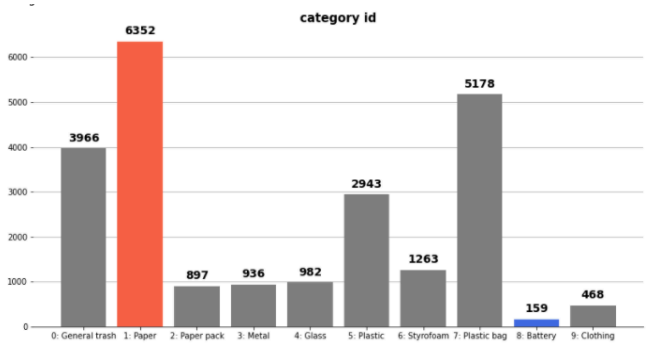
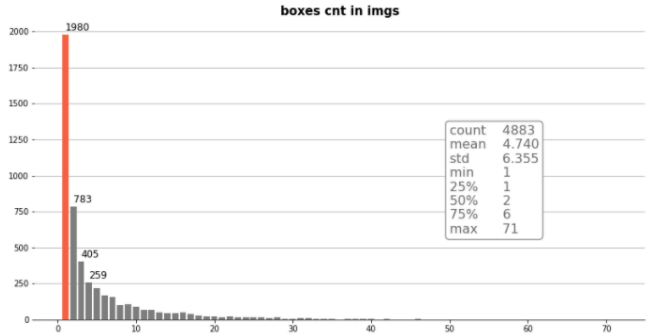
Trash Object Detection

1️⃣ EDA

- train 사진 수: 9754장
- 10 class : General trash, Paper, Paper pack, Metal, Glass, Plastic, Styrofoam, Plastic bag, Battery, Clothing
- image size: (1024, 1024)
- Evaluation: mAP50





- Class Imabalance가 매우 심각하다.
- bbox가 1개인 경우가 많다.

2️⃣ Testing

- Seed: 2021 고정
- 리더보드에 제출한 것들 위주로 정리

Architecture/backbone/Neck	**Val_mAP50**	**LB_mAP50**	**optimizer**	**class_loss, bbox_loss**	**batch, epochs**	**ETC**
Faster R-CNN / ResNet50 / FPN	-	0.428	SGD, StepLR, lr=2e-2	CE, L1Loss	2, 12	Baseline
Cascade R-CNN / (ResNet50, ResNet101) / FPN	0.617, 0.425	0.349, 0.250	SGD, StepLR, lr=1e-3	CE, SmoothL1Loss	4, 36	
Cascade R-CNN / ResNet50 / FPN	0.715	0.521	SGD, **CosineAnnealing**, lr=1e-3 ~ 5e-6	CE, SmoothL1Loss	4, 36	[1], 이후 적용
Cascade R-CNN / ResNet50 / **RFP+SAC (DetectoRS)**	0.810	0.559	SGD, CosineAnnealing, lr=1e-3 ~ 5e-6	CE, SmoothL1Loss	4, 36	
Cascade R-CNN / **Swin base** / FPN	0.620 (12 epochs)	0.540 (12 epochs)	**AdamW, CosineAnnealingWarmRestarts, lr=1e-5 ~ 5e-6**	**FocalLoss, DIoULoss**	4, 36*	[2]
Cascade R-CNN / Swin **large** / FPN	0.220 (5 epochs)	0.170 (5 epochs)	AdamW, CosineAnnealingWarmRestarts, lr=5e-5 ~ 5e-6	FocalLoss, DIoULoss	4, 48*	[3], 이후 적용
Cascade R-CNN / Swin large / FPN	-	-	AdamW, CosineAnnealingWarmRestarts, lr=1e-5 ~ 5e-6	FocalLoss, DIoULoss	4, 60*	[4]
Yolov5 small, large	0.9119, 0.9724	-, 0.492	SGD, StepLR, lr=1e-2	CE	32, 150	[5], 이후 적용
Yolov5 large	0.9777	0.517	SGD, StepLR, lr=1e-2	CE	32, 100 / 200(pseudo)	**pseudo 시도**

[1] Data argument

- RandomRotate90, GaussNoise(p=0.5), Resize(1024, 1024), RandomFlip(p=0.5), Normalize
- Blur, MedianBLur
- RandomBrightnessContrast, CLAHE, RandomGamma
- HueSatuationValue, RGBShift

[2] OutOfMemory, overfitting 대폭 감소

- soft_nms 적용
- 36epoch으로 설정했으나, OutOfMemory로 12 epoch 밖에 안 돔

[3] 성능은 좋으나, 속도가 너무 느려서 중도에 멈춤 (4일 걸림)

- Swin Transformer object detection에 맞춰 mmdetection 2.11.0으로 버전 변경, 맞춰서 코드 수정
- soft_nms 적용
- score_thr=0.0, anchor aspect_ratio=[0.5, 1.0, 2.0] => [0.33, 1.0, 2.0, 3.0]
- 효율적인 학습을 위해 Mixed Precision (NVIDIA apex) 사용

