도메인 프로젝트 1 CV Object Detection 랩업 리포트 (김해찬_T8058)

“재활용 품목 분류를 위한 Object Detection”에서 직접 제출한 모델과

팀에서 공유한 결과물을 활용해 최종 앙상블(ensemble_5, mAP 0.6138)까지 도달한 과정을 정리

프로젝트 개요 → 수행 절차와 실험 흐름 → 결과 및 자체 평가 → 개인 회고 순으로 정리

---

1. 프로젝트 개요

1-1. 문제 정의

- 과제: 10종의 쓰레기(일반, 플라스틱, 종이, 유리, 비닐 등)를 포함한 1024×1024 이미지에서 객체를 탐지하는 Object Detection 문제
- 입력: COCO format으로 주어진 train.json(10개 class, bbox annotation 포함), test 이미지는 annotation 없이 이미지 정보만 제공 도메인 프로젝트1 - 재활용 Object Detectio...
- 출력: 각 객체에 대해 `[class score x y w h]` 형태의 PredictionString으로 구성된 csv 제출
- 평가지표: mAP50 (IoU=0.5 기준)

이 문제는 “분리수거 자동화”라는 실제 도메인과 연결되어 있고, 작은 객체(배터리, 병뚜껑 등)와 큰 객체(의류, 박스 등)가 공존하며, 클래스 불균형과 cluttered scene(박스가 많이 겹치는 이미지)이 존재하는 것이 특징이다.

1-2. 팀 구성 및 역할(요약)

- 김해찬(본인):
  - MMDet 기반 추가 모델링(Cascade R-CNN, VFNet, Deformable DETR, TOOD, Sparse R-CNN, RTMDet 등)
  - csv 기반 박스 레벨 앙상블 파이프라인 설계 및 구현 (ensemble_from_submissions.py)
  - DETA 계열 partial ensemble 실패 분석 및 후속 전략 수립

2. 데이터 특성 및 베이스라인

2-1. 데이터 이해

- 이미지 해상도: 대부분 1024×1024 고정
- 한 이미지 내 bbox 개수 분포: 1개짜리 이미지가 많지만, 10개 이상 bbox가 있는 복잡한 이미지도 상당수 존재 → 작은 객체 학습이 어려울 수 있음
- 클래스 불균형: 일반쓰레기/종이/플라스틱/비닐 클래스가 상대적으로 많이 등장하고, 일부 클래스는 매우 희소
- bbox 위치: 대부분 이미지 중앙 및 중상단에 많이 분포하나, 전역적으로 골고루 등장하여 위치 편향만으로는 학습이 어렵다

이 EDA 결과를 통해

1. 작은 객체에 강한 모델 구조와 multi-scale augmentation이 필요하고,
2. overfitting 대비를 위해 적절한 regularization·augmentation 조합이 중요하다고 판단

2-2. 베이스라인 및 담당한 모델링

(1) Cascade R-CNN + Swin-L

- 설정
  - Backbone: Swin-L (patch4, window7)
  - Input size: 1024
  - Multi-scale training 및 기본 mmdet augmentation 사용
- 성능
  - Public LB: 0.4642 (김해찬_cascade_rcnn)
- 해석
  - 기존 Faster R-CNN(ResNet 계열) 대비 mAP가 상승
  - Swin-L의 global self-attention 덕분에 cluttered scene에서 객체 간 관계를 더 잘 캡처한 것으로 해석

    - 단, anchor 기반 모델 특성상 작은 객체에서 recall이 완전히 만족스럽지 않았고, hyperparameter 탐색 시간이 부족했다.

(2) VFNet

- 설정
    - Backbone: ResNeXt-101-64×4d
    - Focal Loss 변형(Varifocal loss)을 사용해 classification score와 IoU를 동시에 반영
- 성능
    - Public LB: 0.3751
- 해석
    - Positive sample에 IoU-aware score를 주는 구조라 detection “품질점수” 측면에서는 의미 있었으나,
    - heavy augmentation 없이 기본 설정으로만 실험하여 Cascade R-CNN 대비 성능이 다소 낮게 나옴.
    - class imbalance 대응을 위해 Varifocal loss를 택했지만, data level re-balancing(oversampling 등)을 병행하지 않아 potential을 충분히 끌어내지 못했다.

