부스트캠프 AI Tech 5기 CV-10 형셋누나하나

김보경(T5033), 김정주(T5052), 양재민(T5123), 임준표(5175), 정남교(T5188)

1. 프로젝트 개요

A. 개요

대량 생산, 대량 소비의 시대. 우리는 많은 물건이 대량으로 생산되고, 소비되는 시대를 살고 있습니다. 하지만 이러한 문화는 '쓰레기 대란', '매립지 부족'과 같은 여러 사회 문제를 낳고 있습니다. 분리수거는 이러한 환경 부담을 줄일 수 있는 방법 중 하나입니다. 문제 해결을 위한 데이터셋으로는 일반 쓰레기, 플라스틱, 종이, 유리 등 10 종류의 쓰레기가 찍힌 사진 데이터셋이 제공됩니다. 이를 이용하여 사진에서 쓰레기를 Detection 하는 모델을 만들어 이러한 문제점을 해결해보고자 합니다.

B. 환경

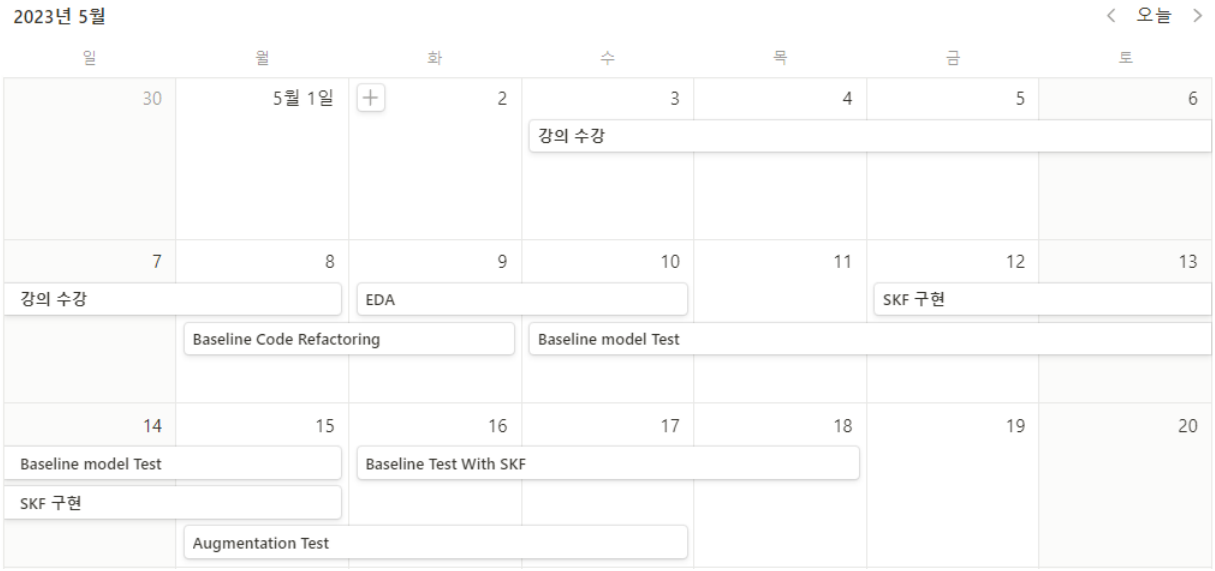
- 팀 구성 및 컴퓨팅 환경 : 5인 1팀, 인당 V100 GPU 및 AI Stages 서버
- 개발 환경 : VSCode, Jupyter Notebook
- 개발 언어 : Python, PyTorch
- 협업 툴 및 기타 : Slack, Notion, Github, Wandb

2. 프로젝트 팀 구성 및 역할

공통	Data EDA, Baseline Test, Ensemble
김보경	Baseline Code 리팩토링, wandb 연동, AI 모델 리서치
김정주	YOLO v8 실험, YOLO 전용 SKF 구현, Augmentation 실험
양재민	Baseline with Stratified K-Fold 리팩토링, Ensemble 적용
임준표	Augmentation 실험 및 적용, AI 모델 리서치 및 실험
정남교	SKF를 활용한 Data Split 구현, AI 모델 리서치 및 실험 설계

3. 프로젝트 수행 절차 및 방법

A. Timeline



B. 데이터셋 구성 요소

i. Image Data

1. train: 4883장의 Training용 train image
2. test: 4871장의 Validation용 test image
   - id: 파일 안에서 image 고유 index, ex) 1
   - height: 1024
   - width: 1024
   - filename: ex) train/002.jpg

ii. Annotation Data

1. train.json: train image에 대한 annotation file (COCO format)
2. test.json: test image에 대한 annotation file (COCO format)
3. id: 파일 안에 annotation 고유 index, ex) 1
   - bbox: 객체가 존재하는 박스의 좌표 (xmin, ymin, width, height)
   - area: 객체가 존재하는 박스의 크기
   - category_id: 객체가 해당하는 class의 id
   - image_id: annotation이 표시된 이미지 고유 index

i. Class Label

0-General trash, 1-Paper, 2-Paper pack, 3-Metal, 4-Glass, 5-Plastic, 6-Styrofoam, 7-Plastic bag, 8-Battery, 9-Clothing

C. Baseline Code

i. MMdetection

많은 모델들이 포함되어 있고, 모듈화가 잘 되어 있어 사용하기 편리하며, 또한, SOTA 모델에 대한 접근성이 높고, detectron와 같은 다른 플랫폼들보다 속도가 월등히 빠른 mmdetection을 이용하여 실험을 진행하였다.

ii. Baseline Code Refactoring

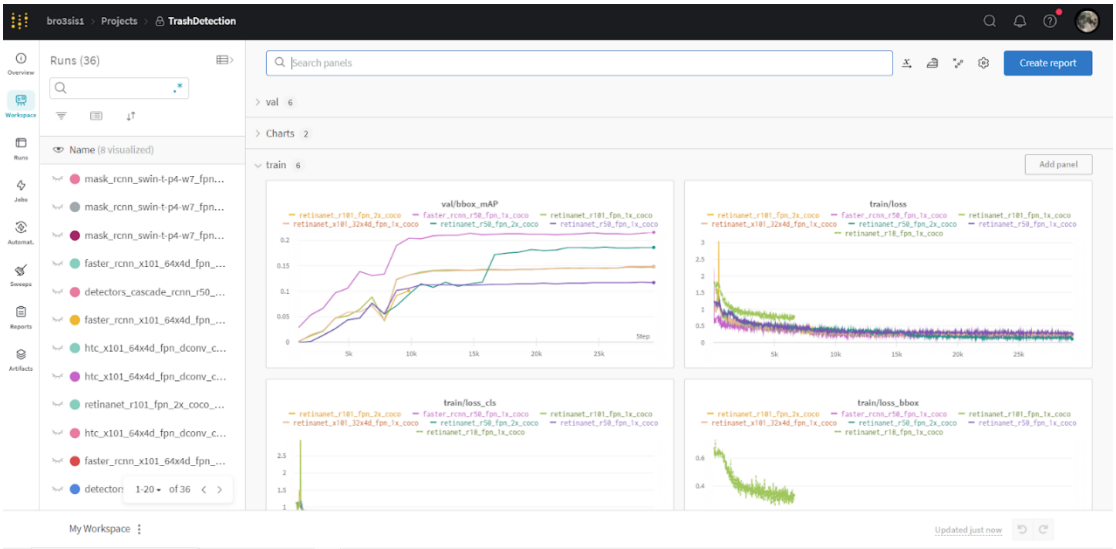
제공된 Baseline 코드를 기반으로 우리 팀만의 Baseline 스크립트를 만들었다. 총 3가지 py파일(train.py, inferece.py, ensemble.py)로 구성되어 있으며, 스크립트 파일을 만든 주된 이유는 아래와 같다.

- Argparse, inference와의 호환성

  baseline model을 테스트하는 과정에서 각자 사용하는 모델, loss등 각기 다른 하이퍼파라미터를 활용해야만 했다. 따라서 argparse를 활용하여 모든 실험에 대해 동작하는 하나의 스크립트를 만들었다. 또 train과 inference에서 사용하는 argument가 다르다면 혼동을 주거나 불편함을 초래할 수 있다고 생각하여, 매개변수를 통일하는 등 호환성을 높이고자 하였다.

- Wandb

  실험 공유가 중요한만큼 wandb로 실험 결과를 공유하고자 했다. Mmdetection의 hook을 통해 wandb연결을 하였고, 그 결과 실험을 실시간으로 확인할 수 있었다. 또 결과 비교를 손쉽게 할 수 있었다.



- 디버깅 툴 활용

  ipynb 코드도 디버깅을 할 수 있지만, 보다 강력한 디버깅 툴인 vscode의 디버거를 활용하기 위해 스크립트 파일을 생성하였다. 그 결과 config파일 외의 파일을 수정하였고, mask 데이터가 없음에도 불구하고 maskRCNN을 돌릴 수 있었다.

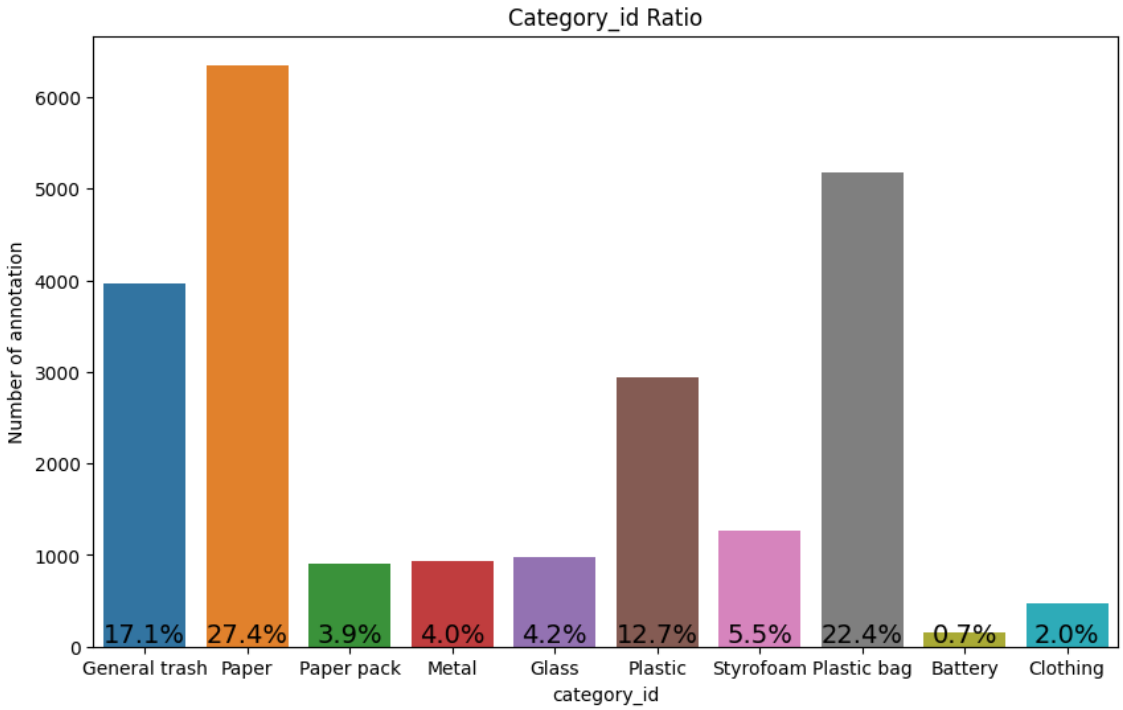
iii. YOLO v8 Open Source 사용

가장 대표적인 1-Stage Detector로 알려진 YOLO 계열 모델의 가장 최신 version으로써, C2f Module 및 Anchor-Free algorithm을 사용하여 NMS의 속도를 높인 SOTA Network이다. 본 competition을 수행하며 좋은 성능을 보일 수 있을 것이라 기대하여 선택하게 되었으며, Ultralytics의 Github에서 제공하는 Open Source를 customizing하여 사용하였다.

