Wrap UP 리포트

프로젝트 주제	재활용 품목 분류를 위한 Semantic Segmentation		
	이름	캠퍼 ID	역할
팀 장	유승리	T3129	UPerNet 실험, SeMask-FPN 추가 및 실험, TTA 적용
팀 원	이창진	T3169	UPerNet, Deeplabv3, UNet 모델 실험, 자체 Baseline 구현
	심준교	T3124	EDA, BEiT 실험,inference&ensemble 코드 작성,ensemble 진행
	전영우	T3192	데이터 파이프라인 생성, Baseline, Backbone/Loss/Scheduler, knet 실험
	송민수	T3113	Lawin 적용, wandb 설정, mislabeling 실험진행
	김하준	T3066	EDA, 결측치제거, multi-label k fold, BEiT 실험, pseudo Labeling

프로젝트 개요	
	○ 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 Segmentation 하는 모델을 만들어 이러한 문제점을 해결해보고자 합니다. 문제 해결을 위한 데이터셋으로는 일반 쓰레기, 플라스틱, 종이, 유리 등 10 종류의 쓰레기가 찍힌 사진 데이터셋이 제공됩니다. 우수한 성능의 모델은 쓰레기장에 설치되어 정확한 분리수거를 돕거나, 어린아이들의 분리수거 교육 등에 사용될 수 있을 것입니다. ○ 개발 환경 CPU : Intel xeon

GPU : V100

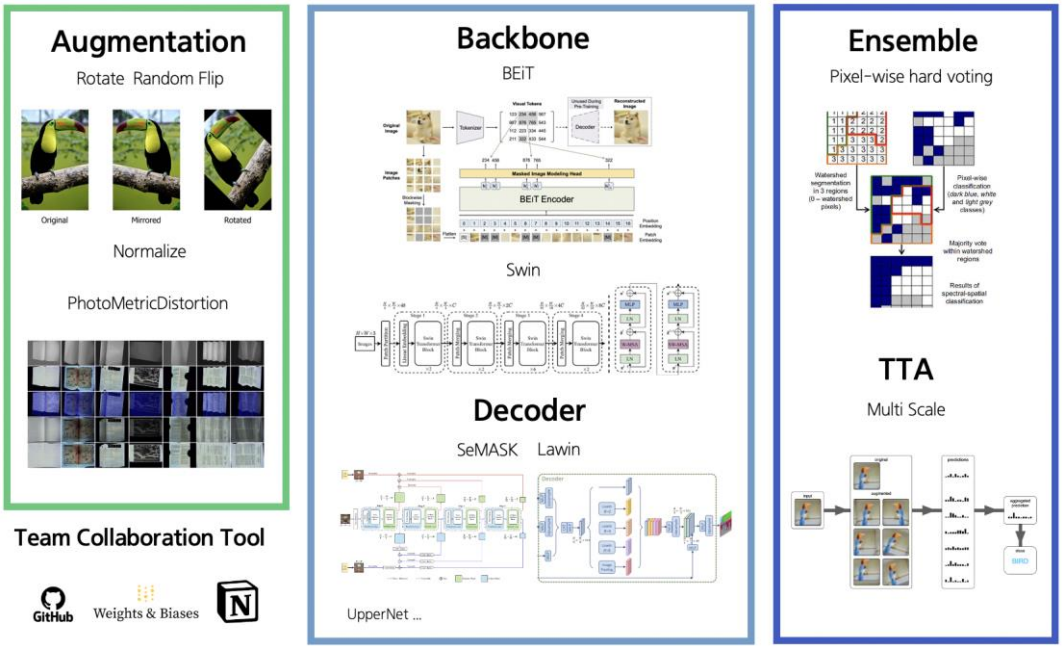
OS : Ubuntu 18.04

개발툴 : Vscode, Jupyerlab

협업툴 : Github, Slack, Notion

라이브러리 버전 : Python 3.6, Pytorch 1.7.1

○ 프로젝트 구조 및 데이터 셋의 구조도



○ 기대 효과

- Semantic Segmentation 의 전반적 과정 경험
- Semantic Segmentation Sota 모델 경험
- 다양한 Backbone, neck, ensemble 등 empirical 한 경험

