Det Warp Up Report

Github repo : https://github.com/bcaitech1/p3-ims-obd-garbagecollector

0. 목차

1. Competition 개요 및 결과

- 이미지에서 쓰레기나 물건 등의 object를 detection 하는 task
- train[image:2617, annotation:21116], test[image: 837개]
- unknown, general trash, paper, plastic bag 등 총 11개 class
- class간 불균형이 매우 심하다. (개체 수 기준 7000개~50개)
- 학습용 데이터는 매우 깔끔하게 전처리되어 제공되었으며, 512×512 사이즈

2. 대회 목표

- Object Detection 과제에서의 최신 트렌드 이해 및 구현
- EDA를 통한 데이터 이해와 실험을 통한 가설 증명

3. 대회 진행 11일간의 과정

Day01~02 5/10~11 (0.4101 : mAP)

- baseline 코드 작성

- detection 모델 구조 공부
- mmdetection 사용법 숙달

Day03 5/12 (0.4101)

- mmdetectoin에 의존하지 않고 pytorch만을 이용한 basline 코드 작성 (EfficientDet 및 torchvision 모델 사용)
- notion으로 협업 환경 구성
- mAP metric에 대한 탐구

Day04 5/13 (0.4470)

- yolo(0.4370) 및 swin-transformer 모델 사용
- anchor box 사이즈에 대해 토론
- 데이터 시각화

Day05 5/14 (0.4507)

- mmdetection에 구현된 모델 탐색

Day06~07 5/15~16 (0.4714)

- swin_base 제출 (0.4714)
- pseudo labeling 시도
- loss Function 및 Augmentation 탐색

Day08 5/17 (0.4755)

- 2차 pseudo labeling
- multiscale Ensemble
- sabl_faster_resnest50_pafpn, ratios 증가 (0.4755)

Day09~10 5/18~19 (0.4945)

- sabl_faster_resnest50_pafpn, ratios 증가 및 scales 증가 (0.4777)
- sabl_faster_resnest50_pafpn, ratios 증가 및 scales 증가, unknown에서 general (0.4860)
- swin_base tta (0.4714 → 0.4945)

- class mix → map 하락

Day11 5/20 (0.5762)

- WBF로 앙상블, k-fold

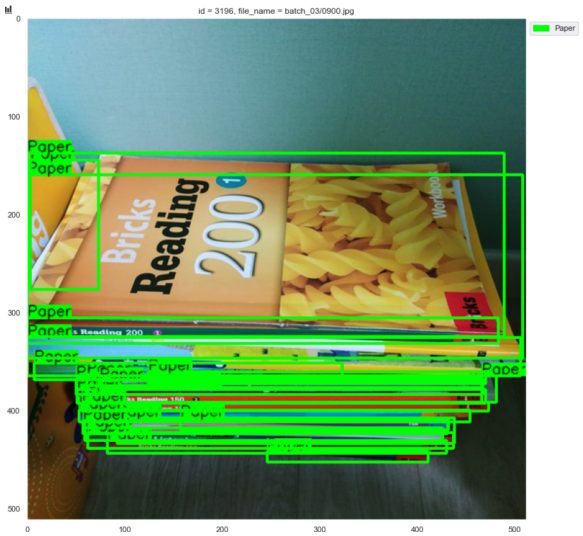
4. 최종 앙상블에 사용한 모델

- sabl_faster_resnest50_pafpn #multiscale (0.5232)
  faster_rcnn에 bbox_head를 sabl head로 사용했습니다.
- faster_rcnn + SCNet101 + FPN #multiscale #TTA (0.5200)
  http s://github.com/MCG-NKU/SCNet
- swin_transformer_base #multiscale (0.4945)
- yolo v5 (0.4628)

5. 성능 향상에 성공한 아이디어

anchor ratio

- EDA 결과 아래와 같이 가로로 긴 박스가 존재하였고, 빨대 등의 세로로 긴 박스도 잘 찾기 위해 anchor box의 ratio를 그에 맞게 적용해 보았습니다.
- 0.1, 2.5등 기존 ratio에 비율을 추가하였습니다.



