

재활용 품목 분류를 위한 Object Detection

AI로 쓰레기를 detection하여 지구를 구해주세요!

#부스트캠프4기 #컴퓨터비전 #비공개 프로젝트

D-Day | 2022.11.16 ~ 2022.12.01 19:00



Wrap Up Report - Object Detection

🕒 Created	@2022년 12월 1일 오후 4:59
🕒 Last Edited Time	@2022년 12월 5일 오전 11:47
📅 Week	Week11
☰ Status	Competition Log In progress
📅 Date	@2022년 12월 2일 → 2022년 12월 5일
👤 Created By	
👥 참여자	
➤ 같은 날 진행	
🔗 link	
📌 대회	Object Detection
➤ 부모 태스크 (마일스톤)	
➤ 자식 태스크(마일스톤)	

프로젝트 개요

대회 주제 및 개요

주어진 사진에서 쓰레기를 Detection 하는 모델을 만들어, 환경 부담을 줄일 수 있는 방법 중 하나인 분리수거에 적용해 '쓰레기 대란', '매립지 부족'과 같은 여러 사회 문제를 해결하고자 한다.

문제 해결을 위한 데이터셋으로는 10가지 종류의 쓰레기가 찍힌 사진 데이터셋을 사용한다.

평가 방법은 Test set의 mAP50(Mean Average Precision)로 평가한다.

(이때 Test set은 총 4871장으로, 이 중 50%만 public이고 나머지는 private이다.)

데이터셋의 구조

- 전체 이미지 개수 : 9754장 (train : 4883장, test : 4871장)
- 10 class : General trash, Paper, Paper pack, Metal, Glass, Plastic, Styrofoam, Plastic bag, Battery, Clothing
- 이미지 크기 : (1024, 1024)
- annotation file은 coco format으로(images, annotations)의 정보를 가지고 있다.
 - images
 - id: 파일 안에서 image 고유 id, ex) 1
 - height: 1024
 - width: 1024
 - filename: ex) train/0000.jpg

- annotations
 - id: 파일 안에 annotation 고유 id, ex) 1
 - bbox: 객체가 존재하는 박스의 좌표 ($xmin$, $ymin$, w , h)
 - area: 객체가 존재하는 박스의 크기
 - category_id: 객체가 해당하는 class의 id
 - image_id: annotation이 표시된 이미지 고유 id
- 단 test set은 train set과는 다르게 annotation 정보가 포함되어 있지 않고 이미지 정보만 가지고 있다.

활용 장비 및 재료

- VS Code, Jupyter Lab - IDE
- git, github - 버전 관리, 업무 분할 및 협업,
- Zoom, Slack, Notion - 협업

프로젝트 팀 구성 및 역할

김범준 : 2-stage model 실험. CV 별 Json 생성 및 실험. Augmentation 실험

백우열 : 2-stage model 실험. utiliy 기능, EDA 인사이트 관련 실험, Augmentation 실험

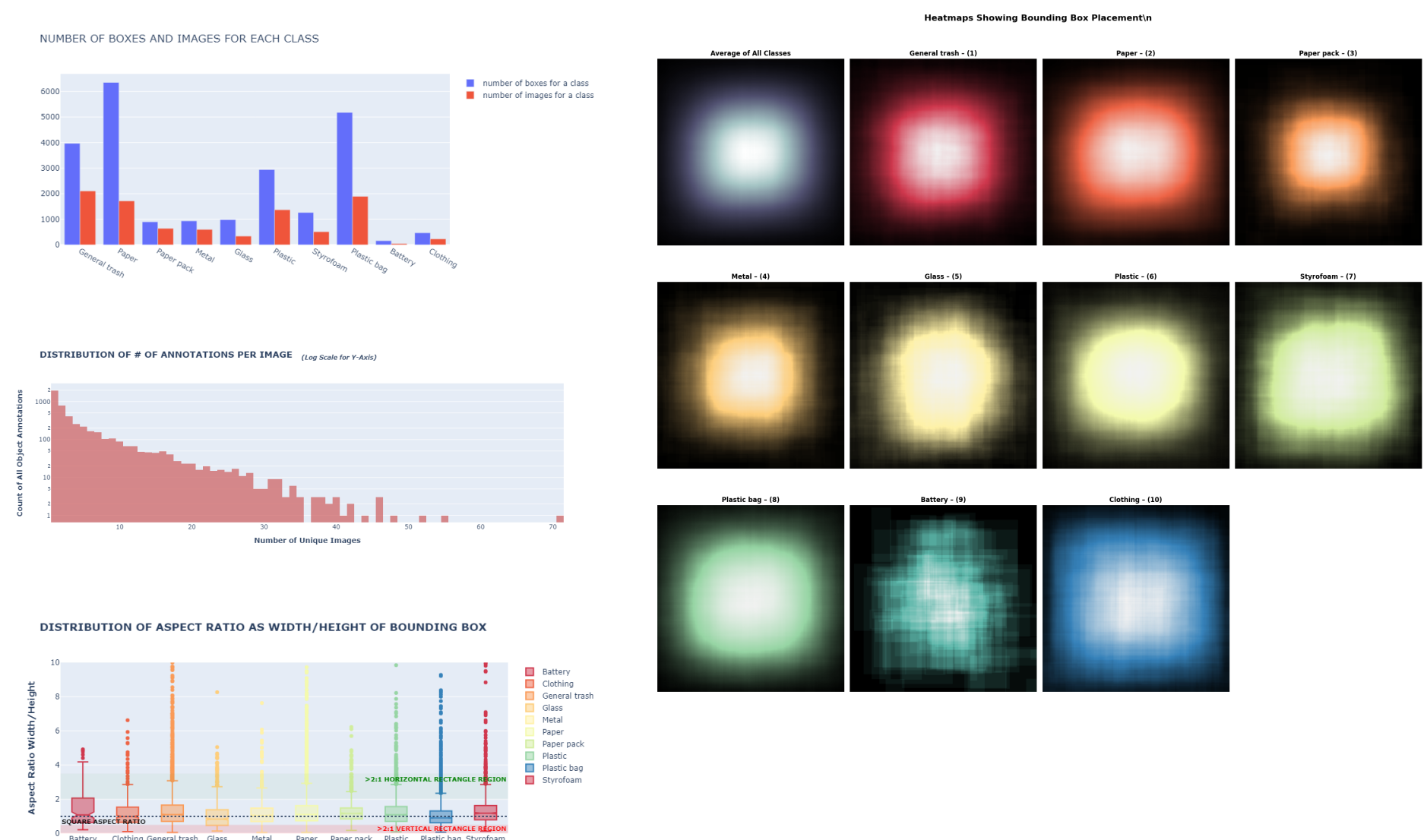
조용재 : 2-stage model 실험. Augmentation 실험, 각종 Template 작성.