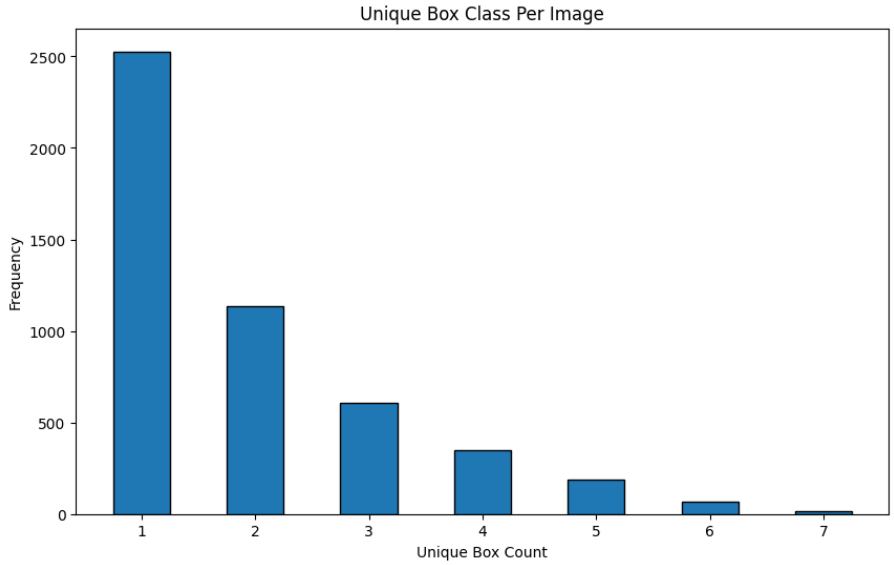
4. 수행 결과

A. Data EDA

i. EDA for Class Label

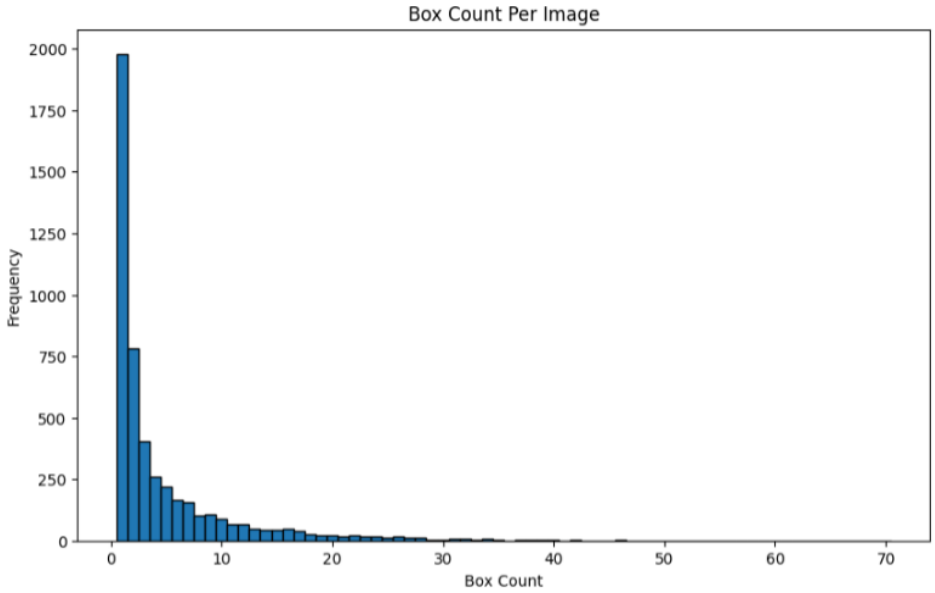


전체 Ground Truth Class Label에 대하여, 각각의 Label당 몇 개의 Instance들이 존재하는지에 대한 비율과 개수를 확인하였다. Paper와 General trash 및 Plastic bag이 가장 많으며, Battery의 경우 개수가 적어 Class Imbalance를 확인하였다.

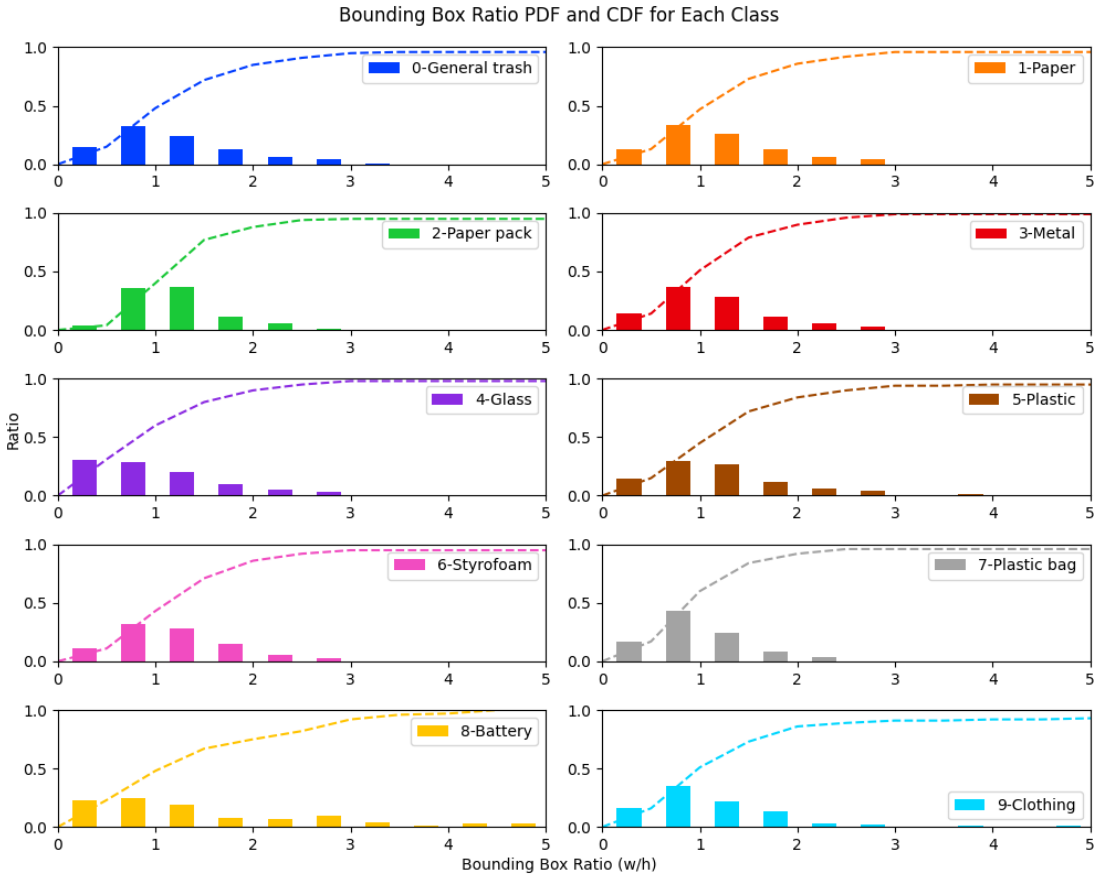


각 image에 대해서 Unique Class Label이 몇 개 존재하는지 확인하였다. 각 image당 1개의 Class Label이 있는 경우가 가장 많았으며 최대 7개의 Unique Class Label이 존재하였다.