unknwon2general

- +------------+-------+--------------+-------+------------+-------+
UNKNOWN	0.001	General trash	0.119	Paper	0.291
Paper pack	0.207	Metal	0.248	Glass	0.282
Plastic	0.146	Styrofoam	0.346	Plastic bag	0.558
Battery	0.362	Clothing	0.205	None	None
  +------------+-------+--------------+-------+------------+-------+
  모델의 class별 성능을 나타낸 지표입니다. UNKNOWN 클래스는 거의 맞추지 못한다고 판단했고, 해당 label을 지우는 방향으로 실험하였습니다.
- 처음에는 단순히 unknown을 전부 general로 재 라벨링하고 전체 class는 11개 그대로 두어 학습했는데 꽤 많은 성능 향상이 있었습니다.
- 두번째로 unknown class를 아예 제거하고 10개의 클래스로 학습을 진행해봤으나, 오히려 기존보다 성능이 하락하였습니다. 이유는 밝혀내지 못했지만, UNKNOWN이 아예 0인것 보다는 잘못 예측하더라도 박스가 무조건 많은 것이 mAP 스코어 향상에 도움이 되었기 때문이라고 생각합니다.

SCNet

backbone	AP	AP.5	AP.75	APs	APm	APl
ResNet-50	37.6	59.4	40.4	21.9	41.2	48.4
SCNet-50	40.8	62.7	44.5	24.4	44.8	53.1
SCNet-50_v1d	41.8	62.9	45.5	24.8	45.3	54.8
ResNet-101	39.9	61.2	43.5	23.5	43.9	51.7
SCNet-101	42.0	63.7	45.5	24.4	46.3	54.6

- EfficientNet보다 빠르며 Resnet보다 성능이 좋은 backbone인 SCNet을 사용하였습니다.
- 비교적 작은 SCNet50으로 실험을 하였고 결과를 SCNet101에 적용하여 성능 향상을 볼 수 있었습니다.
- https://github.com/MCG-NKU/SCNet

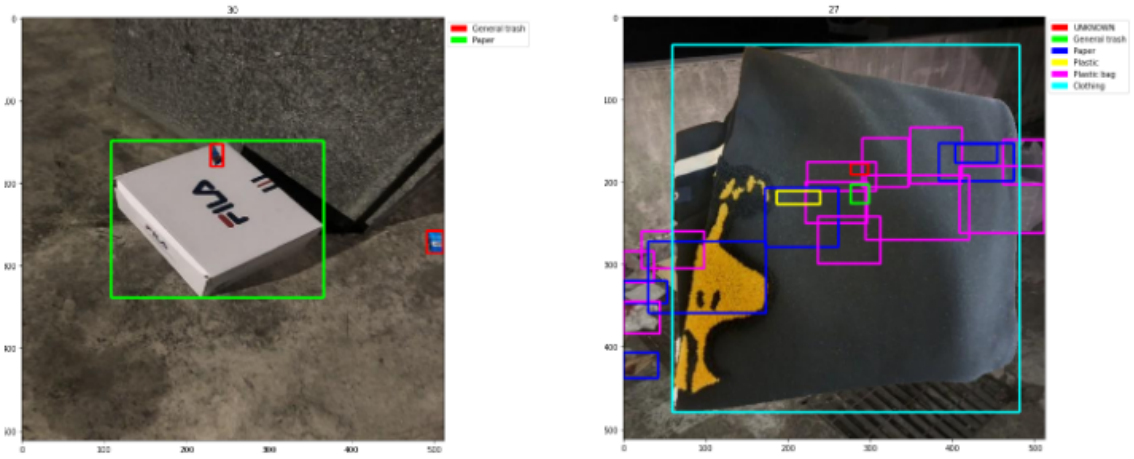
Post Processing

- Inference된 결과의 EDA를 바탕으로 Bounding Box가 과도하게 생성되어 있는 것을 발견하였고, 이를 개선하기 위해 WBF, NMS, Soft-NMS를 적용하였습니다.
- 그 중 WBF의 성능이 가장 좋았습니다.