조윤재 : 1-stage model (yolo v7) 실험, Pseudo Labeling 및 Ensemble 실험

최명헌 : 1-stage model (yolo v5) 실험. CV 별 Json 생성. Pseudo Labeling 및 Ensemble 실험

프로젝트 수행 절차 및 방법

EDA



EDA 후 얻은 Insight 및 실험 계획

1. 문제 : Class imbalance

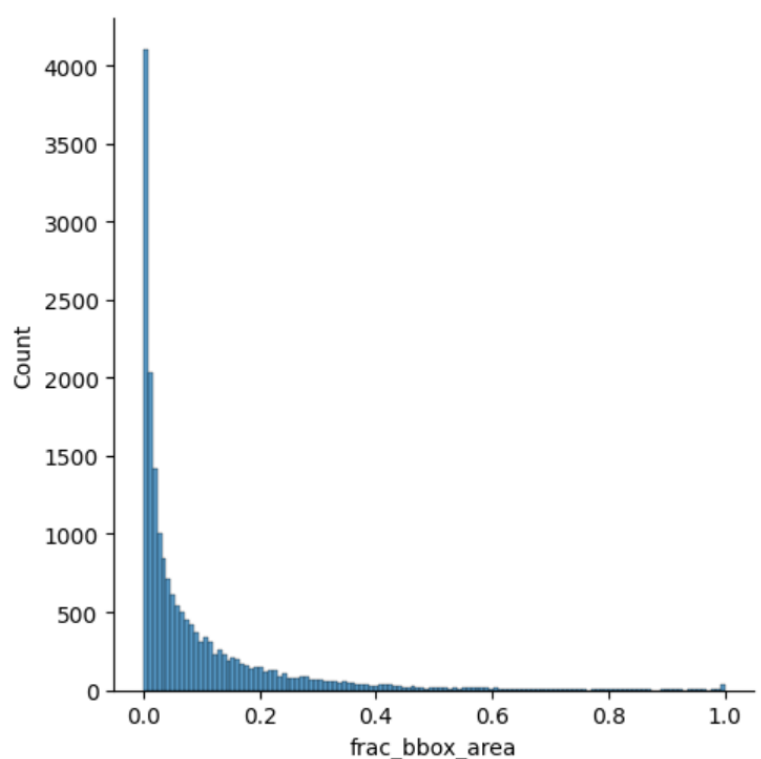
a. 가설 : Class 비율을 맞추는 CV strategy

- 이유 : Class 비율을 맞춰주면 편향된 학습을 할 확률이 줄어든 것이다.
- 전략 : `StratifiedGroupKFold` 를 사용해 Class 비율을 맞춘 5개 fold를 만들어서 train 및 valid를 진행했다. 모델은 *Resnet50-faster_rcnn* 사용.
- 결과 및 분석 : train 및 valid 성능에선 큰 변화가 없었지만, 제출 시 하락하는 점수 폭이 줄어들었다. (약 0.05수준)

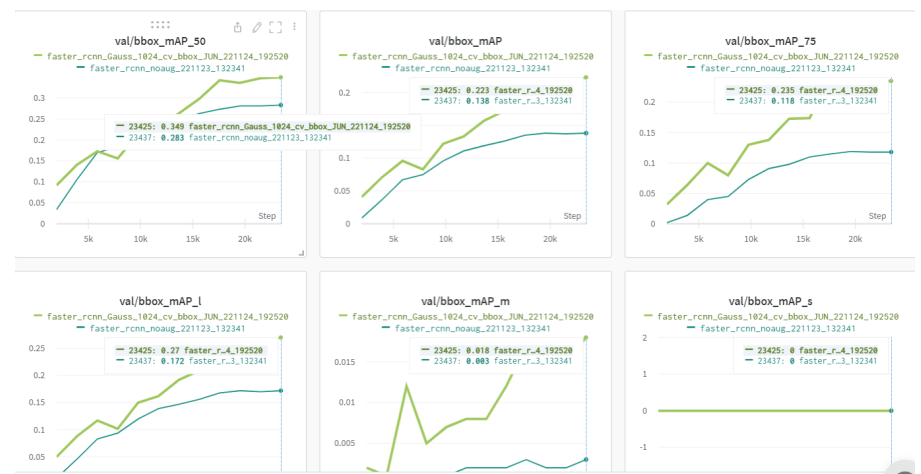
2. 문제 : bbox ratio imbalance

a. 가설 : bbox aspect ratio를 고려한 CV strategy

- 이유 : 학습 시 bbox 크기가 편향되어 있으면 bbox 크기에 따라 성능에 차이가 있을 것이라 추측.
- 전략 : $(x_{max} - x_{min}) * (y_{max} - y_{min}) / (1024 * 1024)$ 로 계산해 `frac_bbox_area` 항목 추가
구간 별로 'small, mid, large'로 나눈 후 `StratifiedGroupKFold` 를 사용해 5개 fold를 만들어 train 및 valid 진행했다.
모델은 *Resnet50-faster_rcnn* 사용.
- 결과 및 분석 : valid mAP50이 처음으로 0.3을 넘었고 그동안 해당 모델로 진행한 실험 중 가장 성능이 좋았다.
그러나 제출을 해보니 1번 실험과 비슷하게 점수 하락이 존재했다.
단, Swin_transformer-faster_rcnn으로 실험 진행 시 mAP50이 0.399에서 0.4303으로 상승했지만, 절대적인 점수가 낮아 해당 CV 유효하지 않다,
즉 test dataset을 잘 대표하지 못하는 것 같다는 결론내리고 후술할 3번 실험 진행했다.



새롭게 만든 속성인 frac_bbox_area 분포



Resnet50-faster_rcnn Wandb 결과

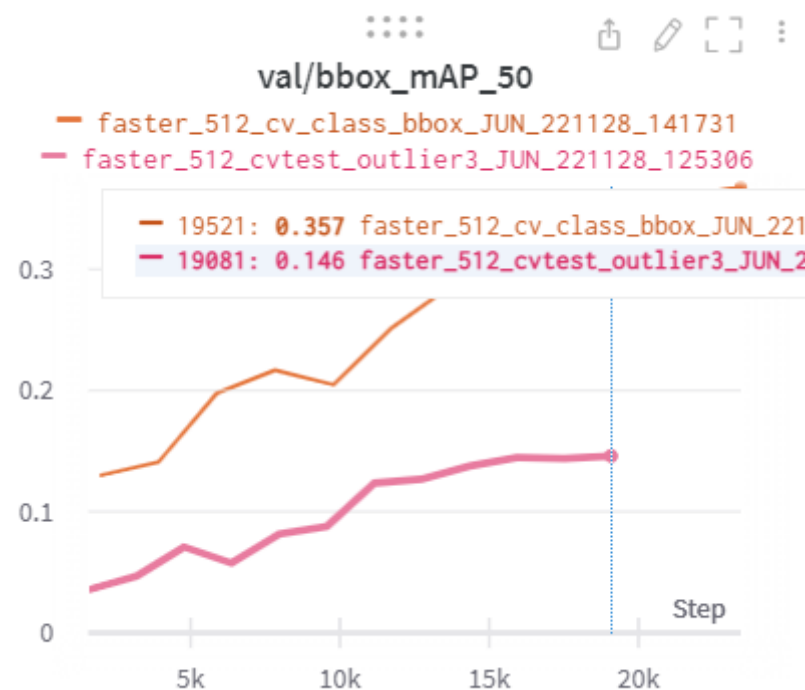
3. 문제 : Class & bbox imbalance (1,2번 실험 결과를 바탕으로 추가 실험)

a. 가설 : Class 와 bbox를 모두 고려한 CV가 성능 향상 및 test set 대표에 적합할 것이다.

- 이유 : Data가 균형을 이루기에 편향된 학습을 하지 않아 모델 성능이 올라가고, Generalization이 잘 돼 test set을 잘 반영할 것이다.
- 전략 : 1번, 2번 실험에서 진행한 Class 비율 및 bbox 비율은 동시에 적용해 5개 fold로 나누고 train 및 valid 진행했다.
- 결과 및 분석 : *Resnet50-faster_rcnn* (valid mAP50 0.368 → leaderboard mAP50 0.3770)과 *Swin_transformer-faster_rcnn* 모두 제출 시 점수 유지 및 상승을 가져왔다. 즉 해당 CV 전략을 토대로 한 train, valid set이 test set을 잘 대표한다고 볼 수 있다.