(3) Deformable DETR (ResNet-50)

- 설정
    - Backbone: ResNet-50
    - Deformable Attention을 통해 multi-scale feature를 sparse하게 샘플링
- 성능
    - Public LB: 0.252
- 해석
    - end-to-end set prediction이라는 구조적 장점이 있으나,
    - 학습 안정화에 시간이 많이 필요하고 hyperparameter 공간이 넓다.
    - 이번 대회에서는 충분한 epoch와 LR schedule 튜닝을 하지 못했고, 초기 설정(기본 config 기반)만으로는 anchor 기반 계열보다 낮은 성능을 보였다.

(4) TOOD / Sparse R-CNN / RTMDet

- TOOD
  - anchor-free + task-aligned head 구조
  - LB: 0.2686 (tood)
- Sparse R-CNN
  - learnable proposal 100개만을 반복 refinement하는 방식
  - LB: 0.2873 (sparse_rcnn)
- RTMDet
  - one-stage
  - LB: 0.0468 (rtmdet)

RTMDet는 Config/augmentation 설정 및 학습 정책이 데이터 특성과 맞지 않아 거의 실패 수준의 스코어를 기록했다.

Sparse R-CNN·TOOD는 이론적으로 anchor-free의 장점을 기대했으나, validation 세팅과 augmentation을 충분히 튜닝하지 못해 “quick try” 수준에서 마무리 된 것 같다.

---

3. 모델 실험 타임라인

아래는 Upstage 리더보드에 내가 제출한 순서(대략적인 시간순)와 점수, 그리고 실험 의도·결과 정리이다.

3-1. mmdet_faster_rnn (0.4177)

- 목적
  - 공식/팀 baseline으로 Faster R-CNN 계열 모델을 사용해 파이프라인이 정상 동작하는지 확인.
- 특징
  - COCO 사전학습 backbone + FPN + 2-stage detector.
  - 기본 augmentation, 기본 NMS threshold 사용.
- 결과 / 인사이트
  - mAP 약 0.42로, “아무것도 안 한 상태”보다는 충분히 의미 있는 출발점.
  - 이 점수를 기준으로 이후 모델·하이퍼파라미터 변경의 효과를 모두 상대적으로 비교.

3-2. DETR (0.2520)

- 목적
  - Transformer 기반 DETR 구조가 이 데이터에서 어느 정도까지 올라갈 수 있는지 감 파악.
- 시도
  - mmdetection의 DETR config를 기반으로 학습 및 추론.
- 결과
  - mAP 0.252로 Faster R-CNN보다 상당히 낮게 나옴.
- 원인 추정
  - DETR류는 일반적으로 긴 학습 스케줄과 세심한 hyperparameter tuning이 필요하지만,

    프로젝트 time-budget 상 충분한 epoch/스케줄을 돌리지 못함.
  - small object 비중이 있는 데이터에서 vanilla DETR은 baseline 대비 약점이 있는 편.
- 교훈
  - “Transformer detector를 쓰려면, 단순 config 변경만으로는 안 되고

    스케줄/augmentation/optimizer까지 세트로 설계해야 한다”는 점을 체감.

3-3. Cascade R-CNN (0.4642)

- 목적
  - 높은 IoU에서의 성능을 올려 mAP 전반을 끌어올리기.
- 특징
  - 3-stage cascade head가 반복적으로 bbox를 refine.
  - 높은 IoU threshold에서 학습하기 때문에 localization quality가 향상.
- 결과
  - mAP 0.4642로 Faster R-CNN baseline 대비 약 +0.05 수준의 개선.
- 인사이트
  - garbage detection 문제에서도 coarse한 박스보다 정확한 localization이 중요하므로

cascade 구조가 전반적으로 유리하다는 것을 확인.

3-4. VFNet (0.3751) 및 VFNet–Cascade 앙상블 (0.4401)

- 목적
  - Anchor-based two-stage와 달리, Anchor-free/IoU-aware loss(VFNet)를 도입해

    classification–localization을 동시에 개선할 수 있는지 확인.
- VFNet 단독 (vfnet, 0.3751)
  - Varifocal Loss, ATSS-style sample assignment 등을 사용.
  - 예상보다 낮은 점수: 0.37대.
  - 원인 추정
    - input scale, augmentation, training schedule에 더 민감했으나,

      baseline 수준의 튜닝만으로는 잠재력을 끌어내지 못함.
- VFNet + Cascade 앙상블 (vfnet_cascade_ensemble, 0.4401)
  - 두 모델의 csv를 단순 병합/포스트프로세싱해 앙상블.
  - VFNet 단독보다는 개선되지만, Cascade 단독(0.4642)보다는 낮게 나옴.
- 교훈
  - 서로 성격이 다른 모델을 섞는다고 항상 시너지가 나는 것은 아니고,
  - “둘 중 하나가 확실히 약하면, 앙상블이 강한 모델의 성능을 깎을 수도 있다”는 경험.

