재활용 품목 분류를 위한 Semantic Segmentation

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.

분리수거는 이러한 환경 부담을 줄일 수 있는 방법 중 하나입니다. 잘 분리배출 된 쓰레기는 자원으로서 가치를 인정받아 재활용되지만, 잘못 분리배출 되면 그대로 폐기물로 분류되어 매립 또는 소각되기 때문입니다.

따라서 우리는 사진에서 쓰레기를 Segmentation하는 모델을 만들어 이러한 문제점을 해결해보고자 합니다. 문제 해결을 위한 데이터셋으로는 배경, 일반 쓰레기, 플라스틱, 종이, 유리 등 11 종류의 쓰레기가 찍힌 사진 데이터셋이 제공됩니다.

- images:
  - id: 파일 안에서 image 고유 id, ex) 1
  - height: 512
  - width: 512
  - file*name: ex) batch*01_vt/002.jpg
- annotations:
  - id: 파일 안에 annotation 고유 id, ex) 1
  - segmentation: masking 되어 있는 고유의 좌표
  - bbox: 객체가 존재하는 박스의 좌표 (x*min, y*min, w, h)
  - area: 객체가 존재하는 영역의 크기
  - category_id: 객체가 해당하는 class의 id
  - image_id: annotation이 표시된 이미지 고유 id

- **Output :** pixel 좌표 별 카테고리 값을 리턴

프로젝트 팀 구성 및 역할

- 남권표 : Vit + uperNet/segmenter 및 여러 모델 실험
- 장수호 : convnext + knet 모델 성능 개선. 베이스라인 코드 및 다양한 실험 유틸 코드 작성
- 유승우 : BEIT + uperNet 모델 성능 개선 및 offline data(Mosaic, CutMix 적용) 제작
- 조유진 : EDA, 여러 모델 실험 및 조사 (convnext, swin, vit, mit-b5 / knet, setr-naive, segformer)
- 김기훈 : convnext+uperNet 모델 성능 개선 및 augmentation, pseudo labeling, weighted ensemble

- 김승규 : Swin-L + uperNet 모델 성능 개선 및 mmsegmentation baseline 코드 작성

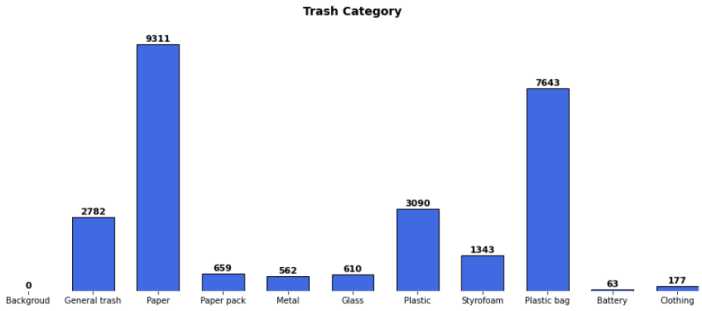
프로젝트 수행 절차

1. EDA
2. 모델 선정
3. 실험 진행 및 결과 분석
4. 앙상블

프로젝트 수행 결과

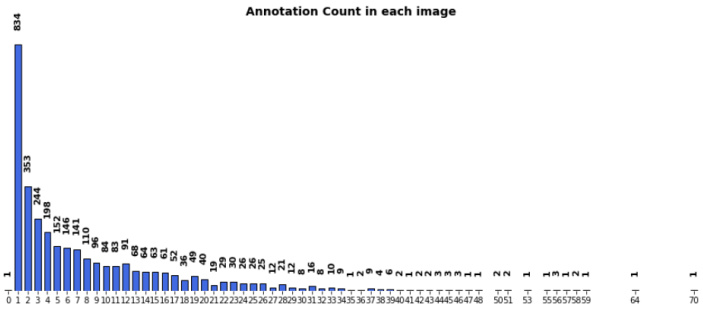
a. EDA

a. 다른 class에 비해 Paper, Plastic bag이 많은 데이터를 가지고 있고, Battery, Clothing은 비교적 적은 데이터를 가지고 있다.



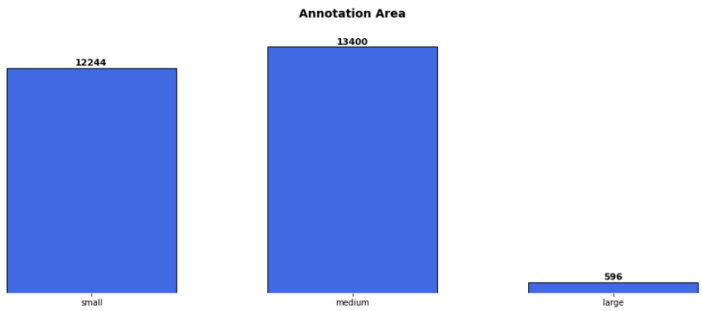
category 별 annotation 수

b. 한 이미지에 annotation이 없는 이미지도 존재하였다.