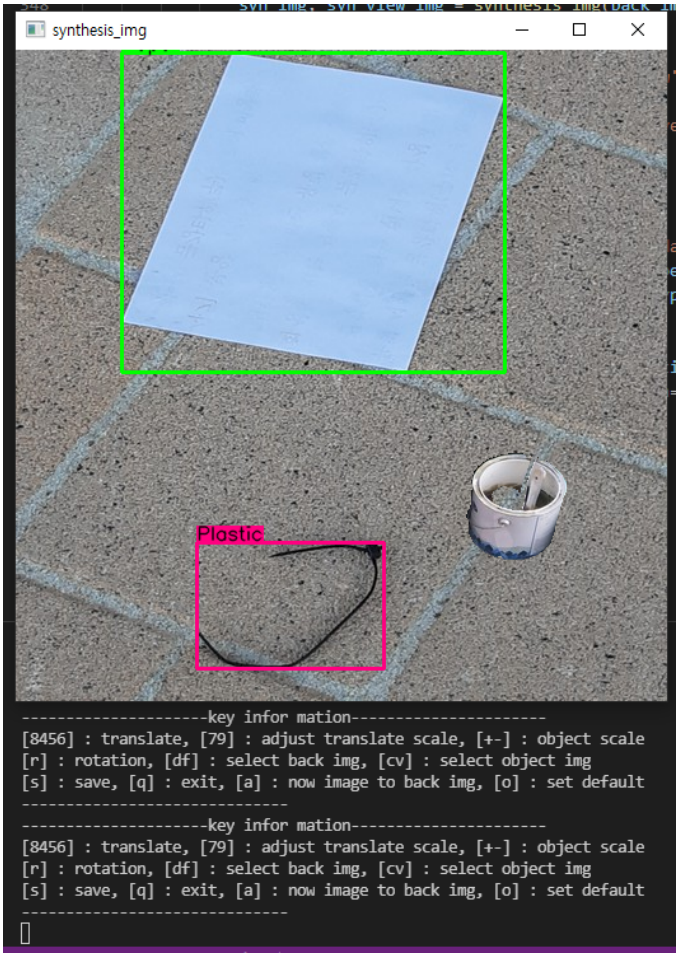
6. 성능 향상에 실패한 아이디어

ClassMix

- 우리 모델의 class별 성능을 보고 낮은 class의 object 위주로 붙여넣었습니다.
- 또한 segmentation에서의 실패 이유를 떠올려, object의 오버랩을 방지하기 위한 아이디어를 강구하였습니다.
- 방법1 : 붙여넣을 object와 이미 존재하는 object간의 iou가 낮을때만 붙여넣는 방법
  너무 작은 object 위주로 합성되거나, 오히려 너무 큰 object가 합성되어 기존의 object가 오버랩되는 경우가 발생했습니다.



- 방법2 : 고정된 위치가 아닌 수기로 위치, 사이즈, 회전을 반영하여 합성을 하였으나 성능이 하락되었습니다. 동일한 오브젝트로 인한 성능하락 일지, 너무 적은 양의 추가 데이터로 인한 결과 인지는 실험하지 못하였습니다.



Loss Function 탐색

- **GIoU**

$$GIoU = IoU - \frac{|C \backslash (A \cup B)|}{|C|}$$

기존의 1-IoU를 사용하는 IoU loss보다 loss함수로 활용하기 좋게 만들어졌습니다. 두 박스의 외곽 박스 크기까지 고려하여 계산합니다.