[4] 성능은 더 좋아졌으나, 그에 비례해 속도가 느려져서 중도에 멈춤 (5일 걸림)

- Multi-size training (384, 384), (512, 512), (768, 768), (1024, 1024) & TTA with Multi-size images

[5] Yolov5 기본

- Yolov5에서 기본으로 제공하는 data argument(Mosaic 포함) + img-size=(512, 512)
- TTA, Multi-size training

3️⃣ Result

비교	모델 (swin, yolov5은 large / resnet은 101)
성능	cascade rcnn+swin > cascade rcnn+resnet(detectoRS) > Yolov5 > cascade rcnn+resnet > faster rcnn
속도	Yolov5 > faster rcnn > cascade rcnn+resnet > cascade rcnn+resnet(detectoRS) > cascade rcnn+swin

비교	속도 및 성능 비례
class_loss	FocalLoss > CE
bbox_loss	DIoULoss > CIoULoss > GIoULoss > SmoothL1Loss > L1Loss
optimizer	Adam > AdamW > SGD
lr scheduler	CosineAnnealingWarmRestarts > CosineAnnealing > ReduceLROnPlateau > StepLR
nms	cluster_nms(예상, 사용 못해봄) > soft_nms > nms

- Train : Valid set, 9:1로 분할

  > Stratified Group K-fold로 진행하려 했으나, 1번 돌리는데도 시간이 너무 많이 소비되어 진행하지 않았다.

- Swin Transformer, mmdetection

  > mmdetection 2.17.0에서 지원해주는 swin transformer 같은 경우, 용량의 몇 배를 더 차지해 버전을 변경해서 진행했다. 대신 버전을 낮춰 진행하다보니, Mosaic, MixUp, 등 최근에 나온 기법들을 활용할 수 없어서 불편했다.

- Wandb sweep, Yolov5 `--evolve` 사용

  > 둘 다 최적의 hyperparameter를 찾는 기능이지만, wandb sweep의 자유도가 더 좋았다. yolov5의 evolve은 마지막에 성능을 쥐어짜는 느낌이었고, sweep과 같은 효과를 볼려면 적어도 300 epochs은 돌려야 한다고 하는데, 모델이 무거우면 시간 대비 효율성이 떨어지는 느낌을 받았다.

- Yolov5에서 label smoothing 적용 시, mAP 하락

  > 실제 visualize하면, 필요없는 박스 개수가 확연하게 줄어든 것을 볼 수 있었으나, 박스 개수 많음 == 답이 얻어걸림 == mAP 상승이기에 자연스럽게 mAP가 하락한 것 같다. overfitting 정도는 label smoothing 유무와 상관없이 비슷했다.

- Pseudo

  > 대체로 성능은 향상했지만, 하지한 현업에서 사용하기에는 위험할 것 같다. (라벨링을 확인해보면 잘못 라벨링된 것, 쓸모 없는 박스들이 꽤 많았기 때문이다.)

- score threshold

  > score threshold를 0으로 하여 박스를 무작정 많이 그리게 했을 때, mAP가 가장 높게 나오는 것을 확인할 수 있었다. 이런 모델은 실제 현업에서는 서비스 과정에 사용이 불가능할 것으로 예상되고, 대회용 trick으로만 활용될 것 같다.

4️⃣ Furthermore...

우선 모델 하나 돌리는데 시간이 오래 걸리고, 초반에 `OutOfMemory`로 시간을 많이 소비했다. 3주였지만, 저번 대회에 비해 여러가지 많이 실험해볼 수 없어서 아쉬웠다. 환경과 메모리의 중요성을 새삼 다시 느낄 수 있었다. 서버가 한 개 더 있었으면 하나는 대회용, 하나는 개인 공부용으로 돌렸을 텐데 아쉬웠다. 여러가지 많이 실험해보지는 못했지만, 그에 반비례해서 저번 대회보다 많은 논문을 읽고 리서치를 한 것 같다.

이번 대회는 최대한 다양한 모델들을 다뤄보기가 목표였는데, 어느 정도 달성한 것 같아서 만족한다. (기본인 faster rcnn부터 cascade rcnn, detectoRS, swin, Yolov5, wandb sweep, pseudo, 다양한 loss, lr scheduler, multi-scale, Mixed Precision, 등을 사용해봤다.) 그리고 처음부터 python script로 시작해서 저번보다 직관적이고, 코드 작성 및 수정이 편했다. 딱히 hyperparameter는 건들지 않고 overfitting 확인을 위해 제출해보는 식으로 진행했다. EDA 부분에 있어서는 아쉬움이 많은 대회였다. 리더보드 신경 안 쓰니까 진짜 공부를 한 느낌이었지만, 이에 더불어 현타가 또 오는 것 같기도 했다. 잘못하다가는 이것도 저것도 아닐까 걱정이 되면서 대회를 마무리 지었다.

> **Reference sites**
>
> **Personal repo** (추후에 여유로울 때, README.md 작성할 예정)