4, 문제 : 지나치게 많은 bbox 개수를 가진 image의 존재

- a. 가설 : 일정 개수 이상의 bbox를 가진 image를 제거한 후 학습을 진행하면 물체의 경계 파악 능력이 향상될 것이다.
 - i. 이유 : 한 개의 image의 너무 많은 bbox가 있는 경우
 - ii. 전략 : 35개 이상의 bbox가 존재하는 image를 제거한 후 CV(class+bbox) 진행.
 - iii. 결과 및 분석 : 급격한 점수 하락이 나타났다. 추측하건대 제거한 image에 예상보다 학습에 도움이 되는 bbox가 많았던 것 같고, 기존 Data의 수가 많지 않은데 거기에 더해 제거를 해버리니 모델 성능에 악영향을 준 것 같다.



Model

Object Detection 모델은 구동 원리에 따라 1-stage model, 2-stage model로 나뉜다.

우리는 다양한 모델로 대회를 진행하는 것이 실험을 통해 얻을 수 있는 insight를 더 많이 확보할 수 있을것이라 생각해 한가지 유형의 모델을 선택하지 않고 두가지 모두 사용하기로 결정했다.

1-stage model

- yoloV5 : 1-stage Model의 대표인 yolo 중 성능이 잘 나오면서 다양한 사람들의 버전이 마련되어 있어 다양한 실험을 할 수 있는 모델.
- yoloV7 : 1-stage Model의 대표인 yolo의 최신 버전. 학습 속도가 빠르고 성능이 보장된 모델.

2-stage model

- Resnet50-faster rcnn : 가장 가벼운 모델로 학습 속도가 빨라, 다양한 실험에 사용했다.
- Swin_Transformer-faster_rcnn, atss : 가장 높은 성능의 모델로 검증된 실험 결과를 적용해 최종 결과 도출시 사용했다.

Experiments

실험 진행 하며 얻은 아이디어 및 실험 계획

1. 문제 : Confusion Matrix 확인 시 Background Prediction이 많음

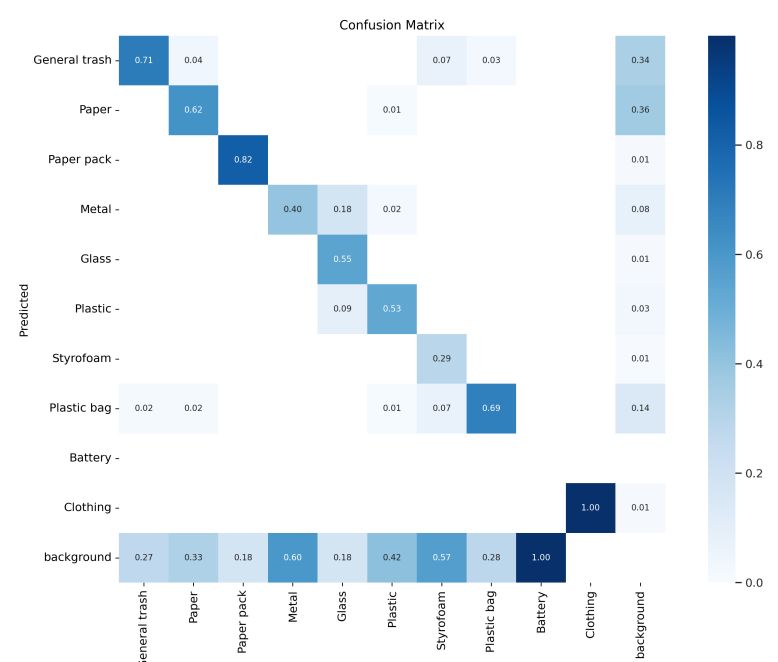
- a. 가설 : Small box를 잘 포착하면 background로 예측하는 경우가 줄어든 것이다.
 - i. 이유 : small box는 배경과 혼동되는 경우가 많을 것이다.
 - ii. 전략 : 이미 기존의 Data를 통해 학습시킨 모델에서 aspect ratio가 0.01 이하인 bounding box만을 따로 분류해 그 bounding box만을 사용해 재학습 시켰다. 일종의 small box를 잘 찾도록 fine-tuned 시키는 것이라 생각했다. 모델은 yoloV5 사용.
 - iii. 결과 및 분석 : 이미 사용한 train set에서 작은 bbox만 다시 학습시켰기 때문에 mAP는 0.743으로 매우 높게 나왔다. 하지만 이것에서 흥미로웠던 부분은 기존 학습에서는 잘 탐지했던 clothing, battery를 small bbox finetuned 시에 잘 탐지하지 못했고, 기존에 잘 탐지하지 못했던 general trash, paper pack들을 잘 탐지

했다. 하지만 이와 동시에 큰 물체에 대해서는 아예 탐지하지 못하고 작은 물체만을 탐지하는 결과를 보였다. 그래서 이 방법론을 끝까지 사용하지 못했지만 기존의 Data로만 학습한 모델과 작은 bbox로 학습시킨 모델을 앙상블하면 어떤 결과가 나올지 확인했으면 하는 아쉬움이 있다.

- b. 가설 : height/width ratio가 일정 비율 이상인 bbox를 제거하면 background로 예측하는 경우가 줄어든 것이다.
- i. 이유 : ratio가 0.5 이하 혹은 2 이상인 경우 bbox 안에 object 보다 background가 차지하는 부분이 많아 이를 제거한다.
 - ii. 전략 : $0.5 \geq$, $2 \leq$ 에 해당하는 ratio를 가진 bbox를 제거하고 EDA-3번 CV로 train 및 valid 진행. 모델은 *Resnet50-faster_rcnn* 사용.
 - iii. 결과 및 분석 : valid-mAP50 0.362에서 leaderboard-mAP50 0.3295 점수 오히려 하락했다. 추측하건대, 해당 ratio를 가진 bbox의 경우 bbox 내부에 background의 비율도 높지만, 그만큼 object의 형태가 명확해서 해당 bbox를 제거하면 보다 정제된 Data를 잃게 되어 점수 하락이 발생한 것 같다.



default setup 상태에서 confusion matrix



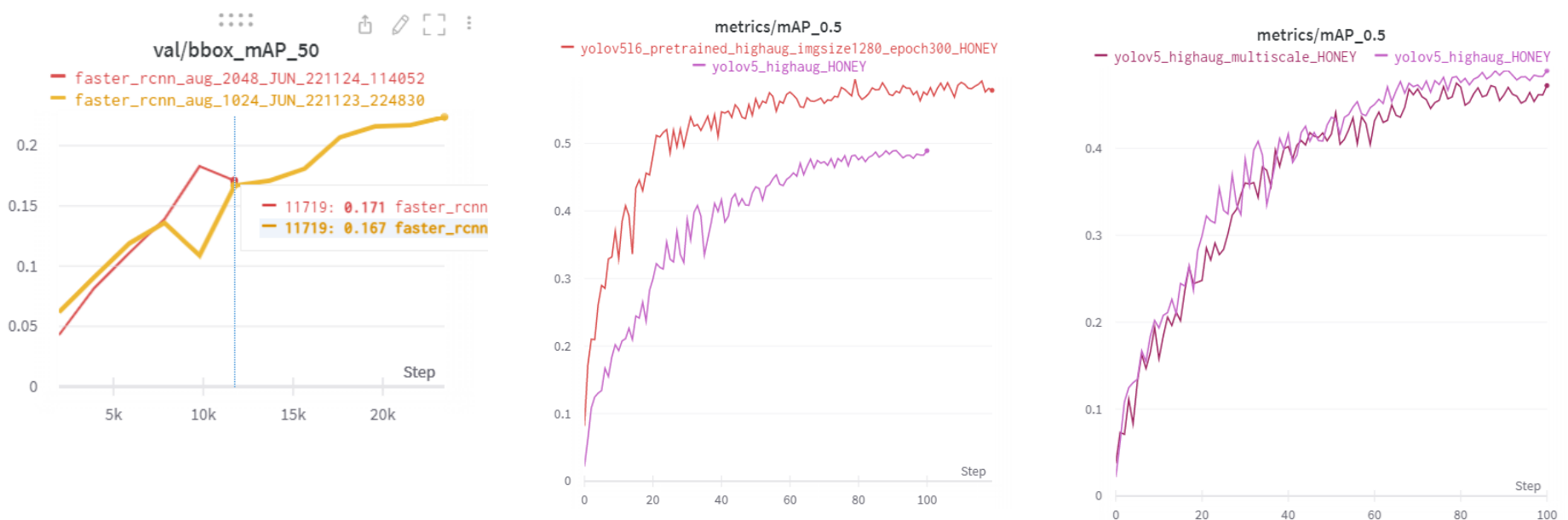
small bbox finetuned 이후의 confusion matrix