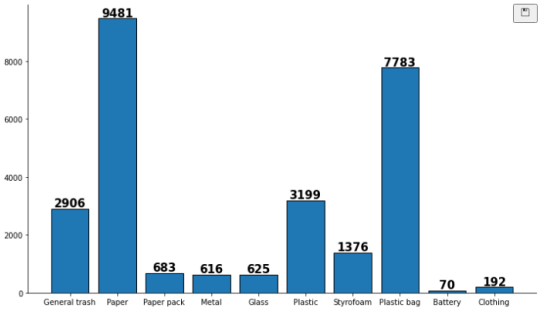
프로젝트 수행 절차 및 방법

○ 날짜 별 프로젝트 수행 절차 및 방법

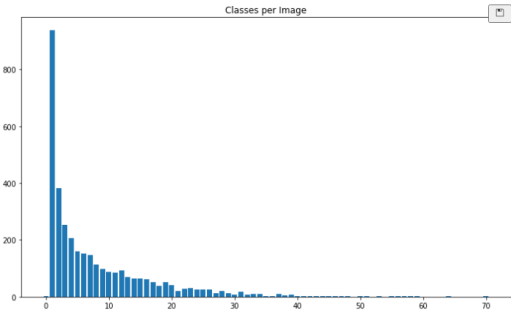
업무	업무 담당자	세부업무	0425 ~0427	0427 ~0429	0429 ~0501	0501 ~0503	0503 ~0505	0505 ~0507	0507 ~0509	0509 ~0512
EDA	김하준	Data Visualization, 전처리 위한 EDA								
	심준교	Data Visualization, 전처리 위한 EDA								
	송민수	Data Visualization, 전처리 위한 EDA								
	이창진	Data Visualization, EDA 통합본 생성								
데이터전처리	김하준	Outlier 제거, 이상치 제거								
Baseline 코드	전영우	베이스라인 코드 실험								
EXT	김하준	Pseudo Labeling								
	이창진	자체 Baseline 구현								
	송민수	이상치 제거 성능 검증								
	심준교	데이터 셋 정리 및 추가, Inference code, 이상치 제거 성능 검증								
	전영우	데이터 포맷 변환 및 파이프라인 파일 생성								
	유승리	TTA(multi scale)								
모델 검증	이창진	UPerNet, Deeplabv3, UNet 모델 실험								
	송민수	OcrNet,Lawin 실험								
	유승리	SeMask, UPerNet 실험								
	심준교	BEiT								
	김하준	BEiT								
	전영우	Backbone, Scheduler, Loss 등 실험/knet 실험								
앙상블	심준교	Pixel wise hard voting								

프로젝트 수행 결과

○ 탐색적 분석 및 데이터 전처리(EDA) - 학습 데이터 소개



<각각의 category 별 개수분포>



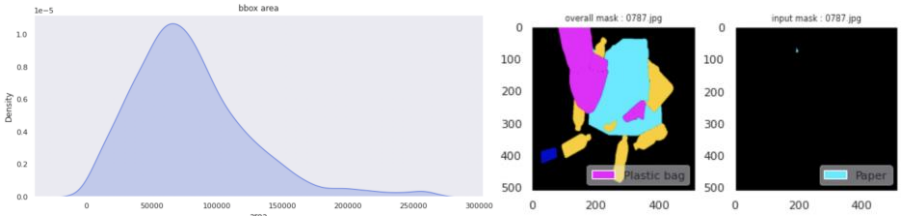
<이미지당 category 개수 분포>

<각각의 category 별 area 분포>

○ 데이터 전처리

● Outlier 제거

i. Area 분포에 따른 작은 area 삭제



EDA 를 진행하였을 때, 특정 annotation 의 area 가 1~30 까지 상당히 작은 annotation 이 존재하는 것을 확인하였다. 따라서 이를 모델이 확인하기에는 어려울 것이라고 판단하였고, 이를 제거하여 학습을 진행해보았으나, 결과는 좋게 나오지 않았다.