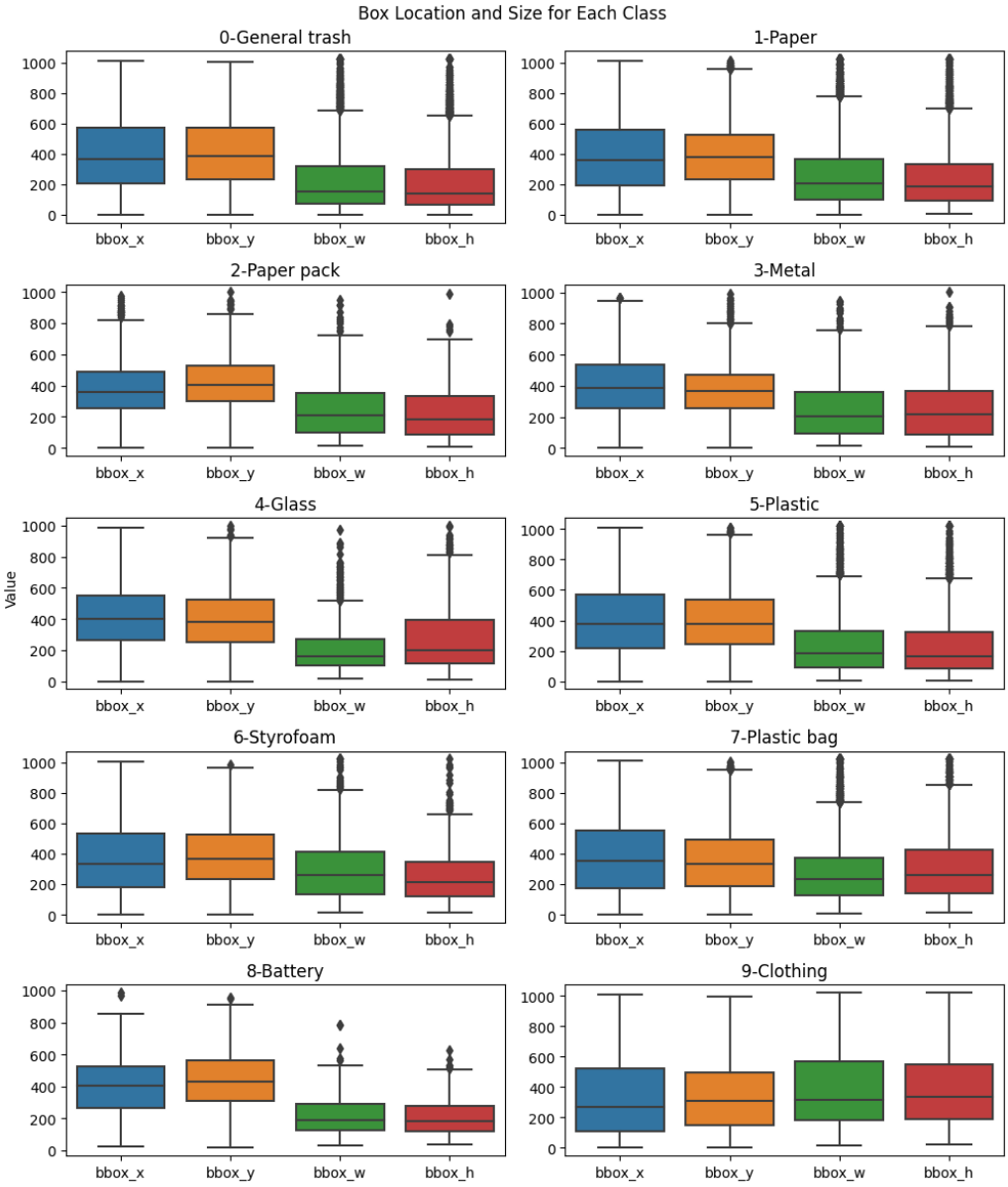
ii. EDA for Bbox



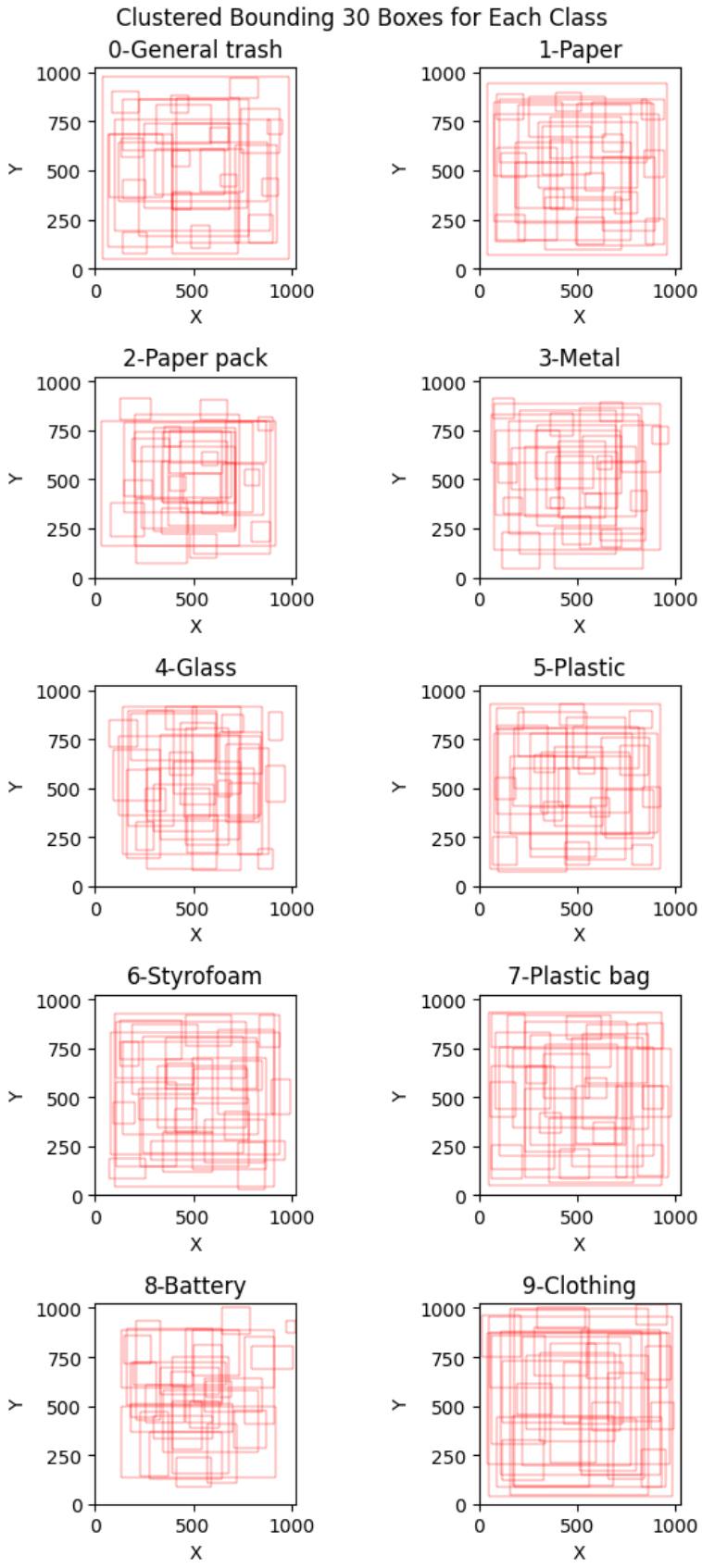
각 이미지에 annotation 되어있는 box의 개수를 확인하였다. 대부분의 이미지가 0~10개의 Bbox를 가지고 있음을 확인했으며 대략 40개 이상의 Bbox를 가지고 있는 소수의 이미지들에 대해서도 잘 동작할 수 있는 robust한 모델을 학습시키고자 하였다.



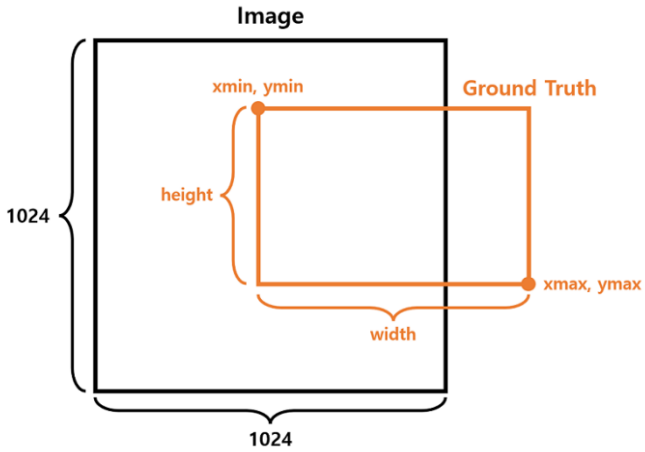
각각의 Class Label별 Bbox의 size가 세로가 긴 직사각형에 가까운지(ratio < 1), 정사각형에 가까운지(ratio ≒ 1), 가로가 긴 직사각형에 가까운지(ratio > 1)에 대한 Ratio 값을 PDF와 CDF Graph로 확인하였다. 대체적으로 정사각형에 가까운 Bbox가 많으며, 4-Glass와 8-Battery의 경우 직사각형에 가까운 비율이 많은 것을 확인하였다.



각각의 Class Label별 Bbox의 Location과 Size를 Graph로 확인하였다. 대부분의 xmin & ymin 좌표는 200 ~ 600이며, 9-Clothing의 경우 xmin & ymin 좌표가 200 이하인 Bbox가 다수 존재하였다. width & height에 대하여 대체적으로 100 ~ 500이며, 8-Battery의 경우 100 ~ 300 크기의 작은 Bbox가 다수 존재하였다.



각각의 Class Label별 K-Means Clsutering을 활용(K=30)하여 대표적인 Bbox의 위치와 크기에 대한 Plot 수행하였다. 대부분의 Bbox가 Image의 중앙 부근에 존재하며, size가 큰 Bbox가 다수 존재함을 확인하였다.



Ground Truth의 Bbox에 대해 x_min/y_min/x_max/y_max 및 width, height가 전체 Image size인 (1024,1024) 내에 있는지 확인해보았으나, 없는 것으로 확인하였다.