2. 문제 : Image Resolution

- a. 가설 : Image Resolution을 높이면 성능이 올라갈 것이다.
- i. 이유 : object의 크기 역시 같이 커지고 더 선명, 정확하게 보이기 때문이다.
 - ii. 전략 : default값인 512×512 를 1024×1024 , 2048×2048 로 키워서 학습한다. 모델은 *ResNet-faster_rcnn*
 - iii. 결과 및 분석 : resolution이 상승한 만큼 GPU 할당량 및 학습시간이 늘어났지만 그에 비례해 점수가 좋아지지 않았다. 1024×1024 는 약간 상승 혹은 같은 수준이었고, 2048×2048 는 오히려 하락했다. 학습시간 및 점수를 고려해, 이후 실험은 1024×1024 로 진행한다.
- b. 가설 : (yoloV5) Image Resolution을 높이면 성능이 올라갈 것이다.
- i. 이유 : object의 크기 역시 같이 커지고 더 선명, 정확하게 보이기 때문이다.
 - ii. 전략 : default값인 640×640 를 1280×1280 로 키워서 학습한다.
 - iii. 결과 및 분석 : resolution이 상승한 만큼 GPU 할당량 및 학습시간이 늘어났지만 그에 따른 성능 향상이 확연하게 보였다. 640×640 에서는 최고 mAP가 0.4892에 머물렀지만 1280×1280 으로 늘렸을 때 0.5922까지 mAP가 상승했다.
- c. 가설 : Upscaling을 하는 과정에서 보간법의 종류를 바꾸면 성능이 올라갈 것이다.
- i. 이유 : 이미지의 선명도가 올라간다면 Small object에 대한 detection을 더 잘하게 될 것이기 때문이다.
 - ii. 전략 : Upscaling은 기본적으로 양선형 보간법이 사용되는데 시간이 좀 더 오래 걸리는 방법이지만 Lanczos 보간법을 사용한다.
 - iii. 결과 및 분석 : 큰 성능의 변화를 보이지 못했다.

d. 가설 : (yoloV5) multi-scale을 사용한다면 성능이 좋아질 것이다.

- i. 이유 : 다양한 scale에서 모델을 학습시키면 이에 대한 robustness가 상승할 것이고, 다양한 Data를 받아들여 더 좋은 성능을 보일 것이다.
- ii. 전략 : 640*640 ~ 1280*1280 사이의 다양한 image resolution을 이용해 학습을 진행했다.
- iii. 결과 및 분석 : 640*640 단일 크기로 학습했을 때 보다 성능 하락을 보였다. 이는 pretrained 모델을 사용하는 yolo 특징에 따른 결과로 판단했다. yolo pretrained 모델의 경우, 640*640 크기 또는 1280*1280 크기의 단일 input size를 사용한 pretrained 모델이 존재했기 때문에 다양한 image resolution을 사용함으로써 pretrained 모델의 성능을 모두 활용할 수 없었던 것으로 판단했다.



3. 문제 : mAP_s이 mAP_m과 mAP_l에 비해 현저히 낮았고, 모델의 예측 결과도 confidence score ≥ 0.3 로 작은 물체를 예측한 bbox들이 적었다

a. 가설 : 이미지 tiling으로 사용하는 픽셀을 증가시키면 물체의 정보량이 많아져 클래스 분류시 confidence score가 증가할 것이다

- i. 이유 : inference시 1024*1024의 kxk 크기의 물체의 경우 bbox 면적이 $\frac{k^2}{1024^2}$ 이지만, 512*512의 kxk 크기의 물체의 경우 bbox 면적이 $\frac{k^2}{512^2}$ 로 CNN 통과시 면적이 약 4배 큰 feature mAP을 활용할 수 있다.
- ii. 전략 : 1024*1024 이미지를 512x512 이미지 4개로 분할해 각각에 inference를 진행 후 원본 이미지를 inference한 결과와 비교
- iii. 결과 및 분석 : confidence score ≥ 0.2 인 기준, bbox의 개수는 증가했지만 크기 변화로 몇가지 클래스를 혼동하는 문제(a-1)와 tiling으로 잘린 큰 물체를 tiling별 bbox 예측을 합치거나 제거해야하는 문제(a-2) 발생

b. (a-1) 가설 : 학습시 같은 클래스의 물체의 크기와 색상 정보를 다양하게 주면 물체 크기와 색상으로 클래스를 판단하지 않을 것이다

- i. 이유 : 물체의 크기와 색상에 따라 다른 클래스로 판단하는 경우 발생
- ii. 전략 : data pipeline에 `MinIoURandomCrop` 과 `RandomCenterCropPad` 로 크기 정보 변경, `PhotoMetricDistortion` 으로 색상 정보 다양화
- iii. 결과 및 분석 : 혼동하는 클래스의 종류가 증가했고 valid set의 mAP도 감소했고 물체의 색상 정보와 크기 정보가 클래스 판단에 중요하게 작용한다고 추측했고 개선 점을 찾지 못해 이후 진행은 보류했다.

c. (a-2) 가설 : 원본 이미지의 inference 결과와 score를 5로 나눈 tiled 이미지들의 결과에 IoU threshold는 약 0.2로 nms를 적용하면 같은 물체에 대해 여러 개 존재하는 bbox들을 제거할 것이다

- i. 이유 : 대부분의 tiling으로 잘린 큰 물체의 경우 중앙에서 4등분 된 경우가 많아 원본 이미지의 bbox와의 IoU가 약 0.25로 판단했고, tiled 이미지의 score가 원본 이미지의 score보다 높은 경우를 대비해 tiled 이미지의 score를 1/10을 곱해 인위적으로 낮추고 nms의 IoU threshold를 0.2로 하면 충분히 제거할 수 있다고 생각했다.
- ii. 전략 : 원본 이미지의 예측과 score에 1/10을 곱한 tiled 이미지의 예측을 IoU threshold = 0.2로 적용한 nms 수행