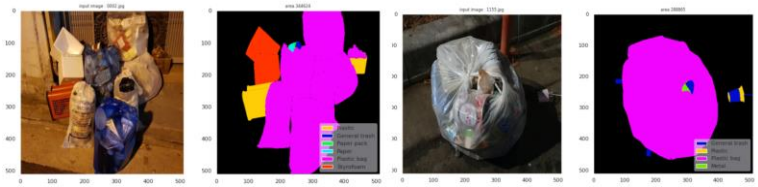
각 이미지는 큰 mask 를 먼저 생성하고, 그 위에 작은 mask 를 덧대어 붙인다. 이 과정에서 큰 mask 는 작은 mask 단위로 쪼개어 지는 경우가 종종 발생하였고,

이러한 mask 를 학습에서 제거하면 성능이 떨어질 수 밖에 없었다. 또한 픽셀단위로 classification 을 진행하므로, 모델이 못잡아 낼 것이라는 생각도 우리의 눈 관점이었다. 또 다른 가설이었던 annotation 실수 설은 전수 조수 결과, 전체 데이터셋에서 2~3 개 정도에 불가했다.

ii. data 전수조사에 따른 annotation 수정

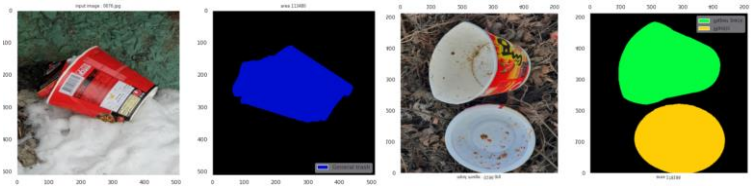
dataset 중에서도 몇몇 data 들의 annotation 이 잘못되어 있을 수도 있다고 판단하였다. 그에 따라 모든 data 들에 대한 전수조사를 진행하였고, 총 4 가지 경우의 annotation 오류를 확인할 수 있었다.

1) 전반적인 minor 실수



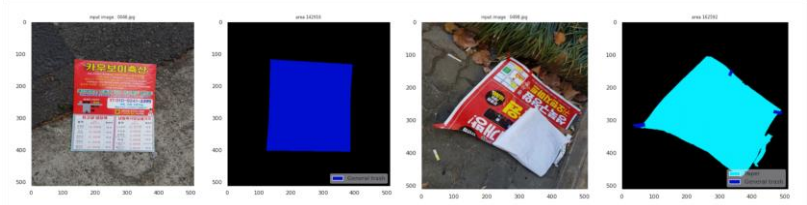
플라스틱 컵 뚜껑이 general trash, 컵 홀더가 general trash 로 구분되어 있는 등 성능에 영향을 끼칠 수 있는 mislabeling 된 data 를 수정

2) 컵라면 컵 분류



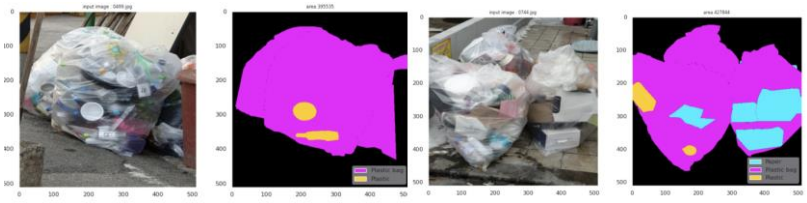
컵라면 컵 annotation 의 절반은 general trash 절반은 plastic bag 으로 확인되었다. 데이터 두개 정도 수정하여 성능 변화를 확인한 결과 general trash 로 확인되었다. (분리수거 기준에서도 일반 쓰레기로 확인되었다.)

3) 명함, 전단지 분류



명함과 전단지 annotation 의 절반은 general trash 절반은 paper 로 확인되었다. 데이터 두개 정도 수정하여 성능 변화를 확인한 결과 general trash 로 확인되었다. (분리수거 기준에서도 일반쓰레기로 확인되었다.)

4) 비닐 봉투 annotation 통일



대부분의 plastic bag(반투명)은 내부 물체의 투시 여부와 상관 없이 전체 plastic bag 처리되었으나 몇몇은 내부의 몇 물체가 annotation 되었다. 해당 부분을 수정하였다.

miss labeling 처리는 다음의 4 가지 기준과 애매한 경우(ex. 음식물이 묻은 경우 등)는 넘어가는 엄격한 기준에 따라 약 50 개의 annotation 을 수정하였다. 해당 miss labeling 을 수정 후 성능을 확인한 결과 큰 폭의 하락이 확인되었고, 이에 따라 기준이 에매하였던 비닐 봉투를 제거하고 성능을 재확인하였음에도 동일 현상이 확인되어 miss labeling 처리를 전면 중단하였다.