한 이미지 당 annotation 수

c. mask area를 (50 x 50), (250 x 250)를 기준으로 small, medium, large 로 나누어 보았을 때, large의 개수가 매우 적었다.



annotation area 분포

b. 모델 선정

a. 선정

SOTA와 mmsegmentation의 최신 모델들을 위주로 여러 모델을 실험하여 validation이 잘나오는 모델을 선정.

1. convNeXt + uperNet
2. convNeXt + knet with uperNet
3. swinL + knet with uperNet

c. 실험 진행 및 결과 분석

a. TTA(Test Time Augmentation): TTA 적용시 대부분의 경우 좋은 성능을 냈다. convNeXt의 경우 0.0027의 성능 향상이 있었고 image ratio를 2.0의 범위에서 1.5까지로 축소했을 때 0.002의 결과 향상으로 이어졌다.

b. Pseudo Labeling: 제일 잘 예측한 결과로 pseudo labeling 했을 때 약 0.1의 점수 향상으로 이어졌다. 그러나 이것을 너무 과도하게 사용한 결과 오히려 public score에 비해 private score가 많이 떨어지는 결과가 나왔다.

c. Multi Scale: Train 단계에서 다양한 scale로 resize 시켜서 학습을 진행하는데 object detection 대회에서 성능 향상을 경험했기 때문에 이번에도 바로 적용을 시켰고 TTA에도 multi-scale을 적용하였다.

d. augmentation : mosaic, randomcutout, 다양한 blur 기법, flip 등을 사용해봤지만 mosaic augmentation을 제외하고 큰 성능 향상이 없었다. mosaic augmentation의 경우도 swin-L가 아닌 다른 모델을 사용했을 때에는 성능 하락이 있어 추후에는 적용시키지 않았다.

e. Pseudo Labeling

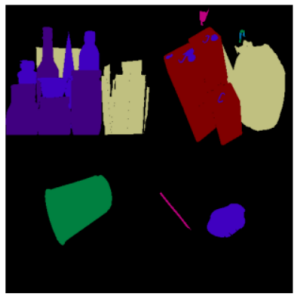
a. 모델을 처음 부터 학습

b. 모델 학습 파일을 불러와서 중간부터 학습

c. 중간부터 pseudo label을 없애고 다시 학습

3가지의 다양한 pseudo labeling 기법을 활용해보았다. (c방법이 제일 효과가 좋았다.)

f. Mosaic Offline Augmentation : 부족한 Label에 대해 Mosaic augmentation을 적용하여 학습 데이터에 추가

g. CutMix Offline Augmentation : 부족한 Label에 대해 CutMix augmentation을 적용하여 학습 데이터에 추가


h. copy&paste augmentation : cutmix와 비슷하게 mask 부분을 잘라서 다른 파일에 붙이는 방식의 augmentation이다. 약 0.003의 성능 향상을 이뤄냈다.

i. Ensemble : hard voting과 class-weighted, 두 가지 방식으로 적용. 하지만 적용 전보다 약간이지만 성능이 오히려 떨어지는 경향을 보였다.

d. 모델 별 최고 mIoU

LB mIOU	Val IOU	Backbone	Architecture	Augmentation

LB mIOU	Val IOU	Backbone	Architecture	Augmentation
`0.8123`	`0.7400`	Swin-L	knet+upernet	RandomBrightnessContrast, HueSaturationValue, GaussNoise CLAHE, blur
`0.8225`	없음	convNext XL	knet+upernet	ShiftScaleRotate,[ElasticTrans,Perspective,PiecewiseAffine], Affine, [RGBShift,ChannelShuffle],RandomBrightnessContrast,CLAHI [Blur,GaussianBlur,MedianBlur]
`0.7742`	`0.7539`	BEiT-L	upernet	Resize,RandomCrop,RandomFlip,PhotoMetricDistortion
`0.8148`	`0.7177`	SVT-L	upernet	Resize, RandomCrop, RandomFlip, PhotoMetricDistortion
`0.8181`	`0.7647`	convNext XL	uperNet	albumentation,copy&paste