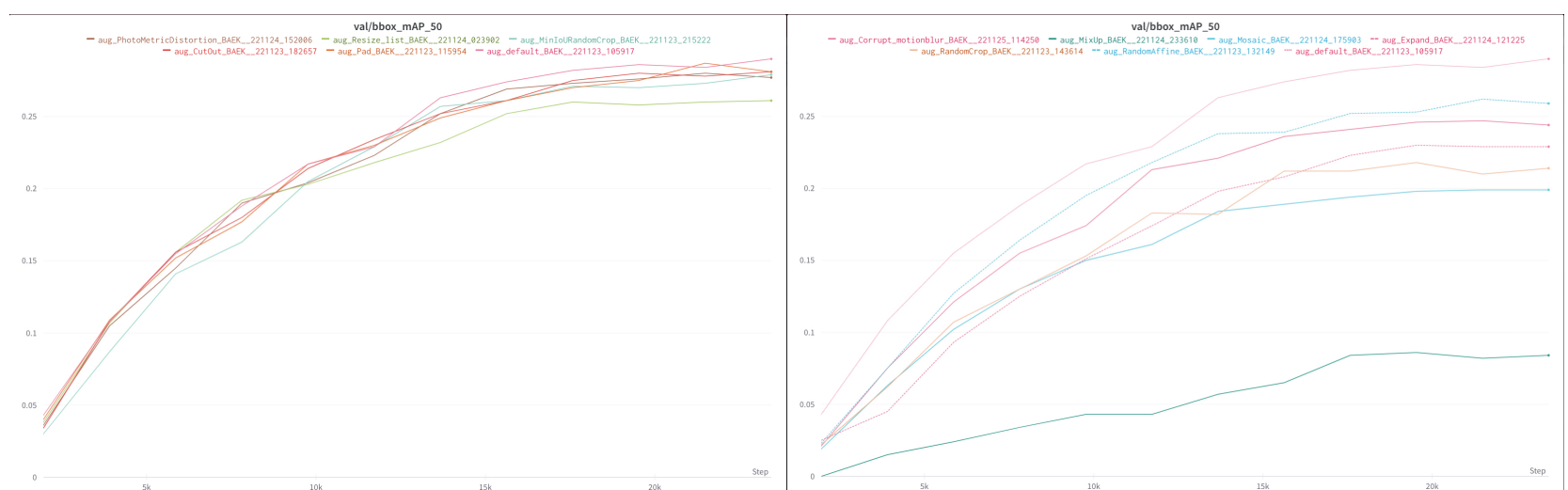
- iii. 결과 및 분석 : 큰 물체의 bbox안에 존재하는 작은 물체의 bbox도 같이 삭제되는 문제가 발생했고 결국 작은 물체의 bbox를 잃어버리게되면 tiling의 이점을 활용할 수 없다고 판단해 보류했다.
- d. 가설 : 이미지들을 모든 bbox에 해당하는 단일 image로 변환하여 학습시킨다면 학습에 필요한 물체들의 feature를 더 잘 잡아낼 것이다
 - i. 이유 : small box에 대한 mAP가 극도로 낮고, background prediction에 대한 이유로 feature를 혼동하기 때문이라고 가정
 - ii. 전략 : segmentation에서 사용하는 grabcut algorithm을 통해 모든 bbox에 대하여 단일 이미지로의 변환 후 학습을 꾀함
 - iii. 결과 및 분석 : bbox area에 대한 segmentation인 grabcut이 깔끔하게 되지 않았고, Data의 개수가 너무 많아져서 학습하는 데에 지장이 있어 중단하였다. 다음 번에 다시 사용해보고 싶은 방법

4. 문제 : 같은 클래스인 물체들의 bbox IoU > 0.5인 경우 nms로 가장 score가 높은 bbox를 제외한 bbox가 제거되는 현상

- a. 가설 : soft_nms를 사용하면 IoU > 0.5 이상인 bbox들의 confidence score를 변경하지만 낮은 score로 보존해 mAP를 증가시킬 것이다.
 - i. 이유 : nms로 bbox가 사라지는 경우 FN으로 항상 mAP가 감소하지만, soft_nms로 낮은 confidence score의 bbox는 TP or FP로 mAP 증가를 기대해 볼 수 있다고 판단
 - ii. 전략 : 모델의 test_cfg의 nms를 soft_nms로 변경
 - iii. 결과 및 분석 : 같은 모델의 checkpoint로 test 이미지에 대해 nms와 soft_nms로 각각 진행한 inference에서 Public 기준 0.5760 → 0.5764로 mAP가 소폭 상승했다

5. 문제 : 4883개의 train set의 불균형한 class와 bbox 크기를 고려해 **StratifiedGroupKfold**를 적용해서 발생하는 Data 감소

- a. 가설: Augmentation 을 사용해 Data 증강 효과를 기대할 수 있다
 - i. 이유 : 모델이 예측에 사용하는 이미지의 중요한 특성들을 제외한 다른 특성들을 변경하면 모델이 다른 이미지로 인식
 - ii. 전략 : mmdetection의 Data Pipelines중 Pre-processing 과정 실험
 - iii. 결과 및 분석 : **PhotoMetricDistortion**, **Resize** (MultiScale), **MinIoURandomCrop**, **CutOut**, **Pad** 는 mAP 하락이 없었고, **Corrupt** (motionblur), **MixUp**, **Mosaic**, **Expand**, **RandomCrop**, **RandomAffine** 은 mAP가 하락했다.



- b. 가설 : test set의 confidence score가 높은 예측 결과들을 train set으로 활용하면 labeled된 새로운 Data로 증강 효과 기대할 수 있다
 - i. 이유 : 모델의 test 이미지의 예측 EDA 결과 confidence score ≥ 0.4 이상인 bbox는 확실한 bbox 범위와 클래스를 갖는다고 판단해 test 이미지들을 학습에 사용
 - ii. 전략 : Pseudo Labeling 실험으로 train : test = 0:1, 1:1, 1:3 으로 각각 3 epochs 실험 후 가장 높은 mAP를 기록한 비율로 사용


iii. 결과 및 분석 : 세가지 비율 모두 valid set에서의 mAP 증가는 있었지만 Public score는 소폭 상승하거나 하락해 fold별로 선택적으로 적용했고 원인은 train set에 overfit 된 것과 score threshold > 0.4 이상인 결과만 사용했지만 모델 성능이 충분이 향상되기전이라 발생하는 오류들로 classification 성능이 저하된 것으로 추측했다

- private score 상승한 경우
 - pseudo label 전: valid: 0.629 → public: 0.5890 → private: 0.5774
 - pseudo label 후: valid: 0.648 → public: 0.6103 → private: 0.5987
- private score 하락한 경우
 - pseudo label 전: valid: 0.591 → public: 0.5760 → private: 0.5635
 - pseudo label 후: valid: 0.686 → public: 0.5620 → private: 0.5435

6. 문제 : 단일 모델의 오류와 mAP 한계

- a. 가설 : 모델별 특성을 고려해 여러 모델들의 공통적으로 예측한 결과를 사용하면 특정 모델의 오류를 감소할 수 있다
- i. 이유 : 모델별로 클래스 분류 기준이 달라 서로 다른 confusion matrix를 갖고 bbox 선정 기준도 다르기 때문
- ii. 전략 : confusion matrix 확인시 모델별로 유의미한 차이는 없다고 판단해 faster rcnn으로 실험적으로 얻은 nms, soft nms, wbf와 IoU threshold별 비교 결과 중 가장 좋았던 IoU threshold = 0.6의 WBF Ensemble 사용했습니다.
- iii. 결과 및 분석 : Ensemble시 단일 모델별 예측한 confidence score가 낮은 bbox들을 모으는 효과를 가져왔고, 단일모델 대비 약 0.1의 mAP 향상을 얻을 수 있었다.