B. Baseline Test

i. Data Augmentation

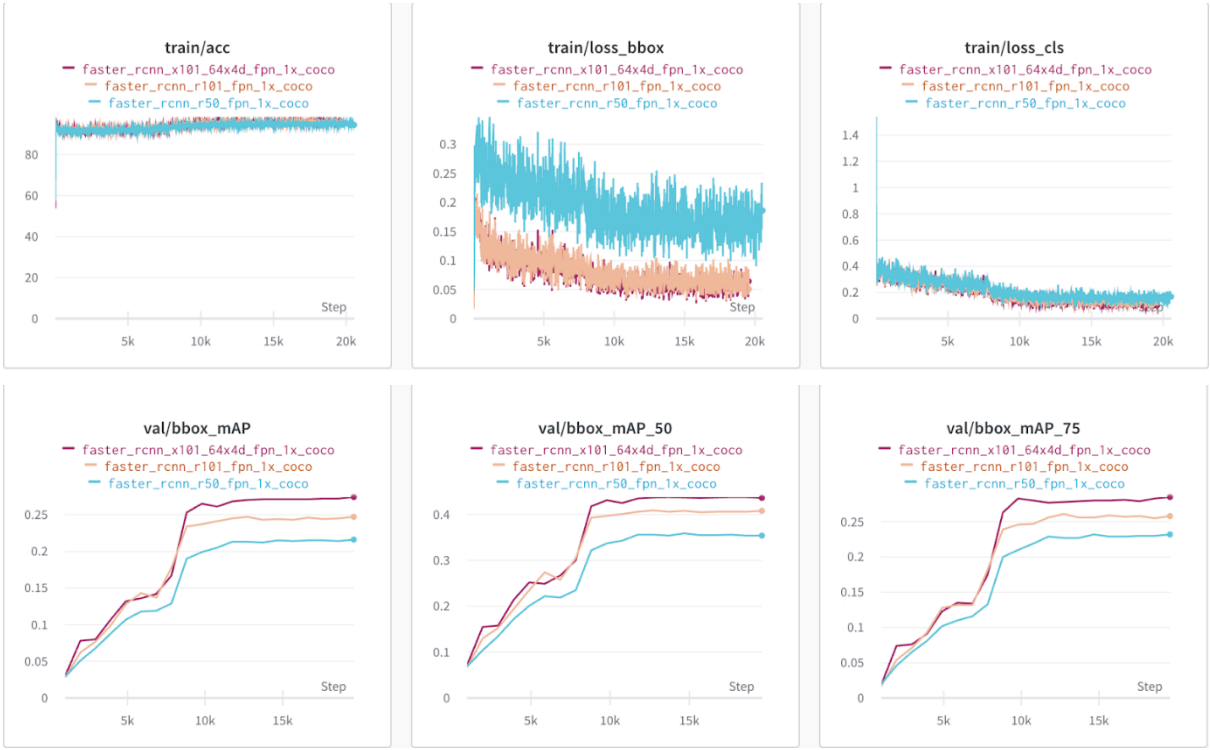
MMDetection Library의 경우 MMDetection 디렉토리 내 coco_detection.py의 Resize, RandomFlip을 주로 사용하고 TTA로는 resize만을 사용했다. 특히 Data EDA 결과 물체가 겹쳐 있는 경우가 많음을 관찰하였고, Mosaic Augmentation이 효과적일 것이라고 생각하여 적용해 보았다. 그러나 데이터의 형식 문제인지 쉽게 적용이 되지 않아서 결국 포기하였다. 또한, 색상 정보에 치우친 Detection을 할 수 있을 것이라고 예상하여 YOLOXHSVRandomAug 적용해 보았는데, 함께 쓴 다른 Augmentation의 영향 때문인지 결과값에서의 큰 차이를 얻지 못하였다.

YOLO v8 Library의 경우 yolov8/ultralytics 디렉토리 내 augment.py 파일 내에 Mosaic를 포함하여 Albumentation package와 torchvision의 Transform package를 활용하여 기본적인 Augmentation 기법들이 구현되어 있었다. 따라서 Object Detection task라는 특성과 본 competition에서 활용하는 dataset의 특성을 바탕으로 Rotate/Resize/Flip/HSV/MixUp/Mosaic을 적절히 조합하여 사용하였으며, 전체 Train dataset에 대해 default.yaml 파일의 Hyper-parameter값으로 Augmentation을 적용할 확률값을 setting하여 Data Augmentation을 수행하였다.

ii. Model

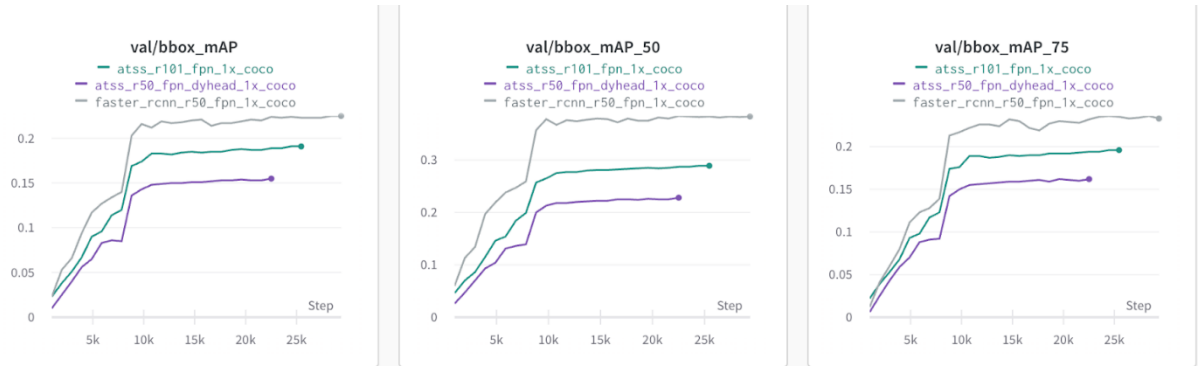
- Faster R-CNN

  ResNet-50, ResNet-101, ResNeXt-101-64x4d 세 가지 Backbone에 대한 Faster R-CNN Baseline Model 실험을 수행하였다. ResNet-50, ResNet-101, ResNeXt-101-64x4d 순으로, 모델 Backbone의 학습 파라미터가 많아질수록 높은 mAP 수치를 보였다. segmentation gt를 필요로 하는 Mask branch를 없애고 RoI Align을 사용하는 Mask R-CNN과 비교하였으나 오히려 Faster R-CNN이 더 높은 mAP를 보였다.



- ATSS

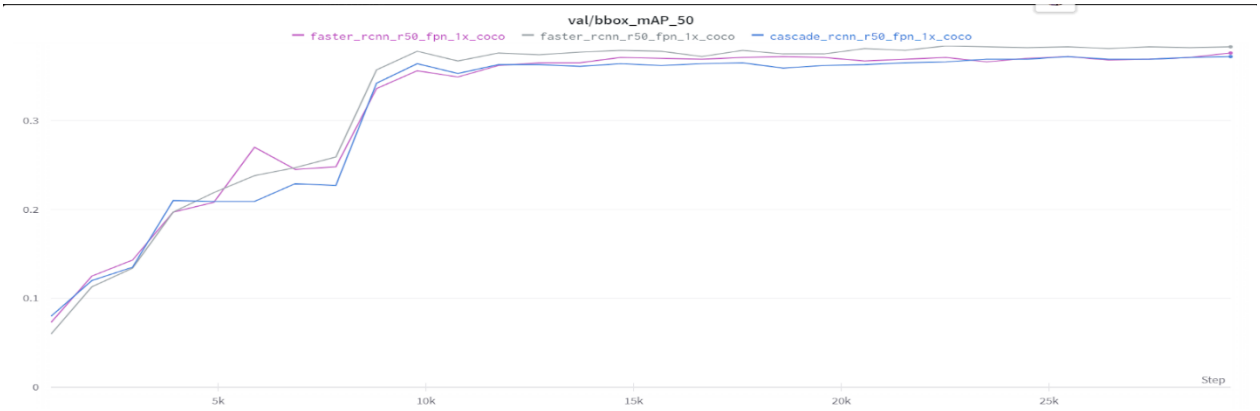
  클래스 분포 및 Bbox 크기 불균형을 해결하기 위한 ATSS(Adaptive Training Sample Selection)을 적용한 모델을 실험하였다.



결과적으로 Valid mAP log상으로 Faster R-CNN 보다 성능이 낮아 사용하지 않았다.

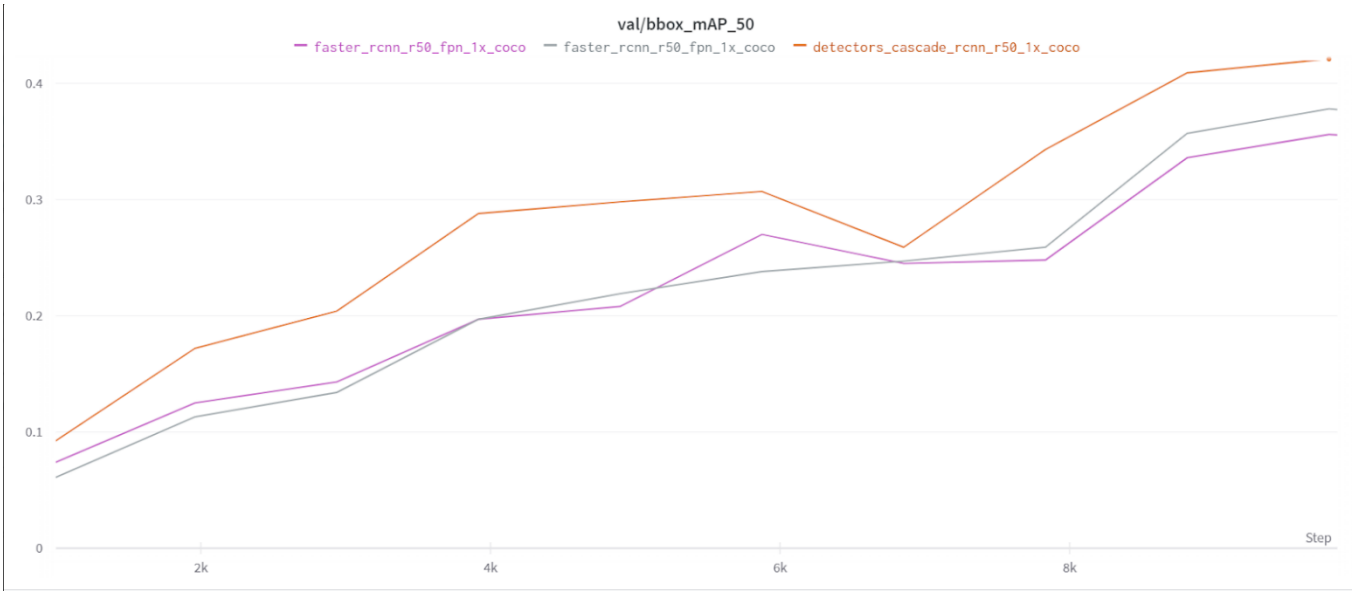
- Cascade R-CNN

  Data EDA 결과 Bbox크기가 다양하며 서로 겹쳐있는 경우가 많음을 확인하였다. 이에 다단계 분류기를 이용하여 여러 IoU Threshold로 학습하는 Cascade R-CNN이 효과적일 것으로 가설을 세우고 실험하였다. 가설과 달리, Faster R-CNN보다 낮은 정확도를 보였다.



