Object Detection

프로젝트 개요

바야흐로 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.

따라서 우리는 사진에서 쓰레기를 Detection 하는 모델을 만들어 이러한 문제점을 해결해보고자 합니다. 문제 해결을 위해 일반 쓰레기, 플라스틱, 종이, 유리 등 10 종류의 쓰레기가 찍힌 사진 데이터 셋을 사용했습니다.

- **dataset**
  - train: 4883장의 train image
  - test: 4871장의 test image
  - 10 class : General trash, Paper, Paper pack, Metal, Glass, Plastic, Styrofoam, Plastic bag, Battery, Clothing
  - 이미지 크기 : (1024, 1024)
- **annotation file**
  - images:
    - id: 파일 안에서 image 고유 id
    - height: 1024
    - width: 1024
    - file*name: ex) train*/002.jpg
  - annotations:
    - id: 파일 안에 annotation 고유 id
    - bbox: 객체가 존재하는 박스의 좌표 (*xmin, y*min, w, h)
    - area: 객체가 존재하는 박스의 크기
    - category_id: 객체가 해당하는 class의 id
    - image_id: annotation이 표시된 이미지 고유 id
- **Output :** 모델은 bbox 좌표, 카테고리, score 값을 리턴

프로젝트 팀 구성 및 역할

- 남권표 : yolov5 실험
- 장수호 : cascade rcnn 실험
- 유승우 : swin-transform을 backbone으로 하여 실험
- 조유진 : efficiendet 실험
- 김기훈 : yolov5,swinL 실험
- 김승규 : swin-transformer 실험

프로젝트 수행 절차

1. EDA 및 데이터 전처리
2. 모델 선정 및 역할 분담
3. 실험 진행 및 실험 결과 공유
4. 앙상블

프로젝트 수행 결과

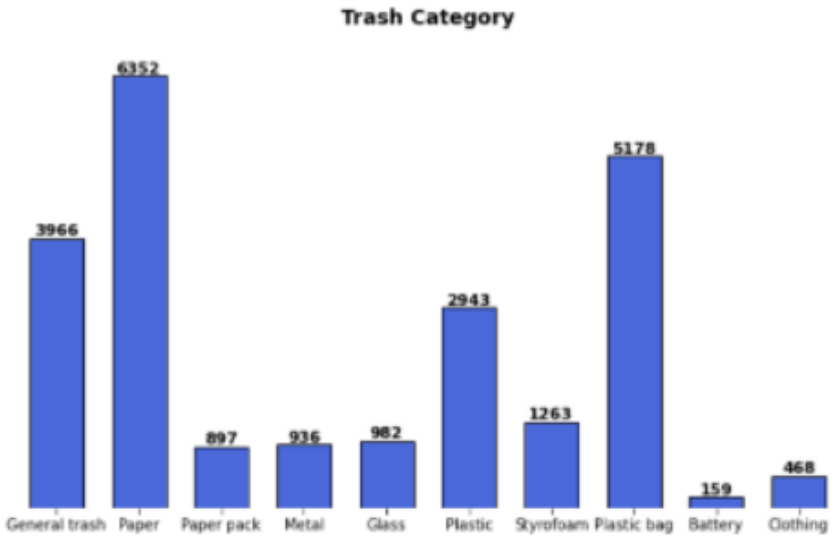
a. 탐색적 분석 및 전처리

   a. 데이터 파이프라인

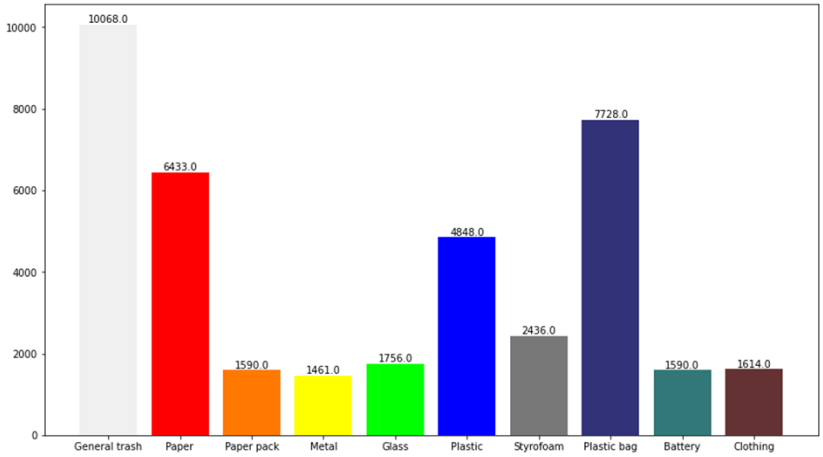
PredictionString	image_id
0 0.11504305 483.02075 577	test/0000.jpg
0 0.083776586 350.1274 251.	test/0001.jpg
0 0.8938034 879.84656 488.0	test/0002.jpg
0 3.2597287e-05 211.76904 9	test/0003.jpg
0 0.83442813 408.6574 405.1	test/0004.jpg
0 0.0008180606 989.5477 84	test/0005.jpg
0 0.9332225 0.0 811.90906 9	test/0006.jpg
0 0.37462467 0.0 472.91483	test/0007.jpg
0 0.9808349 104.74364 905.6	test/0008.jpg
0 0.99991155 292.91437 118.	test/0009.jpg
0 0.24220644 6.2266693 812.	test/0010.jpg
0 0.7763143 0.0 0.0 777.361	test/0011.jpg
0 0.49130407 280.52582 409	test/0012.jpg
0 0.026991138 211.6649 483.	test/0013.jpg
0 0.06413126 22.795303 545.	test/0014.jpg