프로젝트 수행 결과

mAP  (Rank)		Public 9th. Private 12th(9th).
0.6810 → 0.6635		(적을 내용 : 제출 파일 별 사용한 model, CV, Ensemble 등 내용 작성.)
0.6776 → 0.6602		• 단일 모델 제출 최고 점수 : 0.6103→0.5987 (Swin_transformer-atss)
0.6756 → 0.6587		• Ensemble 후
		◦ 0.6810 → 0.6635 : swin-l, yoloV5, yoloV7의 f0~4와 pseudo label 총 30개 WBF Ensemble
		◦ 0.6776 → 0.6602 : swin-l, yoloV5, yoloV7의 f0~4 총 15개 WBF Ensemble
		◦ 0.6756 → 0.6587 : swin-l f0, f2, f4와 yoloV5 f0~4 yoloV7 f0~4 총 13개 WBF Ensemble
9 (-)	CV_01조	

자체 평가 의견

잘한점

1. 가설→실험→분석→재실험의 과정을 지키려고 노력했고, 의사결정 역시 최대한 실험을 기반으로 진행하기 위해 노력했다.
2. 회의 중 안건들에 대하여 팀원들끼리 소통을 통하여 결정하려 하였고, 오프라인 미팅을 통하여 가까워지는 시간을 가졌다.
3. 점수 상승 및 순위에 매달리지 않고 계획한 실험을 진행하는 데 초점을 맞췄다.
4. 서로의 시행착오를 공유해 오류 해결하는 시간을 줄일 수 있었다.

아쉬운점

1. 경진 대회 초기에 태스크에 대한 어려움으로 인해 일정 관리를 철저히 하지 못해 첫 주를 아쉽게 보냈다.

2. 조금 더 많이 분업하여 효율적으로 시간을 사용할 수 있었을 것 같은데 다들 처음이다보니 조금 어려웠다.
3. 모델 학습 속도의 제약으로 더 많은 수의 실험을 하지 못했다.
4. 마지막 Ensemble 단계에서 제출 파일의 용량 및 서버 속도 이슈로, 계획했던 전략을 바탕으로 진행한 실험의 결과를 제출하지 못했다.
5. EDA를 좀 더 구체적이고 세부적으로 했으면 Data를 잘 파악했을 것이고 그에 따른 추가적인 전략을 세울 수 있었을 것 같다.
6. 모델 예측 결과의 충분한 분석과 모델 구조에 대한 고려 없이 mAP만 보고 문제를 정의하다보니 본질적인 문제에 접근하지 못해서, 실험 결과들을 하나의 결과로 수렴시키진 못한 것 같다.
7. 충분한 Augmentation 실험을 했다고 생각되지만 효과적인 방법을 찾지 못한 것 같다.
8. 계획 및 관리 방법에 대해 많은 이야기를 나눴지만 그만큼의 효율을 내지는 못한 것 같다.

프로젝트를 통해 배운점

1. 모든 의사결정은 느낌, 추측이 아닌 실험 진행 후 결과를 분석 한 후 이를 기반으로 해야지 합당하고 올바른 의사결정이 가능 하다.
2. Detection Task에서 가설을 세우고 실험을 하는 데에 많은 시간이 소요되기 때문에, 실험을 시도, 검증하는 과정을 빠르게 거치는 것이 중요해 보인다.
3. 생각보다 많은 기능이 라이브러리 내부에 구현되어 있는 경우가 많아, 이를 잘 활용하면 시간 단축을 할 수 있으니 관련 문서를 잘 읽어보는 것이 중요하다.

개인회고

김범준

이번 프로젝트에서 나의 목표는 무엇이었는가?

- 가설→실험→분석→재실험의 과정을 지키기 위해 노력했다. 즉 이유와 목적이 있는 코드 구동 및 실험을 하는 것이 이번 나의 목표였다.
- 저번 대회 때 해보지 않은 다양한 시도를 하며 경험을 쌓으려 노력했다.

나는 내 학습목표를 달성하기 위해 무엇을 어떻게 했는가?

- 위에 적은 실험 원칙을 지키기 위해 실험 전 구체적으로 기록한 후 실험 진행했다.
- Data Centric한 대회 참여를 위해 Data 분석에 힘을 쏟고, Pseudo Labeling, Ensemble 등 다양한 방법을 시도했다.

내가 한 행동의 결과로 어떤 지점을 달성하고, 어떠한 깨달음을 얻었는가?

- Data 분석에 더 많은 시간을 쏟아도 될 것 같다. 그만큼 많은 insight를 얻고 진행 중인 task에 대한 더 높은 이해도를 얻을 수 있기 때문이다.
- Ensemble이 점수 향상에 큰 역할을 하기에 대회에 참여할 때는 필수적이다. 단 이에 매몰될 수 있기에 대회를 마무리 시기에만 사용해야 한다.

전과 비교해서 내가 새롭게 시도한 변화는 무엇이고, 어떤 효과가 있었는가?

- 저번 대회에선 Tensorboard만 사용했는데 이번엔 wandb를 사용했고 덕분에 실험관리가 용이했다.
- 마무리 될 때 즈음엔 잘 지키지 못했지만, 실험 하기 전 해당 실험에 대한 내용을 구체적으로 기록하고 실험을 진행했다. 그래서 각 실험 별 결과를 비교하기 수월했다.

마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

- 필요한 기능을 직접 코드로 구현하며 해당 능력을 키우고 싶었는데, 이미 라이브러리내에 구현되어 있는 경우가 많고 또 대회 기간이 짧다 보니 가져다 쓰는데 급급했던 것 같다. 물론 하나하나 다 내가 구현하는 것이 대회에 참여하는 현명한 방법은 아니지만, 이미 만들어진 블록을 재조립하는 느낌이 들어서 아쉬웠다.

- 이론을 깊게 공부하지 못한 것이 아쉽다. 지금 당장의 실험 진행이 급하니 개요 정도만 파악하고 추가적인 노력을 하지 않아 Object Detection 모델에 대한 이해도 가 낮은 것이 아쉬워 추후에 추가적인 학습을 진행할 예정이다.

한계/교훈을 바탕으로 다음 프로젝트에서 스스로 새롭게 시도해볼 것은 무엇일까?

- 대회 진행 계획을 구체적으로 짜서 실험원리를 지키며 체계적인 실험을 할 것이다.
- 팀원끼리 좀 더 세밀한 분업화를 해서 실험 속도를 높이겠다.
- 실험 기록 template을 만들어 작성하는데 수고를 덜고 기록 형식을 통일해보겠다.

백우열

이번 프로젝트에서 나의 목표는 무엇이었는가?

- 협업으로 개인으로는 달성하지 못할 성과를 내보고 싶었다.
- 많은 실험으로 모델의 성능 향상에 기여할 수 있는 결과를 찾아 적용해 보고 싶었다.

나는 내 학습목표를 달성하기 위해 무엇을 어떻게 했는가?

- 팀원들과 분업을 했고, 공통의 과정에서는 서로의 시행착오를 공유해 오류 해결하는 시간을 줄이고자 했다.
- 실험 진행시 현재의 문제점을 부분 문제로 나누고 각 부분문제를 해결할 수 있는 방법을 찾는 실험을 하고자 노력했다.
- 모델 구조나 더 좋은 모델을 찾는데 시간 쓰기보다는, train set EDA와 모델의 예측 EDA를 통해 Data 전처리 방법, 불필요한 정보 삭제하는 방법, 그리고 Augmentation 방법을 고민하는데 시간을 사용했다.
- 성능 향상이 된다고 알려진 일반적인 방법들에 시간을 사용하기보다는, 현재 성능 향상을 막고 있는 문제가 무엇인지 먼저 파악하고자 했다.