- **DIoU**
  GIoU는 Horizontal과 Vertical 정보를 고려하지 못한다는 문제점이 있었습니다. 아래와 같이 박스가 어디에 치우쳐져 있든 간에 항상 같은 loss값을 보여줍니다. 중심 좌표

간의 거리를 추가적으로 고려하여 predicted box의 치우쳐진 위치에 따라 다른 loss를 계산할 수 있도록 해결한 방법입니다.

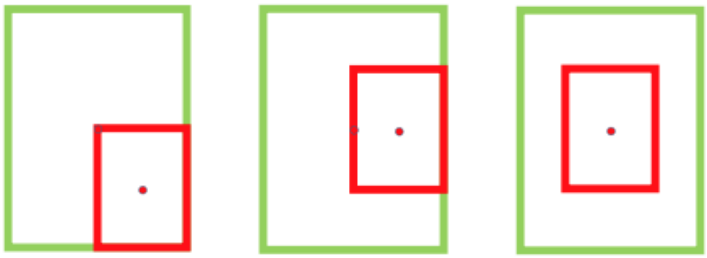


Figure 2: GIoU loss degrades to IoU loss for these cases, while our DIoU loss is still distinguishable. Green and red denote target box and predicted box respectively.

- **CIoU**
  https://github.com/Zzh-tju/CIoU

실패한 모델

- BiFPN, NAS-FPN : 논문과 SOTA지표를 보고 성능이 더 좋다고 하여 사용해봤으나, 성능 하락을 경험했습니다. 대회에 사용된 데이터가 여러 스택을 거쳐 feature를 해야 할 만큼 다양한 feature가 존재하지 않았던 것 같습니다.
- HRNet, HRFPN : 작은 객체들을 효율적으로 잡아낼 수 있다고 판단하여 사용했으나, 다른 Backbone, Neck들과 비교했을때 성능 향상이 나타나지 않았습니다.
- SCNet : Bounding box 뿐만 아니라 mask, instance segmentation 정보 또한 학습에 필요한 모델이기 때문에, dataset에 존재하지 않는 segmentation head를 삭제하여 학습 시켰으나, 성능이 좋지 않았습니다. segmentation data를 생성하여 학습을 진행하였다면 더 좋은 성능이 나올 수 있을 것 같습니다.
- VFNet : 24epoch를 학습한 결과, 성능이 좋지 않았습니다. 학습Log을 살펴봤을때, validation loss가 계속 떨어지고 있었기 때문에, 더 많은 epoch를 학습했을때에 더 높은 성능을 기대할 수 있으나, 24epoch 학습에 8시간 정도의 긴 시간이 걸렸기 때문에, 적용하지 않았습니다.
- EfficientDet : 논문에 성능이 뛰어나다고 소개되어 학습해보았으나, 100 epoch 정도 학습했음에도 성능이 좋지 않았습니다. 실제 논문에서도 저희보다 큰 데이터로 300 epoch을 학습시켜 좋은 성능을 내려면 좋은 데이터와 긴 학습 시간이 필요할 것 같습니다.

pseudo labeling

- 최초 실험 결과 성능이 처참했고, 이유를 분석해 보았습니다.
  1. inference 결과물 자체가 별로라서?
  2. pseudo_labeling할 때 score_thr 값이 문제인지?
  3. 데이터가 많아져서 lr을 조정해줘야 하는건지
- lr감소 스텝을 기존보다 2에폭 당겨서 적용해봤습니다.
  [8,10,12] → [6,8,10]
  여전히 낮은 스코어를 기록하였습니다.
- score_thr값을 높여봤습니다. 어느정도 성능이 높아졌지만, 사용할 수 있을 정도는 아니었습니다.
- 당시 최고 결과물이었던 LB 0.5250 기준 output으로 진행하였을 때, 역시 성능이 하락하여 실패했다고 판단했습니다.