3-5. rcnn_sin (0.4760)

- 목적
  - RCNN 계열에서 augmentation·scheduler 등을 조정해

    mAP upper bound를 조금 더 끌어올려 보기.
- 특징 (추정)
  - Cascade/Faster R-CNN 구조에서
    - 학습 스케줄 조정,

        - 데이터 증강(Flip, Multi-scale, Color jitter 등)을 조합한 변형 실험.
- 결과
    - mAP 0.4760으로 당시 개인 single model 최고 점수.
- 인사이트
    - 구조를 완전히 갈아엎기보다 “검증된 구조 + 적절한 증강/튜닝”이
      제한된 시간에서 훨씬 안정적인 선택이라는 점을 다시 확인.

3-6. RTMDet (0.0468), TOOD (0.2686), Sparse R-CNN (0.2873)

이 구간은 “새로운 구조를 빠르게 찍어본 실험”에 가깝고, 대부분 실패 사례에 속한다.

1. RTMDet (rtmdet, 0.0468)

- 결과
    - mAP 0.0468로 사실상 학습이 제대로 안 된 수준.
- 원인 추정
    - COCO용 config를 거의 그대로 쓴 상태에서 class 수·prior·augmentation 등을
      데이터에 맞게 충분히 커스터마이즈하지 못함.
    - lr 스케줄/epoch 수가 부족해 학습이 수렴하지 못했을 가능성.

1. TOOD (tood, 0.2686)

- Anchor-free + task-aligned one-stage 구조.
- baseline보다는 많이 낮은 0.26대.
- one-stage 계열에서 적절한 scale jitter, warmup, longer schedule 등을
  맞춰주지 못한 것이 원인으로 보임.

1. Sparse R-CNN (sparse_rcnn, 0.2873)

- Query-based two-stage detector로,
  proposal을 fixed query로 학습하는 컨셉.
- 데이터량 대비 모델 자유도가 커서 underfitting.
- 기존 two-stage(Cascade)보다 튜닝 난이도가 컸고,
  충분히 손대지 못해 낮은 점수에 머무름.

3-7. DETA / DETA_nms (0.0031, 0.0025)

- 목표
  - DETA(Swin-L 계열) transformer detector를 활용해

    DDQ-DETR류와 유사한 성능대를 노려 보고,

    나중에 팀원 DDQ-DETR와도 앙상블하기 위한 발판 마련.
- 진행 과정
  1. fold 기반 학습을 진행하며 best_coco_bbox_mAP_epoch_*.pth를 생성.
  2. mmdet의 DetInferencer로 inference 스크립트를 작성했으나,
     - cfg_options 인자를 받지 않는 버전,
     - DetDataSample 출력 구조 변경 등 버전 이슈로 여러 번 에러 발생.
  3. num_classes mismatch
     - COCO 기본 config는 80 classes인데, 대회는 10 classes.
     - 처음에는 cfg_options로 bbox_head.num_classes, dn_query_generator.num_classes를

       patch하려 했지만, DetInferencer 버전 문제로 반영이 제대로 안 됨.
     - 임시 config(.py)를 새로 dump하여 num_classes를 강제하는

       infer_deta_partial_ensemble_v3_fix.py를 작성해 해결.

       - label mapping mismatch
         - inference 결과 `unique labels: [0, 1, 6, 7]`와 같이 일부 클래스에만 집중되는 현상 발생
         - train.json 상 class 순서와 DETA 사전 학습 weight의 class index가 완전히 일치하지 않는 상황에서 metainfo 주입과 config patch만으로는 mapping이 깨끗하게 맞지 않았다.
       - score calibration 문제
         - ensemble 전 개별 예측에서는 class 7에 대해 0.7~0.8대 confidence가 나오지만, fold 간 WBF/NMS를 거치면 score가 0.1 이하로 크게 떨어지는 현상 발생

        - 여러 모델이 같은 위치를 “약하게” 찍는 경우 평균/가중 평균으로 score를 뽑으면 과도하게 희석되는 문제가 있었다.