내가 한 행동의 결과로 어떤 지점을 달성하고, 어떠한 깨달음을 얻었는가?

- 각자가 했던 실험 결과들을 공유해 단일 모델의 성능을 향상시켰고 최종적으로 Ensemble을 통해 개인이 달성할 수 없다고 생각한 mAP 0.67 이상을 달성했다.
- 실험 사전 구상과 신중한 valid set 설정을 기반으로 비교 가능한 최대한 많은 실험을 하고 이를 기반으로 다음 실험 진행 방향을 설정해 나가는 것의 중요성을 느꼈다.
- 모델의 평가 지표와 예측 결과를 모두 고려해 문제 정의를 시도했지만, 낮은 평가 지표와 예측 결과 오류를 서로 다른 문제로 판단해 효율적인 접근을 하지 못했고 신중한 문제 정의의 필요성을 다시 한번 깨달았다.

전과 비교해서 내가 새롭게 시도한 변화는 무엇이고, 어떤 효과가 있었는가?

- github Issue를 사용해 해결해야하는 문제들을 명확히 하고 확실하게 마무리 지을 수 있었다.
- github을 대회중에도 좀더 활발히 사용해 코드 빠른 코드 공유를 할 수 있어 시간을 많이 절약한 것 같다.
- 실험을 시도하기 전에 유사한 실험에 대해 서칭을 먼저 한 후 발생할 오류와 실험에 사용될 시간과 제출 횟수등을 고려하고난 후 진행했다.

마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

- eda로 얻은 인사이트로 정의한 하나의 문제를 부분 문제들로 나눠 해결하고자 했지만, 부분 문제들에 대한 결과를 하나의 문제를 해결하는 결과로 합치지 못했던 점이 아쉽다.
- 실험 비교 방식에 대한 고민이 부족했었고 중간에 실험 비교에 대한 고민으로 gpu가 쉬는 시간이 많아 많은 실험을 하지 못한 점이 아쉽다.

한계/교훈을 바탕으로 다음 프로젝트에서 스스로 새롭게 시도해볼 것은 무엇일까?

- 실험 계획 구상시 대회 플로우를 먼저 보고 단계별 기한을 정해 잘 안풀리더라도 그 기한내에서는 최대한 깊게 고민해보자.
- 하나의 방법에 두세번의 시도로 안풀린다고 넘어가지 말고 해당 논문 서칭등으로 좀 더 깊게 고민해보자.
- 실험 비교 방식을 체계적으로 설정해 현재 성공한 실험의 실험군이 바로 다음 실험의 대조군이 될 수 있도록 설계해보자.

조운재

이번 프로젝트에서 나의 목표는 무엇이었는가?

- 레벨2 팀원들과 처음 하는 협업으로, detection이라는 task는 팀원들 모두가 생소한 것이기에 성과도 물론 중요하지만 손발을 맞춰보는 것을 중점으로 하였다.

- 개인적으로는 detection에 대한 이해를 우선으로 하였다. 지속적으로 공부하고 싶은 분야를 정해야하는 나로서는, 모든 과정을 한 번씩 경험하며 가장 나와 맞는 것을 찾아야 하기 때문이다.

나는 내 학습목표를 달성하기 위해 무엇을 어떻게 했는가?

- 팀원들이 소통하는 방식은 무엇인지, 혹은 하고 싶은 것은 무엇인지를 파악하며 각각이 원하는 작업을 할 수 있도록 하였다. 협업을 하는 데에 나보다 더 익숙한 팀원들로부터 github나 notion 사용을 더 배울 수 있었다.
- Detection을 알기 위하여 강의를 우선적으로 듣고, 나중에 대회를 진행하였다. 이 task는 잘 정제되어 있는 Data로도 상당히 많은 시간이 걸리고, 마주하는 한계점들이 있었다. 팀원들과의 이야기를 통하여 한계점을 보완하려고 하였고, 혼자였으면 떠올리지 못했을 아이디어들이 많이 나와서 이해에 도움이 되었다.

내가 한 행동의 결과로 어떤 지점을 달성하고, 어떠한 깨달음을 얻었는가?

- 여러 가지 CV Strategy와 Validation Set을 선정하는 과정을 통하여 Data의 중요성에 대하여 알게 되었다.
- mmdetection과 yoloV7 라이브러리를 참고하여 코드를 돌려보고, 어떠한 모델을 사용하는 것이 좋을지에 대해 고찰하였다. 강의에서 들은 swin transformer 기반의 2 stage model을 시도하여 좋은 성능이 나오는 것을 확인하였다.

전과 비교해서 내가 새롭게 시도한 변화는 무엇이고, 어떤 효과가 있었는가?

- 이전 대회에서 아쉽게 SMOTE oversampling 사용을 해보지 못하여 해보고 싶었다. 그러나 이번 데이터셋은 불균형이 심함에도 불구하고 classification 문제는 없었다. Tiling과 같은 기법을 사용했을 때 SMOTE를 함께 사용했으면 classification이 덜 되는 문제를 해결할 수 있지 않았을까 생각이 든다.
- 기존에 주어진 baseline에 만족하지 않고 다른 library를 찾아 적용해보려고 노력하였다. Tiling 아이디어를 통해 팀원이 찾은 yolo-tiling이라는 라이브러리를 사용하여 실험을 해 보았는데, 성능이 오르지 않았다.
- Baseline이나 Library에 의존하지 않고 직접 코드를 짜서 기능을 구현해보려고 노력하였다. Grabcut Algorithm을 사용하여 이미지 한 장당 존재하는 bbox내 물체 이외에 다른 부분을 모두 배경으로 처리하여 검은색으로 만드는 작업이었다. Bbox area 내에 물체를 잘 잡는 편이었지만, 물체에 변형을 주고 학습할 이미지가 과도하게 많아지는 등 한계점이 존재하였다.

마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

- 실험을 기반으로 한 의사결정을 하고싶었지만 미리 여러 가지를 생각해놓고 순서대로 행하는 방식을 채택하지 못했다. 따라서 각자의 gpu가 쉬는 시간이 길었다.
- 나의 협업 실력이 생각보다 별로였다. Github 사용에 있어 미숙한 점을 보여 팀 전체에 혼란을 야기할 뻔 하였다.
- 내가 해보고자 하는 것, 우리 팀이 해봤으면 좋겠는 것들이 옳은 생각인지를 검증해보고 이야기하려다가 놓친 것이 아쉽다.

한계/교훈을 바탕으로 다음 프로젝트에서 스스로 새롭게 시도해볼 것은 무엇일까?

- 회의를 할 때 당장 해야할 것을 정하는 데에서 그치지 않고, 앞으로 무엇을 할 것인지에 대해 이야기하고 팀원들이 분업하여 각 실험을 행할 수 있도록 한다.
- Git과 Github, Issue와 같은 것들에 익숙해지고, 사용하는 법을 더 잘 익혀서 팀원들과 소통하는 데에 어려움이 없게 한다. 또한, 프로젝트에 대한 실험은 협업 툴을 사용하여 항상 기록하고 정리한다.
- 떠오르는 아이디어가 있으면 역시 Slack, Notion, Github Issue와 같이 여러 툴을 이용하여 팀원들이 그것에 대해 생각하고 의견을 이야기해줄 수 있게 한다.

조용재

이번 프로젝트에서 나의 목표는 무엇이였는가?