   a. 결측치 확인
   결측치는 없었지만 잘못된 라벨링이 존재하여 해당 라벨링은 수정하였다.
   ex) 323번 사진: 캔이 여러 개 모여있지만 plastic bag으로 분류되어 있었음. (plastic bag → metal)

   b. Class 분포도
   Paper, Plastic bag, General trash의 경우 다른 class에 비해 많은 데이터를 가지고 있고, Battery, Clothing의 경우 상대 적으로 적은 비율의 데이터를 가지고 있다.



다른 데이터에 적은 비율을 가진 Battery와 Clothing, 다양한 형태가 존재하는 General trash를 offline augmentation으로 데이터 양을 늘려주었다.



test image 갯수 4883개 → 9501개로 증가

b. Augmentation

   a. 기본 augmentation (resize, flip, rotate, noise, brightness, huesaturation 등)

   b. 여러 이미지를 섞는 augmentation (Mixup, Cutmix, Mosaic 등)

c. 모델 선정 및 실험 분석

   a. 선정
   각자 연구한 신경망을 사용해 만든 모델 중 가장 성능이 좋은 것들을 뽑아 앙상블시켰다. 앙상블은 wbf 방식을 사용했다.

   b. 분석

      a. TTA(Test Time Augmentation): TTA 적용시 대부분의 경우 좋은 성능을 냈다. yolov5의 경우 mAP가 0.04 정도 향상됐다.

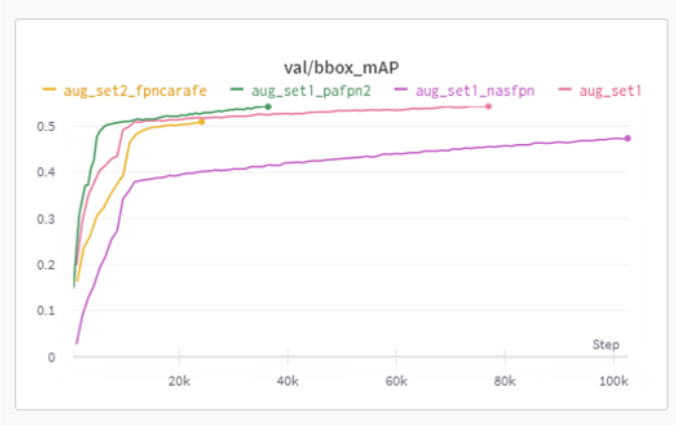
b. Psuedo Labeling: Train 데이터만으로 학습을 시킨 후에 Test 데이터를 넣어서 라벨을 만든다. 그 다음 score를 0.7을 기준으로 높은 박스들만 살려서 다시 학습 데이터와 함께 학습을 시켜서 성능을 향상시켰다(yolov5의 경우 mAP 약 0.03 향상).

c. Multi Scale: Train 단계에서 다양한 scale로 resize 시켜서 학습을 진행하는데 그것을 test 할 때에도 적용시켜서 성능을 향상시킬 수 있었다.

d. Label Smoothing
yolov5에 적용해본 방식이다. Label Smoothing을 사용하지 않은 기존 모델에 비해 mAP는 떨어지지만(mAP 약 0.04 하락) loss 함수가 좀 더 안정적인 모습을 보인다. multi-scale과 함께 적용 시 오히려 성능이 안 좋은 모습을 보였다(mAP 약 0.02 하락).

e. neck: cascade RCNN의 neck 모델을 바꾸어가며 베이스 모델인 FPN과 비교 실험을 진행하였다. NASFPN의 경우 성능이 현저히 떨어졌고, PAFPN은 FPN과 큰 차이를 보이지 않았다. FPN-carafe는 훨씬 적은 epoch에도 비슷한 성능을 보여 FPN-carafe로 선정하였다.