- 리더보드 결과
  - DETA 단일 제출: 0.0031 (DETA)
  - DETA + NMS 후 제출: 0.0025 (DETA_nms)

    → 실질적으로 실패 실험으로 분류.
- 핵심 원인 정리
  - training weight 자체는 정상인데,

    inference 스크립트에서
    - num_classes/label mapping,
    - score threshold,
    - NMS/WBF 파이프라인이 꼬이면서
    - 대부분의 예측이 제거되거나, 잘못된 클래스로 기록됨.
  - 특히 fold ensemble을 직접 박스 수준에서 하려다 보니,

    구현 복잡도가 올라가면서 bug risk가 크게 증가.

3-8. CSV 앙상블 시도 1: ensemble_2

이 구간에서는 mmdet 모델 여러 개를 csv 수준에서 합치는 실험을 수행했다.

사용 csv :

1. ddq_swinl_10_full.csv
   - DDQ-DETR, Swin-L, full train, test size 1024
   - Public LB: 0.6185
2. cascade_rcnn...4_12e.csv
   - Public LB: 0.5320
3. ddq_swinl_1024_12e.csv
   - DDQ-DETR, 12 epoch 버전
   - Public LB: 0.5986

4. ensemble_2

- DDQ-DETR(Swin-L 계열) submission
  - ddq_swinl_1024_12e (0.5986)
  - ddq_swinl_10_full (0.6185)
- cascade_rcnn...4_12e.csv를 같이 앙상블하는 스크립트

  ensemble_from_submissions.py를 작성해 실험:
  - PredictionString 파싱
    - "cls score x y w h ..." → (cls, score, x1, y1, x2, y2) 리스트로 변환.
  - per-image, per-class로 박스를 모은 뒤
    - WBF(Weighted Box Fusion) 또는 NMS 중 선택.
  - 가중치(weights) 옵션으로 모델별 신뢰도 반영.

- 초기 버전 특징
  - WBF에서 fused score를 “평균”으로 계산 →

    서로 비슷한 박스가 많이 겹쳐도 score가 0.1~0.2대로 내려가는 현상.
  - score_thr는 0.03 정도로 낮게 잡았지만,

    원래 0.7~0.8이던 box들이 0.1대로 내려가 precision이 망가짐.
- 결과
  - ensemble_2 single 모델보다 낮은 0.49~0.50대.
- 교훈
  - WBF 점수 스케일을 잘못 잡으면,

    “좋은 박스 + 좋은 박스”가 “미지근한 박스 한 개”가 되어버릴 수 있음.
  - score calibration과 threshold는 앙상블 성능에 영향을 준다.

3-9. CSV 앙상블 최종: ensemble_4, ensemble_5 (0.6138)

위 문제를 해결하기 위해 ensemble_from_submissions.py를 다음과 같이 수정하고,

최종적으로는 ensemble_5를 얻었다.

사용 csv :

1. ddq_swinl_10_full.csv
    - DDQ-DETR, Swin-L, full train, test size 1024
    - Public LB: 0.6185
2. codetr_swinl_1024_7e.csv
    - Public LB: 0.6910
3. ddq_swinl_1024_12e.csv
    - DDQ-DETR, 12 epoch 버전
    - Public LB: 0.5986

    이전까지 사용하던 Cascade R-CNN 계열 csv 대신,

    성능이 가장 좋은 CODetr Swin-L 모델을 새로 포함시켜 모델 간 구조적 다양성을 확보했다.
4. WBF 점수 계산 방식 변경

- 기존: cluster 내 score 평균

  fs = sum(scores) / len(scores)

- 수정: cluster 내 score 최대값

  fs = max(scores)

- 이유
    - DETR 계열 모델의 probability calibration이 이미 괜찮기 때문에,
      여러 모델이 같은 객체를 잡았으면 “가장 자신있는 score”를 살려주는 것이
      precision 측면에서 유리하다고 판단.

- Ensemble_4
    - WBF 사용
    - Public LB: 약 0.50대 초반 (ensemble_4: 0.5035)

- Ensemble_5 (최종 제출 선택)
  - NMS 방식으로 전환 (`-use-wbf` 제거)
  - weights:
    - ddq_swinl_10_full: 0.85
    - ddq_swinl_1024_18e_full: 0.8
    - codetr_swinl_1024_7e: 1.0

      → CODetr에 가장 높은 weight를 주되, DDQ 두 모델도 보조 evidence로 사용
  - score_thr = 0.03, IoU_thr = 0.5
  -

1. WBF 대신 NMS-only 조합도 실험

- `-use-wbf`를 끄면
  - 단순히 모든 박스를 모아 class-wise NMS를 수행.
  - box 위치는 "가장 score 높은 박스"가 대표 박스가 됨.
- 최종적으로는 WBF를 끄고 + 수정된 score(max) 조합이

  리더보드 상에서 가장 안정적인 결과를 보였고, 이를 기반으로 제출.(극단적인 오검출(outlier)을 NMS로 제거하는 안정화 효과 를 기대하며 최종 제출로 선택했다.)