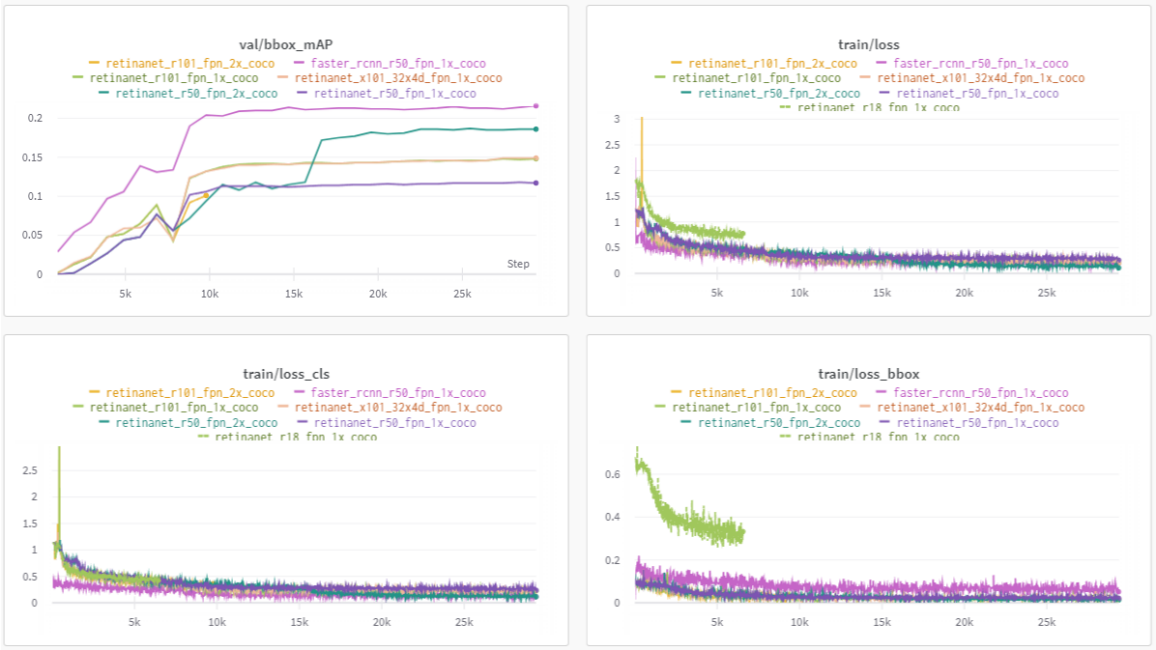
- DetectoRS

  MMDetection 라이브러리의 여러 모델 중 대회 데이터셋과 비슷한 COCO Dataset으로 학습되어 있는 모델이 특히 더 높은 정확도를 보일 것으로 예상하였다. 따라서 Bbox AP가 3번째로 높은 DetectoRS를 pre-trained모델로 하여 실험하였다. 그 결과 팀내 최고 성능 모델인 Faster R-CNN보다 월등히 좋은 성능을 보임을 확인하였다.



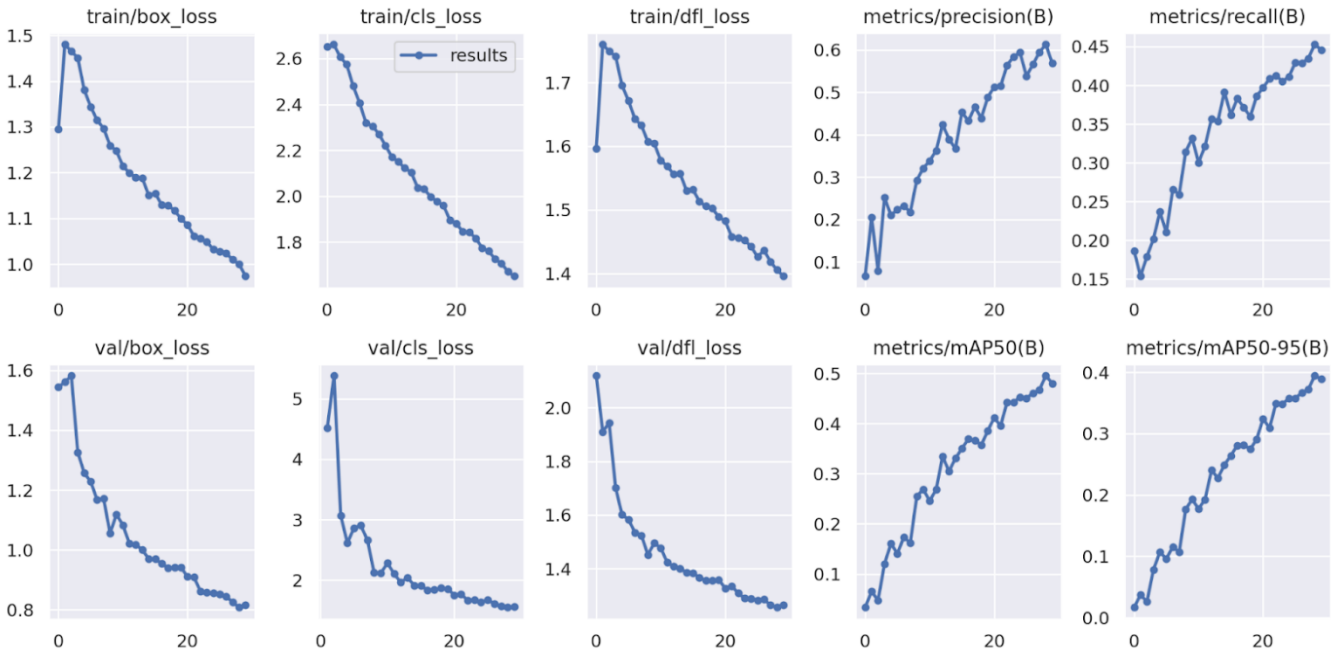
- RetinaNet

  Data EDA 결과 특정 class의 개수가 적은 클래스 불균형 문제를 확인하였다. 또 Faster R-CNN에서 사용하는 RPN이 작은 객체에 대한 정확도가 떨어질 수 있다는 점을 개선하고자 하였다. 따라서 Focal Loss를 활용하여 클래스 불균형을 해결하고, FPN을 활용하여 작은 객체에서도 높은 정확도를 유지할 수 있는 RetinaNet을 사용한다면 성능이 개선될 것이라 판단하여 실험을 수행하였다. 학습 결과 backbone의 학습 파라미터가 많을수록, 스케줄러가 변할수록(2x) 성능이 좋아짐을 확인하였다. bbox_mAP는 Faster R-CNN의 성능이 더 좋은 반면, 다른 지표인 train_loss, loss_cls, loss_bbox에서 모두 RetinaNet이 성능을 앞서는 것을 볼 수 있었다.



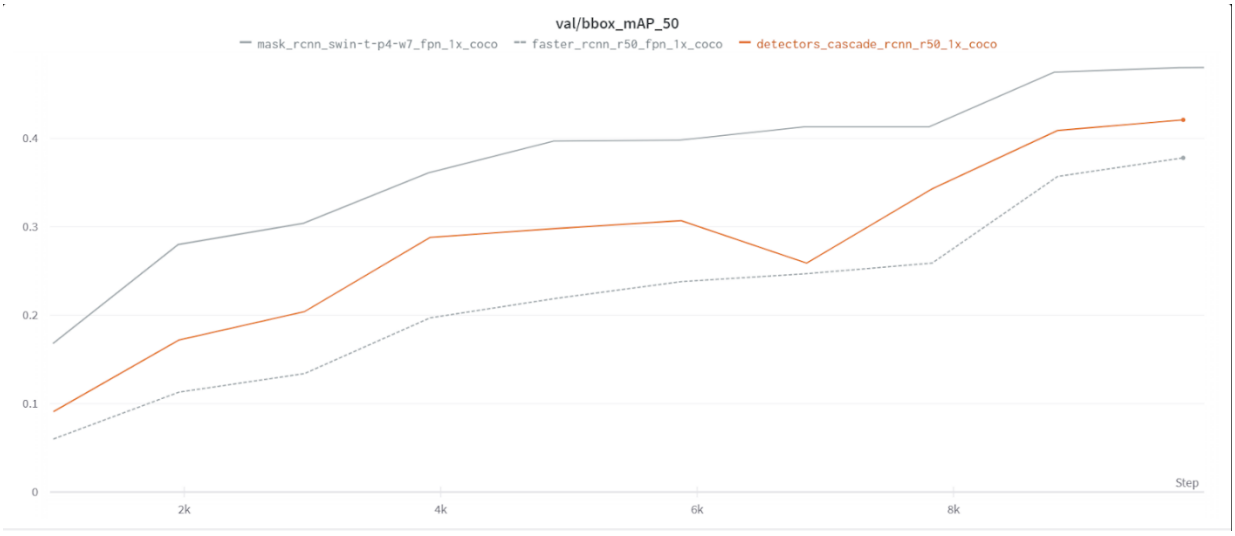
- YOLO v8

  가장 대표적인 1-Stage Detector로 알려진 YOLO 계열 모델의 가장 최신 version으로써, C2f Module 및 Anchor-Free algorithm을 사용하여 NMS의 속도를 높인 SOTA Network이다. YOLO v8의 경우 MMDetection과 별개로 Open Source를 import하여 사용했기 때문에, COCO format의 Ground Truth를 YOLO format으로 convert하는 coco2yolo.py와 SKF를 구현한 skf_dataset_split.py를 별도로 구현하였다. 더불어 학습 과정에서 필요한 Hyper-parameter를 정의하는 default.yaml 파일을 5개의 fold를 학습할 동안 변화하여 추후 Ensemble을 수행 시 모델의 일반화 성능을 향상시키고자 하였다. 아래는 5개의 fold로 분할된 Train, Valid set 중 가장 advanced한 성능을 보인 5번째 fold에 대한 학습 결과이다.



- Mask R-CNN 기반 Swin Transformer

  데이터셋 중에 한 이미지에 여러 Bbox가 있는 데이터가 많음을 확인하였고, 패치 단위로 분할하여 처리하는 트랜스포머 계열이 성능향상에 도움이 될 것이라고 예상했다. 그중, 다양한 크기의 객체를 검출하는데에 효과가 있으며, SOTA모델인 Mask_R-CNN_swin_t를 사용하였고, Faster R-CNN과 Detector RS보다 성능이 좋음을 확인하였다.



C. Stratified K-Fold

- 사용 이유

  Data EDA를 진행한 결과 Train data의 Class Imbalance를 확인하였다. 따라서 편향되어있지 않은 Train, Valid set을 구성 하기 위해 Train, Valid set을 기존 Total Train set과 같은 비율로 구성하는 Stratified K-Fold Cross Validation을 사용하였다. data_split.py 를 통해 전체 Train data의 Annotation을 5개의 Train, Valid set으로 split하였다.