1. [Backbone] convnext XL / [Architecture] knet+upernet : `0.8225`
2. [Backbone] swin-L / [Architecture] knet+upernet
    a. 기본 mIoU `0.7186`
    b. tta, mosaic augmentation 적용해서 mIoU `0.7962`
    c. pseudo labeling 적용해서 mIoU `0.8123`
3. [Backbone] BEiT-L / [Architecture] upernet : `0.7742`
4. [Backbone] SVT-L / [Architecture] upernet: `0.8148`
5. [Backbone] contNeXt xl / [Architecture] uperNet
    a. 기본 mIoU `0.7328`
    b. 최종 가장 높은 점수 : `0.8181`

a. 최종 제출

	Backbone	Architecture	mIoU
Public Score	convnext xl	knet+upernet	`0.8225`
	hard voting ensemble		`0.8205`
Private Score	convnext xl	knet+upernet	`0.7527`
	hard voting ensemble		`0.7475`

- hard voting ensemble models

Backbone	Architecture
convNext-XL	knet + upernet
convNext-XL	upernet
swin-L	knet+upernet
SVT-L	upernet

자체 평가 의견

a. 잘한 점들
    a. 초기에 SOTA 기반 다양한 모델을 빠르게 테스트해보고, 그 중 리더보드 점수가 좋은 모델들을 골랐다.
    b. 같은 Valid set을 가지고 테스트해서, 다른 캠퍼들과 비교가 용이했다.
    c. notion과 google 스프레드 시트, weights & biases를 통해 실험 결과를 공유.
    d. github issue를 통해 유용한 정보나 에러 해결 등을 공유.
    e. Pseudo Labeling을 여러 모델에 적용해서 점수를 좀 더 올렸다. (mIoU 0.79xx → 최종 0.82xx)
    f. copy&paste를 사용하여 기존에 적용하던 augmentation들과는 조금 다른 형태의 새로운 augmentation을 적용해 보았다.
b. 시도 했으나 잘 되지 않았던 것들
    a. 픽셀별 Hard voting 방식이나 추가적으로 클래스 별로 가중치를 두어 voting한 방식이 효과가 좋지 못했다.
    b. 앞선 대회들과는 다르게 앙상블이 별로 효과를 보지 못했다.

c. cutmix 방식의 offline augmentation으로 좋은 성능을 얻지 못했다.
d. segmentor, segformer 등 일부 모델을 시도해봤으나 실행 중 오류가 많이 발생하여 실험해보지 못했다.
e. MLflow 활용을 시도했으나 사용에 어려움이 많아 활용하지 못했다.

c. 아쉬웠던 점들
a. mmsegmentation만을 사용했는데, smp와 같은 다른 라이브러리도 써봤으면 좋았을 것 같다.
b. mmsegmentation 내의 모든 기능을 파악하고 사용한 것이 아니라서, 더 다양한 실험(모델 내 파라미터 수정 등)을 하지 못했다.
c. 초기에 선정했던 valid set이 처음에는 public score와 비슷했지만, 추후에는 valid 점수와 public 점수가 많이 달라져서 적절한 valid set을 고르지 못했다는 아쉬움이 있었다.
d. seed도 통일하여 실험을 진행했으면, 실험들 간의 비교가 좀 더 명확했을 것이다.
e. valid mIoU 기준 최고 점수를 저장해서 사용했는데, valid set이 test set과 조금 달라 적절한 epoch에서 저장되었는지 알 수 없었다. valid loss값을 구할 수 없어 early stopping을 적용하지 못했던 것도 아쉬웠다.
f. soft voting 방식을 마지막에 해보다가 시간이 부족해서 제출을 못해본 부분이 많이 아쉽다.
g. data cleansing 작업을 하지 않아 성능 향상에 제약이 있었다.
h. valid mIoU보단 public test mIoU에만 집중하여 일반화 성능을 갖지 못해 private test에서 점수가 많이 떨어진 것 같다.
i. wandb의 visualization기능을 사용했다면, 기존의 실험들의 비교를 효율적으로 할 수 있었을 것이다.

d. 프로젝트를 통해 배운 점
a. mmsegmentation의 사용법을 익혔고 segmentation task에 더 익숙해진 것 같다.
b. cutout, offline augmentation 등 여러 기법을 직접 코딩해 보고 적용해 보면서 해당 기법의 효과를 확인해 볼 수 있었다.
c. 과도하게 pseudo labeling을 사용하지 말아야 한다.
d. seed를 고정하는 것이 성능 비교에 있어서 상당히 중요하다.
e. 학습을 끝낸 후 피드백 하는 과정이 조금 부족해서 그 부분을 보충하는 것이 필요하다.
f. 당장의 높은 점수보다는 일반화 성능에 더 초점을 맞추는게 중요하다.