1. 모델별 가중치 설정

- 최종 실행 (ensemble_5)
  - csvs
    - ddq_swinl_10_full.csv (0.6185)
    - ddq_swinl_1024_12e.csv (0.5986)
    - codetr_swinl_1024_7e.csv (0.6910)
  - weights
    - 0.85, 0.8, 1.0
    - "성능 좋은 모델에 조금 더 weight,

1. 결과

- ensemble_5: mAP 0.6138

---

4. 기술적 인사이트 정리

1. Detection 모델 선택

- 시간 제한이 있을 때
  - 처음부터 복잡한 transformer 계열(DETR, DETA 등)에 올인하기보다,
  - Faster/Cascade R-CNN과 같이 검증된 구조로 baseline을 안정화하는 것이 우선.
- 그 위에
  - 특정 데이터 특성(작은 객체, dense한 장면 등)을 보고

    VFNet, TOOD, Sparse R-CNN 같은 대안 구조를 “하나씩” 추가하는 전략이 더 현실적.

1. mmdet + mmengine 환경

- 동일 모델이라도 버전 차이로 인해
  - DetInferencer의 인자 지원 여부(cfg_options),
  - 출력 객체 타입(DetDataSample vs dict),
  - config 옵션 경로가 모두 바뀔 수 있다.
- 이번 프로젝트에서
  - “임시 config 파일을 dump해서 num_classes를 직접 수정하는 방식”이

    cfg_options보다 더 안정적인 해결책이었음을 확인.

1. CSV-level 앙상블

- PredictionString을 직접 파싱하면
  - framework에 상관 없이(예: mmdet vs YOLO vs 기타 툴)

    다양한 모델의 submission을 하나의 파이프라인에서 합칠 수 있다.
- 중요한 포인트

- 좌표계(x,y,w,h ↔ x1,y1,x2,y2) 변환 정확성
- class id 일관성(1-based/0-based)
- score calibration(WBF vs NMS, score_thr)
- 모델별 weighting

---

5. 실패 사례 및 원인 요약

1. RTMDet, TOOD, Sparse R-CNN이 기대 이하였던 이유

- config를 거의 그대로 사용해 “간단히 돌려본” 수준에 그쳤고,

  이 모델들이 요구하는 학습 스케줄, data augmentation, optimizer 설정을

  데이터에 맞게 조정하지 못함.
- lesson: 성능 좋은 논문 모델이라도,

  “시간 투입 < 튜닝 난이도”이면 baseline보다 못한 결과가 나올 수 있다.

1. DETA / DETA_nms 0.003대 문제

- num_classes=80 vs 실제 10 클래스 mismatch.
- inference 파이프라인에서 label/mapping, NMS, score threshold를 동시에 건드리다 보니

  bug를 잡기가 매우 어려웠음.
- 단일 이미지 디버깅에서는 잘 나와도, 전체 pipeline에서는 거의 모든 박스가

  잘못된 class거나 threshold에서 사라지는 문제가 존재.
- lesson
  - 모델 구조를 직접 ensemble하기 전에,
  - “일단 각 모델이 제대로 된 submission.csv를 잘 뽑는지”를

    가장 먼저 검증해야 한다.

1. 초기 CSV 앙상블(ensemble_2/4)이 single보다 나빴던 이유

- WBF score를 평균으로 계산해 score 스케일이 전체적으로 낮아짐.
- score_thr=0.03임에도 불구하고 상위 박스들이 과도하게 깎여 precision이 하락.
- lesson
  - 앙상블에서 가장 중요한 하이퍼파라미터는

"IoU threshold" 못지 않게 "score 계산 방식"이다.

- 특히 detection에서 AP는 threshold-sensitive하므로
  일관된 calibration을 유지해야 한다.