	General trash	Paper	Paper pack	Metal	Glass	Plastic	Styrofoam	Plastic bag	Battery	Clothing
training set	17.14%	27.45%	3.88%	4.04%	4.24%	12.72%	5.46%	22.37%	0.69%	2.02%
train - fold0	16.91%	27.64%	4.08%	4.13%	4.21%	12.56%	5.42%	22.58%	0.68%	1.80%
val - fold0	18.08%	26.64%	3.03%	3.69%	4.38%	13.37%	5.61%	21.53%	0.73%	2.94%
train - fold1	17.26%	27.04%	3.86%	4.04%	4.31%	12.70%	5.45%	22.52%	0.75%	2.07%
val - fold1	16.66%	29.04%	3.93%	4.08%	3.99%	12.79%	5.47%	21.79%	0.43%	1.82%
train - fold2	17.12%	27.48%	3.92%	3.87%	4.49%	12.70%	5.46%	22.05%	0.70%	2.20%
val - fold2	17.20%	27.30%	3.70%	4.72%	3.29%	12.77%	5.45%	23.60%	0.62%	1.35%
train - fold3	16.81%	27.70%	3.98%	4.17%	4.18%	12.69%	5.37%	22.42%	0.64%	2.05%
val - fold3	18.41%	26.44%	3.48%	3.56%	4.48%	12.83%	5.82%	22.19%	0.86%	1.93%
train - fold4	17.58%	27.36%	3.54%	4.02%	4.03%	12.94%	5.59%	22.29%	0.66%	2.00%
val - fold4	15.29%	27.80%	5.28%	4.14%	5.14%	11.80%	4.92%	22.71%	0.80%	2.11%

split의 개수를 너무 크게 한다면 학습시키는 시간이 많이 사용되고, split의 개수를 너무 작게 한다면 데이터셋 분할에 대한 다양성이 적어지고 각 fold에서 학습시키는 데이터의 개수가 적어져서 충분한 학습이 이루어지지 않을 수 있다. 따라서 가장 일반적인 split의 개수인 5개를 선택하였다.

- data split.py

  Train, Valid set을 기존 train set과 같은 비율로 구성하기 위해 Scikit-Learn 패키지 내의 model_selection에 구현되어 있는 StratifiedGroupKFold 함수를 사용하였다. StratifiedGroupKFold 함수를 통해 클래스 비율에 맞게 annotation id를 나누고 그 id에 따라 train.json에서 각 5개의 Train set과 Valid set으로 분할 하였다.

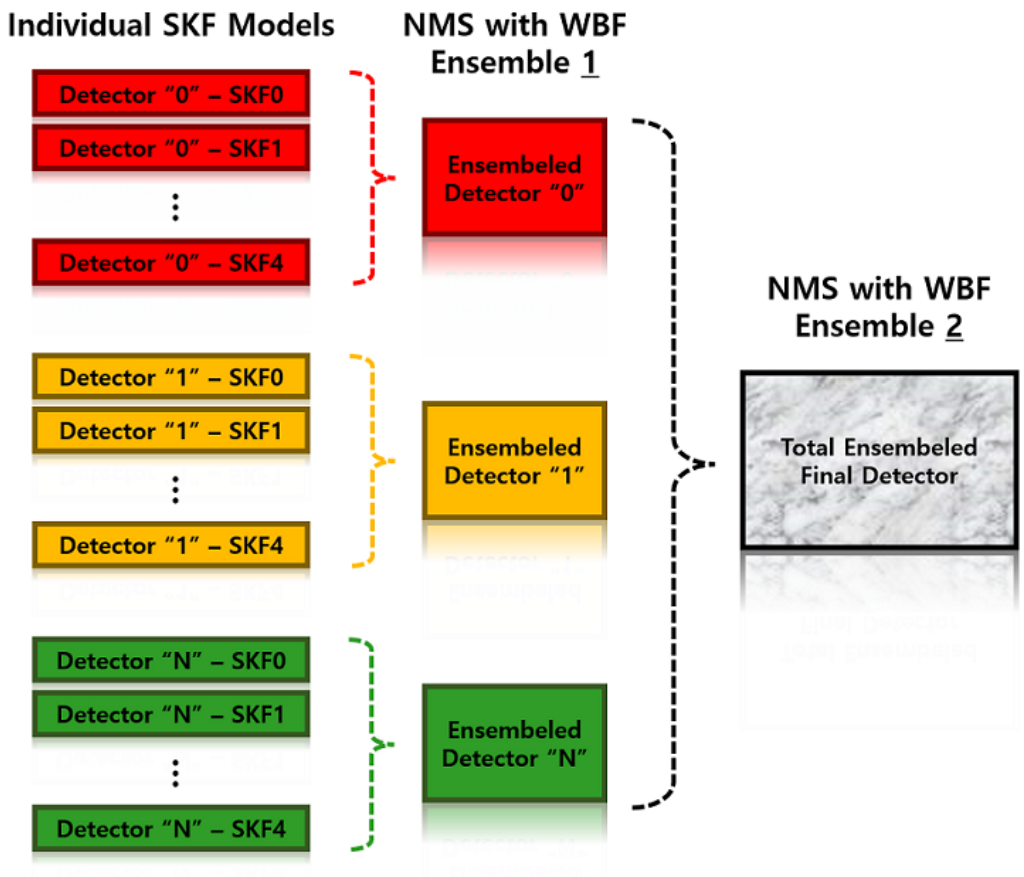
- 활용 전략

  총 5개의 fold를 각각 학습시켜 5개의 학습된 모델(.pth 파일)를 얻었다. 그 후 5개의 모델을 load하여 Inference 과정을 거치고 5개의 submission.csv을 생성하였다. 그 5개의 submission.csv를 WBF가 접목된 NMS 기법을 활용한 Ensemble을 통해 1개의 submission.csv를 생성했다.

D. Model Ensemble

i. Ensemble 조합 및 수행 결과

각 팀원들이 수행한 5개의 Stratified K-Fold에 대해 WBF algorithm이 접목된 NMS를 수행하여 1개의 Best Model을 추출하고, 5개의 Best Model을 다시 WBF 기반의 NMS를 수행하여 최종적으로 1개의 Best Model을 추출하고자 하였다. Inference를 위한 전반적인 Model Ensemble 수행 과정은 아래의 그림과 같으며, Ensemble을 위해 선택한 Best Model의 조합은 아래의 표와 같다.



Models	mAP
YOLO v8 + RetinaNet	0.4556
YOLO v8 + RetinaNet + Faster R-CNN + Cascade R-CNN	0.4585
YOLO v8 + RetinaNet + Faster_R-CNN + Cascade R-CNN (iou_thresh=0.5)	0.4663
Swin Transformer KFold(split 0~3) + Faster R-CNN + DetectoRS	0.5222
Swin Mask R-CNN + DetectoRS (iou_thresh=0.6)	0.5255
Swin Transformer (split 0,1,3) + Faster R-CNN + DetectoRS	0.5270

ii. Weighted Box Fusion(WBF)

모델 간 Ensemble을 수행하는 방식으로 WBF(Weighted Box Fusion)을 사용하였다. 개별 모델들이 예측한 Bbox에 대응하는 Ground Truth Box와의 IoU value를 기반으로 Weighted-Sum을 수행하여 예측 Bbox의 좌표를 미세 조정하는 Ensemble 기법을 사용하였다.

iii. Non-Maximum Suppression(NMS)

WBF algorithm을 기반으로 Non-Maximum Suppression을 수행하여 Model Ensemble을 수행하였다. 예측한 Bbox에 대해 Confidence score를 기반으로 1차적인 filtering을 수행한 이후에, IoU value의 Weighted-Sum을 기반으로 한 WBF를 수행함으로써 미세 조정된 예측 Bbox에 Ground Truth Bbox를 가장 잘 표현하는 단 1개의 예측 Bbox만을 남기고 나머지는 제외하는 NMS를 수행하였다. 특히 그 과정에서 IoU threshold값을 변화하며 1개의 object를 표현하는 filtering 이후의 Bbox 개수를 달리하여 mAP 성능을 측정 및 비교하였다.

iv. IoU threshold

Weighted Box Fusion 및 Non-Maximum Suppression에서 IoU threshold를 기준으로 겹치는 Bbox를 판단한다. IoU threshold를 0.4, 0.5, 0.6, 0.7으로 바꿔 실험을 진행하였고, 결과적으로 적절한 IoU threshold(0.5~0.6)를 사용하는 것이 mAP score가 더 높게 나왔다. 이에 대한 원인으로는, IoU threshold가 높을 수록 겹치는 것으로 판단되는 Bbox가 적고 이에 따라 더 많은 Bbox가 예측되기 때문에 높은 mAP를 보이는 경향이 있었다.

5. 자체 평가 의견

A. 잘한 점

- 지속적인 오프라인 미팅으로 대회에 몰입할 수 있었다.

- 각자 원하는 실험을 선택하고 돌릴 수 있었다.
- 대회 기간 내 싸우지 않고, 적극적으로 의사소통하였다.
- 팀원들끼리 적극적인 토의와 질문을 통해 어려움을 해결해 나갔다.
- 좋은 성능의 AI 모델을 설계하기 위해 각자 공부해 온 인사이트를 공유하며 협업했다.

B. 시도했으나 잘 되지 않았던 것

- 최신 SOTA 모델들을 가져와서 우리 Baseline Code에 적용하기
- UniverseNet을 적용하려 하였으나 Inference과정에서의 오류를 해결하지 못해서 실패
- K-Fold를 적용하면서 Train/Valid log monitoring(wandb) 하기
- wandb hook을 보다 심도있게 이해하여 customize하기
- YOLO v8 설계 시 train.py에서 SKF를 접목하여 학습시키는 코드를 작성하지 못한 점
- Object Detection에 최적화된 다양한 데이터 증강 기법 활용

C. 아쉬웠던 것

- Augmentation, Hyper-parameter tuning에 시간을 투자하지 못했다.
- Baseline Model test시 epoch을 크게 두어, 시간을 너무 많이 소비했다.
- Baseline Model test시 Cross Validation(kfold)을 적용하며 시간을 너무 많이 소비했다.
- 너무 기본적인 구식 모델에 시간을 많이 쓰고, 큰 모델을 뒤늦게 사용했다.
- PapersWithCode와 같은 인사이트를 활용해 더욱 최신식의 SOTA 모델을 사용해보지 못한 것이 아쉬웠다.
- 학습 후 EDA를 진행해보지 못했다.
- MMDetection과 YOLO v8 모두 Open Source Library를 활용하여 기본적으로 주어지는 Dataset 관련 파일들을 본 task에 알맞게 customizing해야 하는데, Data EDA의 결과를 바탕으로 파악한 특징들에 최적화된 customized Dataset과 Augmentation을 구현하기에 기존 코드가 매우 복잡하여 시도하지 못했던 것이 아쉬웠다.
- 만들어진 Architecture가 아닌, 주어진 dataset에 최적화된 Backbone, Neck, Head 등을 설계해보지 못했던 점이 아쉬웠다. 기회가 된다면 Open Source Library를 활용하는 것 보다 scratch로 구현한 Model을 학습/추론하는 방법으로 프로젝트를 수행해보고 싶다.