- validation 결과



- test 결과

1. FPN : `70 epoch` `0.5161 mAP`
2. NASFPN: `96 epoch` `0.4441 mAP`
3. PAFPN: `70 epoch` `0.5380 mAP`
4. FPN-carafe: `20 epoch` `0.5359 mAP`

d. 모델 평가 및 개선

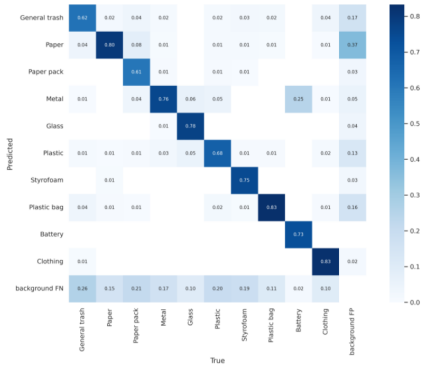
a. 모델 평가
train set과 validation set을 분리하여 validation mAP를 평가했다. 또한 예측 결과의 bounding box을 그려서 눈으로도 직접 비교해보았다.

b. 모델 성능 개선 방법
augmentation, pseudo-labeling, multi-scale, TTA 등의 기법을 사용했다.

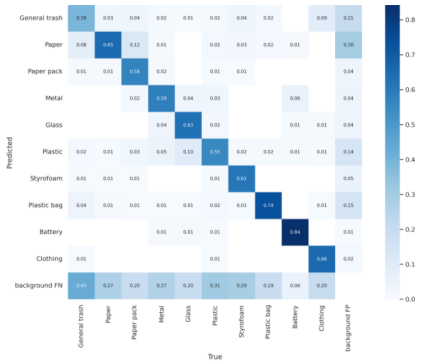
e. 모델 성능

a. yolov5

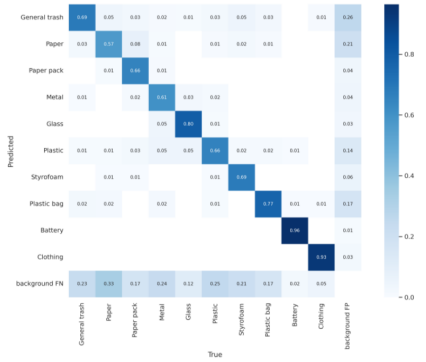
- 기본적인 yolov5이고 tta, multi-scale을 적용하지 않았을 때 LB : `0.5652 mAP`



- multi-scale,tta,pseudo labeling를 적용했을 때 LB: `0.5967 mAP`



- multi-scale,tta,psudo labeling을 적용한 것에 augmentation을 추가한 것 : `0.6139 mAP`



- label smoothing, TTA를 적용한 것에 augmentation을 추가한 것: `0.5630 mAP`

b. cascade r-cnn
backbone 으로 swin-transform 계열을 사용하고, augmentation, TTA 적용했다.
swin-base 를 사용했을때 가장 성능이 좋았다. LB : `0.6192 mAP`

c. efficientdet
efficientdet D5에 offline augmentation과 TTA를 적용하여 LB : `0.5161 mAP` 까지 성능을 올렸다.

d. 최종 제출 (ensemble)
위 신경망들을 베이스로 각각 다르게 학습한 모델들 중에 성능이 좋은 모델들을 앙상블했다. 앙상블은 wbf 방식을 사용했다. 그 결과 LB : `0.6912 mAP`를 달성했다.

자체 평가 의견

a. 잘한 점들
    a. notion과 google 스프레드 시트, weights & biases를 통해 실험 결과를 공유.
    b. github issue를 통해 유용한 정보나 에러 해결 등을 공유.
b. 시도 했으나 잘 되지 않았던 것들
    a. swin L 모델을 너무 늦게 학습을 시켜서 더 다양한 조건에서 성능을 끌어올리지 못한 점이 아쉽다.
    b. 각자 다양한 실험을 진행해봤지만, 서로 다른 valid dataset으로 검증했기에 명확한 실험의 효과를 나타낼 수 없었다.
    c. stratified k-fold를 통해 dataset을 class별로 균등하게 나누긴 했지만 5개의 train,valid set들을 다 활용하지 못했다.
    d. mmdetection도 사용해보고 싶었지만 yolov5만 사용해 본 것 같아서 아쉽다.
c. 아쉬웠던 점들
    a. 실험을 좀 더 계획적으로 수행했으면 좋았을 것 같다.
    b. 코드를 주기적으로 merge해서 기능을 추가해주었으면 좋았을 것 같다.
    c. 학습 후의 Test 데이터에 대한 eda를 진행해봤으면 더 정확한 분석을 할 수 있었을 것 같다.
d. 프로젝트를 통해 배운 점
    a. mmdetection 사용법.