- 이전 대회에서 아쉬웠던 실험 계획 및 관리하는 부분을 좀 더 체계적으로 진행하고 싶었다.
- 계획적으로 접근하지 않고 생각나는대로, 그냥 되는대로 식의 접근을 방지하고 싶었기에 기록과 계획에 집중하려 했다.
- 문제 정의에 대해 수시로 고민하면서 진행하려고 했다.
- 그리고 영상처리 개념을 실제로 많이 적용해보고 싶었다.

나는 내 학습목표를 달성하기 위해 무엇을 어떻게 했는가?

- 노션에 마일스톤을 만들어서 큰 목표와 작은 목표를 세분화시켜서 관리하고자 하였다.
- 지금 고민하고 있는 논점이 어디서부터 파생됐고, 고민하기에 적당한 주제인지를 계속 자문자답을 했다.

내가 한 행동의 결과로 어떤 지점을 달성하고, 어떠한 깨달음을 얻었는가?

- 사실상 긍정적으로 도움이 되는 시도를 많이 해보지 못했다.

- 성능을 올리기 위해선 사소한 Augmentation 보다는 Data 전처리와 CV전략이 정말 중요하다는 것을 느꼈다.

전과 비교해서 내가 새롭게 시도한 변화는 무엇이고, 어떤 효과가 있었는가?

- 마일스톤을 도입한 것이었다. 지금 하고있는, 해야 할 업무에 대해 가시적으로 인지할 수는 있었지만 아직 본인 스스로도 그 령고 팀이 함께 적용하기에 어색한 부분이 많아 효과가 있었다기엔 조금 아쉬웠다.
- 비슷한 느낌의 깃허브 이슈를 작성한 것은 그래도 꽤 업무 분담과 결과에 대해 잘 관리를 하려 해서 익숙해지면 더 좋은 소 통 도구가 될 것 같다.

마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

- WandB에 이미지와 confusion matrix를 업로드 할 수 있게 한다고 말을 했지만 이 부분을 해결하지 못했다.
- mmdetection의 폴더와 코드 구조에 대해 제대로 이해를 하고 있지 못해서 어느 함수에 어떤 문제가 생긴 것인지 정확히 파악을 못한 것 같다.
- EDA에 시간을 더 투자 했어야 했는데 안되는 것을 포기 못하고 붙잡고 있다가 타이밍을 많이 놓쳐서 너무 아쉬웠다.

한계/교훈을 바탕으로 다음 프로젝트에서 스스로 새롭게 시도해볼 것은 무엇일까?

- 시간을 정하고 최대한 그 시간 내에 정한 문제를 해결하려 노력하지만 안될 경우에는 팀원의 도움을 받고 빠르게 포기하고 다 른 태스크에 집중할 마인드가 필요한 것 같다.
- 필요하다면 다른 사람의 페이퍼 요약을 참고하더라도 어느 정도 모델이나 방법론에 대한 이해도가 필요하다는 것을 느꼈다.

최명헌

이번 프로젝트에서 나의 목표는 무엇이였는가?

- 가장 첫 번째로 처음 마주하는 object detection에 대해 확실히 이해하고 Data에 맞는 Object Detection 모델을 구 성하고 그 모델을 구성하기 위한 라이브러리에 대한 이해를 확실히 하는 것이었다.
- 또한 “무지성”으로 모델을 구성하고 다양한 방법론을 적용하는 것이 아닌 실험을 통한 결과 분석과 그에 따른 보완점을 파악 한 후, 그에 맞는 방법론을 적용하여 문제점을 해결하는 흐름으로 프로젝트에 임하기를 원했다.
- 또한 이전 프로젝트에서는 git을 이용한 협업, 브랜치 전략 및 관리가 소홀했다고 느꼈기 때문에 이를 철저히 따르며 협업을 하고 싶었다.

나는 내 학습목표를 달성하기 위해 무엇을 어떻게 했는가?

- 최대한 confusion matrix, Validation set에서 나온 bbox를 통해 현재 학습시킨 모델의 문제점이 무엇인지 확인하려 노력했고, 그에 따른 문제 해결 방법을 고안하기 위해 노력했다. 하지만 단순히 confusion matrix를 확인하는 것이 모든 결과 분석은 아니었기 때문에 이를 class 단위로 나누어 생각해보거나 bbox의 크기로 나누어 문제를 파악하는 것 같은 섬 세한 결과 분석이 잘 되지 않았던 것 같다. 이는 task와 사용한 라이브러리에 대한 이해가 부족함에서 기인한 문제라고 생 각하기 때문에 이를 계기로 보충 공부가 필요할 것 같다.
- 팀 내에서 정한 git convention을 지키며 다른 팀원이 내가 사용한 브랜치를 사용하고자 한다면 사용할 수 있게끔 관리하 려 노력했다. 하지만 git 안의 issue 같은 부분은 아직 미숙하여 관리가 잘 되지 않았던 것 같다.

내가 한 행동의 결과로 어떤 지점을 달성하고, 어떠한 깨달음을 얻었는가?

- 다양한 CV strategy를 제안했고, 이를 코드로 구현하여 팀원들에게 나눠주어 통일된 CV strategy 상에서 실험을 전개할 수 있었다.
- yoloV5 모델을 사용하여 단일 모델로써 꽤 괜찮은 성능을 보였고 이를 사용한 앙상블을 통해 좋은 성능을 보였다.
- 모델 결과 분석을 통해 small bbox finetune, scheduler에 대한 제안 등 다양한 Ideation을 제안했다.

전과 비교해서 내가 새롭게 시도한 변화는 무엇이고, 어떤 효과가 있었는가?

- 이전 대회에서 TensorBoard를 통해 실험 결과를 관리했는데 이는 결과 공유 측면에서 부족한 점이 많았다. wandb를 사용 하여 팀원간의 결과 공유를 용이하게 하여 진행상황 파악과 결과 분석에 더 좋은 영향을 끼쳤다.
- git convention 및 branch 관리 전략을 더욱 철저히 따르며 협업 측면에서 도움이 더 되었고 더 많이 배웠다.
- 새로운 library를 보며 그에 대한 이해를 하려 노력했고, 그 library에서 사용되는 parameter, metric에 대한 부분 들에서 의문이 생길 때마다 submodule에 대해 확인하며 이해도를 높이려 했다.

마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

- 첫 번째로는 Data-Centric 한 분석이 부족했던 부분이다. 1주 동안 Data에 대한 이해도를 높이고 어떤 부분을 신경써야 하나 고민을 많이 했지만 많은 강의에 대한 이해와 모델링에 대한 부담감으로 인해 EDA 과정이 부족하여 계획했던 Data-Centric한 분석이 부족했던 부분이 아쉬웠다.

- 두 번째는 모델에 대한 이해가 부족한 상태로 좋은 모델이라는 이유 하나로 모델링을 진행했던 것 같다. 그렇기 때문에 모델 customizing에서 많은 시간과 노력이 필요했던 것 같고, 성능 향상에 있어서 놓쳤던 부분이 많았던 것 같다. 또한 이에 연결되는 문제로서 library 에 대한 이해 또한 부족했던 것 같다.

한계/교훈을 바탕으로 다음 프로젝트에서 스스로 새롭게 시도해볼 것은 무엇일까?

- Data에 대한 이해를 확실히 하고 진행하는 것이 중요할 것 같다. 다양한 시각에서 프로젝트를 진행하며 신경써야 할 부분을 체크하여 더 효율적으로 시간을 활용하여 실험을 계획할 수 있을 것 같다.
- 또한 논문이나 강의를 통한 모델에 대한 철저한 이해가 뒷받침 된 후에 모델링을 해보는 것이 더 좋은 결과와 결과 분석에 용이할 것이라고 생각한다.