D. 프로젝트를 배운 점 또는 느낀 점

- Augmentation, Hyper-parameter, Loss function, Model Ensemble 등으로 얻을 수 있는 성능 향

상은 모델이 가진 한계를 넘지 못한다. 결국 가장 높은 성능 범위를 보장하는 Detection 모델을 찾는것이 중요하다. 같은 이유로, 현재 성능이 목표치에 크게 뒤떨어진다면 다른 모델을 사용하는 것을 고려해야한다.

- baseline model test시 log를 보며 충분히 학습될 수 있는 적당한 epoch을 찾아서 실험을 진행해야 시간을 아낄 수 있다.
- Object Detection task를 수행하기 위한 Model Architecture 및 Ensemble에 대한 이론, Data Annotation Format과 같은 데이터 관점에서의 이론에 대해 알 수 있었고, MMDetction을 포함해 다양한 Open Source Library를 활용해 Customizing하는 방법에 대해 자세히 탐구할 수 있었다.

Lv.2 Competition Personal Wrap-up Report

김보경_T5033(CV-10 형셋누나하나)

학습 목표 및 달성하기 위한 노력

- 저희 팀은 private test dataset에 대해서 mAP를 높이는 것과 mmdetection 활용법을 익히기 위한 다양한 시도를 해보는 것을 목표로 하였습니다. 이를 달성하기 위해 저는 크게 아래 시도들을 시도해 봤습니다.

  1. baseline code 리팩토링

     baseline 모델을 실험하기에 앞서 파이썬 스크립트를 만드는 것이 디버깅, 백그라운드 실행, 코드 가독성, 코드 공유 등에서의 이점이 있다고 생각되어 train.py, inference.py, ensemble.py를 만들었습니다. 스크립트 작성 시 실험 시 사용되는 파라미터를 고려하여 작성한 덕분에 팀원들이 모두 하나의 파일을 가지고 학습을 할 수 있었습니다.

  2. wandb 연동

     wandb 연동 코드를 추가하여 실험 공유에 도움이 되고자 하였습니다. wandb 덕분에 실험 공유 및 실시간 학습상황을 볼 수 있었고, 업무의 효율성을 높일 수 있었습니다. 다만 wandb 연동을 위해 mmdetction의 hook을 활용하였는데, 이를 제대로 이해하지 못하고 사용하여 다른 프로젝트에 저장이 된다거나, 1epoch마다 저장이 되지 않아 그래프가 지저분해지는 등의 문제점을 겪었습니다.

  3. RetinaNet 실험

     Baseline model을 실험할 때 RetinaNet을 맡아서 실험하였습니다. Backbone과 Lr schd에 따른 성능 비교를 하였습니다.

Limitations

- 앞선 Image Classification 과제에서 제공된 문서와 코드를 제대로 보지 않고 실험한 부분이 개선할 점이라고 생각하였습니다. 따라서 이번 대회에서는 제공된 코드, 강의 등을 모두 제대로 보겠다는 마음이 컸고 그 결과 정작 실험할 시간이 부족해졌습니다.
- 적극적으로 성능 개선에 임하지 못했습니다. wandb hook를 customize 못한 점, config파일을 customize 못하고 기존에 제공되는 파일의 일부만 수정하는 등 적극적으로 실험에 임하지 못했습니다.

FeedBack

- 앞선 대회와 이번 대회를 통해 모든 일은 중요하다는 것을 깨달았습니다. 따라서 시작하기에 앞서 업무의 종류 및 중요도를 판단하여 적절한 시간 배분을 해야겠다고 다시 한번 느꼈습니다.
- Mmdetection 사용법을 몰라 실험에 소극적이였던 것 같습니다. 이번 대회를 통해 mmdetection 사용법을 익힐 수 있었기 때문에 다음 대회에서는 다양한 개선 방안을 고려하고 적극적으로 개선 시도를 해야겠다는 생각이 들었습니다.

Lv.2 Competition Personal Wrap-up Report

김정주_T5052(CV-10 형셋누나하나)

Project Direction Setting

- Data EDA로 파악한 Class Imbalance 문제를 해결하기 위해 **적절한 Augmentation & Loss function 선택**
- AI Detector로 YOLO v8을 채택하고 **COCO format을 YOLO format으로 변환하기 위한 코드 구현**
- **K=5로 setting한 Stratified K-Fold Cross Validation**을 구현하기 위한 코드 구현
- 각각의 fold에 대해 **Hyper-parameter를 달리 하여 Training 수행**
- 각각의 fold로부터 추출된 Best Model에 대해 **WBF가 접목된 NMS를 수행하여 Model Ensemble 구현**

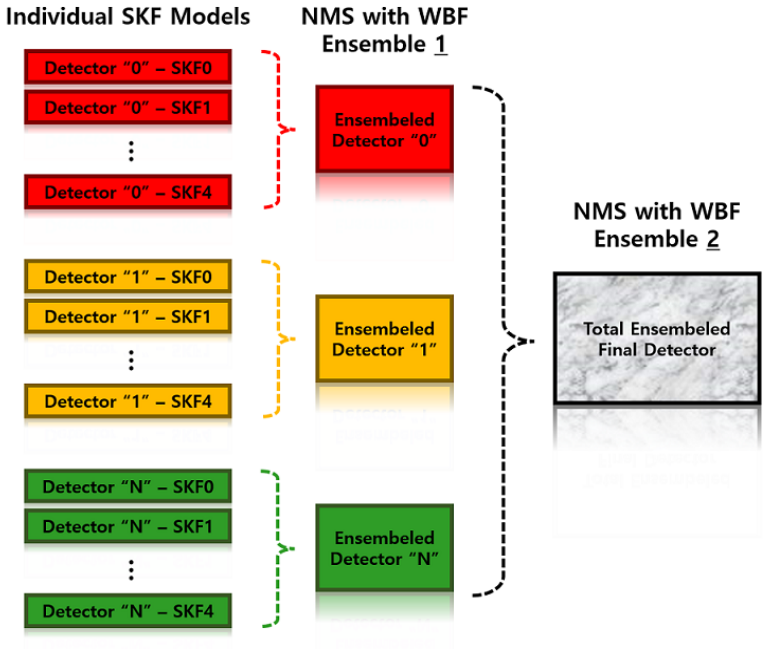
Trials

- 대표적인 1-Stage Detector인 YOLO 계열의 가장 최신 version인 **YOLO v8의 Open Source Library**를 활용하여 AI Detector 구축. 좋은 성능을 기록하기 위해 **YOLO v8 Large Scale**을 선택
- coco2yolo.py 코드를 구현하여 주어진 COCO format의 Annotation(xmin, ymin, width, height)을 **YOLO format의 Annotation(xcenter, ycenter, width, height)으로 convert하는 매커니즘**을 구현
- Ultralytics에서 제공한 기본적인 Augmentation 기법을 활용해 **Resize/Rotate/Flip/HSV/MixUp/Mosaic**을 적용해 모델의 일반화 성능 향상을 고취
- K=5로 setting한 **Stratified K-Fold Cross Validation**을 skf_dataset_split.py 코드로 구현하여 Object Detection task가 갖는 **고질적인 Class Imbalance 문제를 해결하고자 노력**함
- 최종적으로 생성된 5개의 Train/Valid dataset 쌍에 대해 **default.yaml 파일을 통해 Hyper-parameter를 달리 하여 5번의 Training을 수행**하고, 결과적으로 5개의 Best Model을 추출함. 모든 Trial들은 **기본적으로 Label Smoothing factor α=0.1로 setting**함
- **팀원들이 효과적으로 모델 Ensemble을 수행할 수 있도록 WBF가 접목된 NMS algorithm을 구현함**

① **Stratified K-Fold**로 분할된 각각의 5-pair dataset에 대해 Training을 수행한 **5개의 Best Model 추출**

② **5개의 Best Model**에 대해 **사전에 정의한 IoU threshold**를 활용하여 **WBF가 접목된 NMS** 수행

③ **팀원들 각각이 추출**한 1개의 Best Model들에 대해 **동일한 매커니즘으로 Ensemble 수행**



- 프로젝트를 진행하며 팀원들 각자가 수행한 내용과 그 이유 및 장단점에 대해 **log를 남길 수 있는 별도의 Notion page를 구축해 효율적인 협업**을 위해 노력함

  https://flower-bobolink-12a.notion.site/Trash-Recycling-Object-Detection-97bb787cb5cb429c80cf28f9fc64f4ce

FeedBack

- *Open Source Library를 활용하는 것이 아닌, scratch부터 직접 구현한 AI Detector를 실험해보지 못한 점이 아쉬움. 추후 기회가 생긴다면 팀원들과 함께 SOTA 모델을 scratch부터 구현하여 학습/추론해 좋은 mAP 성능을 보이는 모델을 구현하고자 함*
- *YOLO v8 이외에 다양한 SOTA 모델을 시도하지 못했던 것에 대해 아쉬움. 특히 강의에서 추천한 MMDetection과 Detectron2 Library를 활용해 competition을 수행하지 못한 것에 대한 아쉬움이 큼. 추후에 개인 프로젝트로 MMDetection 혹은 Detectron2 Library를 활용해 Object Detection task를 수행해보고자 함*
- *YOLO v8 train 과정에서 wandb를 사용해 training log를 남기도록 구현하지 못한 것에 대해 아쉬움. 팀원들 모두 wandb를 활용하여 Training log를 별도의 directory를 생성해 기록했는데, Ultralytics에서 제공한 YOLO v8과 같은 Open Source Library는 기존에 구현되어 있는 매우 복잡하여 customizing 하기가 어려움을 느낌. 직접 scratch부터 구현하는 것도 좋지만 다른 엔지니어가 구현한 SOTA Library를 정확하고 자세하게 분석하여 customizing하는 방법을 학습하는 것의 중요성을 실감하게 됨. 추후에 이어질 프로젝트에서는 꼭 wandb를 사용하여 Training log를 남겨보고자 다짐함*
- *성능이 좋을 것으로 기대했던 YOLO v8이 예상 외로 성능이 좋지 않아 어려움을 겪음. 잘 알려진 Network나 이전에 유명했던 SOTA Network보다는 PapersWithCode와 같은 인사이트를 적극 활용해 최신 SOTA Network를 활용하여 AI Detector를 설계해보지 못한 것에 대한 아쉬움을 느낌*

Lv.2 Competition Personal Wrap-up Report

양재민_T5123(CV-10 형셋누나하나)

Project Direction Setting

- AI 프로젝트에 대한 경험이 부족하고 object detection에 대한 이해가 부족했기 때문에 이번 프로젝트를 통해서 프로젝트의 흐름을 잘 따라가며 협업에 익숙해지는 것과 강의수강과 추가 학습을 통해서 object detection의 개념을 이해하고 실험을 통해 이해한 내용을 적용 및 활용하는 것을 목표로 했습니다.