실패한 Augmentation

- mosaic augmentation을 mmdetection에 적용을 하였으나 데이터에 작은 object들이 많았고 이 데이터에 mosaic을 적용해 더욱 작은 object가 되었기 때문이라고 판단합니다. input size를 더욱 크게 주었다면 결과가 달라 졌을 수도 있을 것 같습니다.

7. 회고

mAP에 대한 고찰

- nms로 중첩되는 bounding box를 제거 후 시각화를 통해 명확해진 detection 결과를 확인하였습니다. 하지만, LB mAP50에는 성능이 하락했습니다.
- confidence score threshold를 낮추었을 때, 수 많은 bounding box가 만들어 지는 것을 확인했으나, LB mAP50 성능은 상승하였습니다.

아쉬웠던 부분

- 이번 대회는 얼마나 유의미한 데이터로 학습을 하는가가 많이 중요했던 것 같습니다. class mix 데이터 합성 도중 `전단지` 처럼 `paper`나 `general trash`로 나뉘어져 있는 경우를 보았는데 이러한 데이터들의 label을 하나로 통일하면 더욱 성능이 올랐을 것 같습니다.
- Anchor box 관련 가설을 대회 초반에 세우고 제대로 검증하고 모델에 적용했다면 성능 향상이 있었을 것 같습니다.

- 초반에 모델에 대한 이해가 부족하여 모델 중 한 부분을 유기적으로 변경하지 못하였고 따라서 많이 실험해보지 못하였습니다.
- ensemble을 하는 시간을 미리 확보를 했다면 조합이 좋은 모델들에만 몰입하여 더 빠른 실험을 진행 할 수 있었을 것 같습니다.
- 동일한 객체에 대해 여러개의 다른 label의 Bounding Box가 생성되는 것을 발견하여, Post Processing을 통해 성능 개선을 위해 노력하였으나, confidence score만을 기반으로 Bounding Box를 줄여나갔기에, correct label이 지워지는 결과가 발생하였고 결과적으로 성능 향상을 보지 못하였습니다.
- 토론 게시판에 올라온 성능이 좋다고 하는 모델들을 많이 실험해봤지만, 저희가 실험해본 결과는 좋지 않았고 이유를 알아내지 못하였습니다.
- HTC 기반의 모델들은 bounding box, mask, segmentation 정보를 학습에 이용합니다. 그러나 주어진 dataset에는 segmentation 정보가 없어, segmentation head를 삭제하고 학습을 진행할 수 밖에 없었습니다. segmentation 데이터를 직접 생성하는 코드를 구현하여 HTC 기반 모델들을 적용했다면 더 좋은 성능을 얻어낼 수 있다고 생각합니다.
- 대회 초반부터 Swin Transformer 모델 및 SOTA 모델들부터 시도하지않고 base 모델을 사용 및 실험을 강행한 측면에서 효율적인 시간사용이 아니었다고 생각합니다.
- 대회 도중 mAP를 제대로 이해하지 못하고 실험을 강행하여 제출 기회를 낭비한 경우가 있어 metric을 더욱 꼼꼼히 읽어보고 이해한 후 실험들을 진행해야겠다고 생각했습니다.

잘했던 부분

- 논문에 제시된 다양한 모델들을 구현하여 실험을 하였고, 성능이 좋은 모델들을 선별할 수 있었습니다. 최종적으로 앙상블을 적용하는데 큰 도움이 되었습니다.
- nms실험, anchor ratio, classmix 등 대회 데이터를 고려한 가설을 세우고, 실제로 구현하는 데 성공하였습니다.
- 새벽에도 실험을 진행할 만큼 끝까지 포기하지 않고 성능 향상을 위해 노력하였습니다.

- 주어진 데이터, 코드에 만족하지 않고, 새로운 기능을 직접 구현하여 데이터를 가공하였습니다.

시도해보지 못한 부분

- UniverseNet (1 Stage)
- Sampler 변경 : Random Weighted Sampler, Balanced Sampler
- InstaBoosts
- SWA (Stochastic Weight Averaging)
- swin + HTC

  관련 내용