릿함
- 노이즈 적용 시 각인·edge 정보가 먼저 손실됨

→ 클래스 간 시각적 구분 신호 약화

원인 ②

- 과증강(over-augmentation)
- YOLO 모델은 이미 강한 내부 증강을 포함
- 사용자 증강 추가 시 내부 증강과 중첩 발생
- 위치 이동·스케일 변화로 인해
- Bounding Box–feature 정합 붕괴

→ Noise 증강은 일반화보다 클래스 구분 신호를 먼저 훼손

핵심 설계 원칙

01

dl_idx 기반 클래스 식별

- dl_idx를 유일 클래스 식별자로 사용
- 클래스 혼선 최소화
- 일관된 식별 체계 유지

02

통일된 처리 체계

- JSON, YOLO txt, 제출 CSV 모두 통일된 dl_idx 기준 처리
- 데이터 변환 오류 방지

03

증강 데이터 활용 원칙

- 증강 데이터는 train 용도로만 활용
- 검증 데이터 오염 방지
- 성능 평가 신뢰성 확보

04

COCO 원칙 유지

- 1 image = 1 image_id 원칙 유지
- 구조 안정성 확보
- 재현 가능한 파이프라인

결론 및 시사점

이번 알약 이미지 객체 탐지 프로젝트를 통해 다음과 같은 중요한 시사점을 얻을 수 있었습니다.

- 객체 탐지 성능 문제는 모델보다 데이터 구조 및 어노테이션 정합성 개선이 더 중요함

- 베이스라인 → 구조 분석 → 데이터 복원 → 전략 전환 과정에서 성능 병목이 모델 이전 단계에 존재함을 확인

- 데이터 품질과 구조적 안정성이 모델 성능의 핵심 요소

- 선별적 학습보다 전체 클래스 학습이 더 나은 일반화 성능을 제공할 수 있음

- 확장성과 안정성을 모두 갖춘 실무 적용 가능한 객체 탐지 시스템 구축 가능성 확인