---

6. 개인 회고

6-1. 학습 목표와 실행

- 목표
  - 다양한 detection 아키텍처를 실제 대회 환경에서 실전 적용해 보고,
    "어떤 상황에서 어떤 모델이 강한지" 감각을 얻는 것.
  - mmdetection/mmengine 기반 pipeline을 자유롭게 커스터마이즈할 수 있는 수준까지 올라가기.
  - 최종적으로는 single 모델뿐 아니라 앙상블까지 스스로 설계·구현해 보는 것.
- 실행
  - 프로젝트 초반에는 Faster R-CNN → Cascade R-CNN → VFNet 순으로
    비교적 전통적인 모델들을 빠르게 돌려보며 baseline을 구축.
  - 중반 이후에는 DETR, Sparse R-CNN, TOOD, RTMDet 등
    다양한 구조를 시도하며 장단점을 직접 체감.
  - 마지막에는 팀원이 제작한 DDQ-DETR 계열 csv와
    내가 만든 DETA 계열 csv를 PredictionString 단위로 묶는
    ensemble_from_submissions.py를 직접 작성해
    box-level 앙상블을 구현.

6-2. 이번 프로젝트에서 새롭게 시도한 변화와 효과

- csv-level 앙상블 파이프라인 직접 구현
  - framework에 의존하지 않고, 단순 csv만으로 WBF/NMS 앙상블이 가능한 코드 작성.
  - 모델별 weight, IoU threshold, score threshold 등을 파라미터로 제어하도록 만들었고,
    이를 통해 ensemble_5(0.6138)라는 최종 결과를 얻음.

- mmengine 버전 이슈 직접 해결
  - DetInferencer가 cfg_options를 받지 않는 버전에서
    temporary config 파일을 생성해 num_classes를 패치하는 전략을 사용.
  - DetDataSample 객체 구조를 직접 확인하고,
    bboxes/scores/labels를 뽑아내는 작은 test 스크립트를 만들어
    한 이미지 단위로 디버깅한 경험이 이후 다른 프로젝트에도 큰 자산이 될 것 같다.

6-3. 한계와 아쉬웠던 점

- transformer 계열(DETR/DETA)을 충분히 살리지 못함
  - 학습 스케줄, augmentation, optimizer 조합을 더 적극적으로 바꾸지 못했고,
  - 결국 제대로 된 single submission score를 확보하기 전에
    time-budget이 소진되어버렸다.
- error analysis 부족
  - 시간 압박 때문에 per-class mAP, confusion matrix,
    difficult image set 등에 대한 정량적 에러 분석을 깊게 못 했다.
  - 그래서 어떤 class에서 ensemble이 이득인지,
    어떤 경우에 손해인지까지는 충분히 파고들지 못했다.

6-4. 다음 프로젝트에서의 계획

- detection 세팅 템플릿화
  - “COCO 스타일 10-class detection”에 최적화된
    기본 config 템플릿(Faster/Cascade/DETR/RTMDet)을 하나 만들어 두고,
  - 새로운 대회가 열리면 이 템플릿으로 바로 시작할 수 있도록 준비할 것.
- error analysis 자동화
  - per-class mAP, size-wise AP, IoU 분포 등
    error analysis 코드를 별도의 notebook/script로 만들어 두고,
    모델을 바꿀 때마다 자동으로 리포트를 뽑아보는 workflow를 만드는 것이 목표.
- 앙상블 라이브러리화
  - 이번에 만든 ensemble_from_submissions.py를 기반으로,

    - 다양한 detection 대회에서도 바로 plug-in 할 수 있는 “submission ensemble 툴”로 정리해 둘 계획.

이상으로, “재활용 품목 분류 Object Detection” 대회에서

내가 제출한 모델들을 중심으로 한 전체 실험 흐름과

성공/실패 사례, 그리고 그 과정에서 얻은 인사이트를 정리했다.

이 내용을 바탕으로 팀 랩업 리포트의 “프로젝트 수행 결과/자체 평가”와

“개인 회고” 섹션을 Notion에 옮겨 적으면 된다.

- label mapping mismatch
    - inference 결과 `unique labels: [0, 1, 6, 7]`와 같이 일부 클래스에만 집중되는 현상 발생
    - train.json 상 class 순서와 DETA 사전 학습 weight의 class index가 완전히 일치하지 않는 상황에서 metainfo 주입과 config patch만으로는 mapping이 깨끗하게 맞지 않았다.
- score calibration 문제
    - ensemble 전 개별 예측에서는 class 7에 대해 0.7~0.8대 confidence가 나오지만, fold 간 WBF/NMS를 거치면 score가 0.1 이하로 크게 떨어지는 현상 발생
    - 여러 모델이 같은 위치를 “약하게” 찍는 경우 평균/가중 평균으로 score를 뽑으면 과도하게 희석되는 문제가 있었다.