Trials for Myself

- 위의 목표를 달성하기 위해서 가장 중요한 것은 강의 수강이라고 생각해서 강의를 매우 꼼꼼하게 듣고 추가 학습을 한 후 개인 노션에 해당 내용을 정리하며 복습했습니다. 그 후 원활한 실험을 위해 baseline code의 흐름을 이해하려고 노력했습니다. 실험을 할 때는 기계적으로 하는 것이 아니라 알고 있는 이론이 어떤식으로 반영되는지 이해하려고 했습니다. 원활한 협업을 위해서는 팀원들이 어떤 작업을 하고 있는지, 내가 현재 해야할 작업은 무엇인지를 확실히 인지하고 맡은 작업을 확실히 수행하려고 했습니다. 또한 스스로 해결할 수 없다고 판단되면 팀원들에게 조언 및 도움을 구해서 해결했습니다.

Problem-solving Process and Results

- 팀에서 baseline test와 Stratified K-fold의 train.py, inference.py 구현, K-fold ensemble 구현을 맡았습니다. baseline test에서는 faster RCNN과 mask RCNN을 맡아서 test를 했습니다. faster RCNN 실험은 완료했으나 mask RCNN test시 mask branch를 제외하고 train을 시켜야하는 문제에 직면하였습니다. 혼자 해결하려 했으나 해결되지 않아서 팀원과 함께 디버깅을 하며 해결했습니다.

- Stratified K-fold train을 구현할때 한번의 train 함수 실행으로 5개의 fold의 train이 되게 하려고 했습니다. 기존에 팀원이 만들어 주었던 train.py 를 리팩토링해서 Stratified K-fold를 적용한 train를 구현했습니다. 구현하는 과정에서 팀원이 이미 data를 split 해주었지만 그것을 확인하지 못하고 꽤 많은 시간을 사용하여 같은 걸 구현했습니다. 이 후로 팀원이 어떤 작업을 완료했고 진행중인지 인지하는 것에 더욱 신경을 쓰게 되었습니다.

- Stratified K-fold inference을 구현할 때 train과 마찬가지로 한번의 inference 함수 실행으로 5개의 fold의 inference를 수행시키기 위해서 기존에 팀원이 만들어 주었던 inference.py 를 리팩토링해서 Stratified K-fold를 구현했습니다. 또한 5번의 inference를 통해 나온 5개의 submission.csv를 1개의 submission.csv 파일로 ensemble 할 수 있는 코드를 구현했습니다.

Limitations

- Stratified K-fold를 구현하는 것에 많은 시간을 사용했던 것 같아서 아쉽습니다. 또한 Stratified K-fold를 적용한 train 과정에서 wandb연동을 어떤식으로 해야하는지 확실히 고민을 해보고 넘어갔어야 했는데 일단 팀원들에게 배포를 해야겠다는 생각에 해당 부분을 대충 넘어갔고 그로인해서 Stratified K-fold를 적용한 실험에서 원활한 log 추적을 못했던 것이 아쉽습니다.

- baseline test 할 때 기본이 되는 모델부터 실험해보자는 생각으로 실험을 시작했는데 mmdetection에서 실험하는 환경에 익숙하지 않아서 실험이 늦어졌고 그로인해 더 무겁고 최신의 모델을 많이 적용 못해본

것이 아쉽습니다. 모델 선정이 늦어진 결과 다양한 data augmentation을 적용해보지 못했고 hyperparameter tuning도 다양하게 해보지 못해서 아쉽습니다.

- 디버깅하는 것에 익숙하지 않아서 혼자 에러를 해결한 것보다 팀원의 도움을 받아서 해결한 것이 더 많은 것 같습니다. 팀원의 도움을 받는 것이 나쁜 것은 아니지만 혼자 해결할 수 있는 힘을 기르는 것도 중요하다는 걸 느꼈습니다.

Future Direction

- 다음 프로젝트에서는 이론 공부를 조금 더 서둘러 해서 실험하는 시간을 확보해서 좋은 성능을 낼 수 있게 해보고 싶습니다. 또한 이번 프로젝트보다 더 엄밀하게 이론을 바탕으로 가설을 세워보고 그걸 증명해 나갈 수 있는 실험을 진행해보고 싶습니다. 프로젝트 중에 더 다양한 기능들을 맡아서 구현해보면서 python 구현능력을 키우고 팀에 도움이 되겠습니다.

Lv.2 Competition Personal Wrap-up Report 임준표_T5175(CV-10 형셋누나하나)

학습 목표

- 지난 대회 때 모델 선정을 위해 여러 모델을 돌려보았으므로 모델 선정이외의 부분을 경험해 본다.
- mmdetection의 config 파일과 실행 원리에 대해 이해한다.

나의 역할

- cascade R CNN 을 여러 조건에서 실험해보기
- 그외 모델 중 성능이 좋은 모델 찾기

모델 개선 방법

- cascade RCNN은 세 가지 분류기를 갖고 있으므로 roi_head를 각각 설정 해야 한다.
- debugging을 이용하여 오류를 분석한다.

프로젝트 회고

- **노력한점**
  1. 데이터 증강 기법 중, Mosaic, randomflip, YOLOXHSVRandomAug 등을 구현시도하였다.
  2. 후처리 방법인 hard voting에 대해 탐구해보았다.
  3. cascade RCNN, Detector RS, swin transformer에 대해 이해하고, 실험했다.
- **아쉬운점**
  1. 코드 구현 능력 문제로 Mosaic와 hard voting, soft voting, under sampling, over sampling에 대해서 구현 실패하였다.
  2. 시간 상의 문제로 neck, backbone, head를 주어진 데이터셋에 맞게 설정해보지 못했다.
  3. 학습 후 결과를 이용하여 EDA를 진행해보지 못했다.
  4. 더 다양한 모델을 실험해보지 못하고 학습시간이 오래 걸려 3가지 모델만 학습했다.
  5. 배치사이즈, 에포크, optimizer, loss function을 조절하여 학습해보지 못했다.

[총평]

강의를 미리 듣고 대회에 참여하는 것을 목표로 해서 대회에 참여하는 시간이 길지 않았다. 그러나, 팀원의 도움으로 config 파일 이해와 mmdetection의 작동 방식 이해를 할 수 있었다. 또한, cascade RCNN과 cascade RCNN기반 모델인 DetectorRS를 사용해보았고, mask rcnn도 사용해보는 좋은 기회가 되었다. 실험 외적으로는 debugging방법과 nohup사용법, 학습 진행 시 wandb기록, 제출 모델 기록의 중요성을 배웠다. 시간적 제약 때문에 data augmentation과 voting을 구현하는데에 실패했지만, 그 부분을 제외하고도 의미 있던 시간이었다. 기회가 주어진다면, neck, backbone, head, optimizer, batch size, loss function, 후처리 기법 등 더 시도해 볼 수 있었으면 좋겠다.

Lv.2 Competition Personal Wrap-up Report

정남교_T5188(CV-10 형셋누나하나)

Project Direction Setting

- 우리팀은 mmdetection 라이브러리를 활용하여 프로젝트 진행을 하기로 하였다. 우선 대회 시작하기에 앞서 팀적인 진행방식에 대해 회의를 하였다. EDA 를 다 같이 수행하며 데이터를 뜯어보고, 이후 model 별로 분담하여 baseline model 실험을 진행하기로 하였다.

Trials for Myself

- mmdetection 코드 상으로 validation 을 수행하기 위해서는 오프라인으로 train/val data split 을 수행해야하는 것으로 판단하여, Stratified k fold 방식에 따라 클래스 별 균형을 맞춘 k 개의 train/val fold 로 나눠주는 코드를 구현하였다.
- Faster RCNN 의 실험을 진행하였고, backbone 을 변화시키며 이에 따른 성능차이를 확인하였다.
- 클래스, bbox 불균형이 있는 데이터 특성을 고려하여 ATSS 기법을 적용한 비교적 최신 모델을 실험하였다.

노력한 점

- 가장 보편적으로 object detection 에 사용되는 오픈소스 라이브러리인 mmdetection 을 통해 진행함으로 그 사용법을 익히고자 하였다.
- 개별화된 역할 분담보다는 회의, 공유 등을 통해 전반적인 진행 과정을 함께하여 모든 팀원이 수행 내용 전체에 대해 이해할 수 있었다.
- 각자의 수정사항으로 인해 코드가 개별화가 되지않도록 깃을 활용하여 통합적인 하나의 코드 관리를 하도록 노력하였다.
- 주어진 baseline 코드를 그대로 활용하기 보다는 스크립트 형식의 py 파일로 우리의 필요에 맞게 커스텀한 코드를 구축하여 사용하였다.

아쉬운 점

- 강의 및 실습 내용을 모두 수행한뒤 대회를 시작하였기 때문에 거의 대회 기간의 절반 정도만 온전히 투자할 수 있었다. 강의는 부지런히 들으면서 동시에 빠르게 대회를 시작했으면 어땠을까 하는 아쉬움이 남는다.
- base model 실험을 빠르게 완료하지 못했다. 이 단계에서 cross validation 을 수행하며 시간이 너무 길어졌고, 결정적으로 좋은 성능을 보장받기 위해서는 최신 sota 모델을 활용하는 것이 좋았겠지만 강의에서 다룬 구식 모델들에 너무 집중하였다.
- 깃을 활용한 협업을 하고자 노력하였으나 팀원들 모두 깃 사용이 미숙한 점이 있어서 제대로 하지 못한점이 아쉽다.
- ATSS 를 사용한 실험 성능이 낮게 나와서 실망 했었는데, 나중에 다른 팀 얘기를 들어보니 성능이 좋게 나왔다고 한다. 내가 실험을 하는 과정에서 무언가를 잘못해서 성능이 안나왔을수도 있을 것 같아 아쉽다.