다음의 두가지 이유를 생각해볼 수 있을 것 같다.

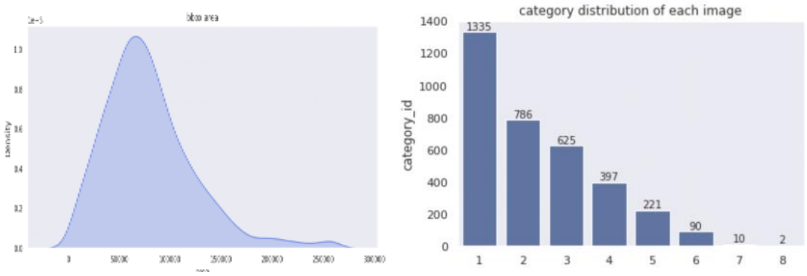
1) test 셋에서도 annotation 에 대한 기준이 불명확했을 경우
2) 우리의 예상보다 더 정밀한 annotation guide line 이 존재하였으나 눈치채지 못한 경우

- Train, validation set

  i. Stratified group K fold

  기존 train, validation set 이 제공될 때 stratified group k fold 로 데이터가 나눠져 제공되었다.

  ii. Multi-label stratified group K fold



EDA 결과를 기반으로 한 팀원간 논의 결과, 각 이미지 별 mask 가 차지하는 비중과 각 이미지 별 등장하는 category 의 수의 비율을 train 과 validation 에 적절하게 나눠줘야 한다는 결론을 내렸다. 이에 따라 multi-label stratified group K fold 를 구현하여 총 데이터 카테고리의 비율, mask 의 비율, category 수의 비율을 적절하게 나눠서 실험을 진행하였다. 하지만 성능에 유의미한 변화를 확인하기 어려웠고, 오히려 떨어지는 경우도 목격되었다.

원인은 다음의 2 가지로 유추 된다.

1) 각 이미지의 annotation 은 큰 마스크가 여러개로 쪼개진 경우가 많다. 이에 따라 하나의 물체임에도 여러 물체로 오인할 수 있다. 이에 따라 각 이미지에서 등장하는 동일 카테고리의 경우 한 카테고리 처리하였다. 이를 통해 multi-label 의 수를 최소화하는 효과도 얻을 수 있었으나 실제 이미 검수과정에서 확인 했듯이 각 이미지에는 동일한 카테고리의 물체가 여러번 등장하는 경우도 많기때문에 적절한 기준이 아니었을 수 있다. 대략적으로도 반영하는 것이 효과적일 것이라 생각하였으나, 이에 대한 근거도 충분하지 않다.

2) 각 이미지의 mask 크기의 비율을 맞춰주는 것이 정말 성능이 영향이 있는지에 대한 선행 확인이 없었다. 결국에는 실험을 통해 확인할 수 밖에 없는 영역이었고 이번 구현을 통해 확인했다고 생각할 수 있다. (물론 category 수를 같이 반영했다는 점에서 영향이 있을 수 있으나, category 수, 이미지 마스크 크기를 랜덤으로 나눈 것과 성능이 비슷하다는 점에서 유의미하지 않다고 볼 수 있다.)

○ 모델 선정 및 실험 과정

- Model

	Backbone	Neck	Optimizer	Scheduler	특이사항
lawin	swinL	NaN	AdamW	CosineAnnealing	
UperNet	Swin-L	-	AdamW	Poly, CosineRestart	hybrid loss (cross entropy + dice loss)
SeMask	SeMask Swin-L	FPN	AdamW	Poly, CosineRestart	official mmsegmentation 에서 제공 X
beit	beit	feature2pyramid	AdamW	CosineAnnealing	
knet	resnet, swin-L	-	AdamW	Poly, CosineAnnealing	val score 와 test score 의 차이가 큰 편

- Pseudo Labeling
  각 모델 별 가장 성능이 좋았던 모델로 test 셋 이미지에 pseudo labeling 을

진행하였고, 유의미한 성능 향상을 확인 할 수 있었다. 다만, ensemble 한 결과, 즉 가장 성능이 좋은 결과로 pseudo labeling 을 진행 하였으면 성능 향상이 어땠을지에 대한 의문이 여전히 남아있다. 다만, 이는 각 모델의 다양성이 줄어드는 결과를 가져왔을 염려가 있다.

- TTA

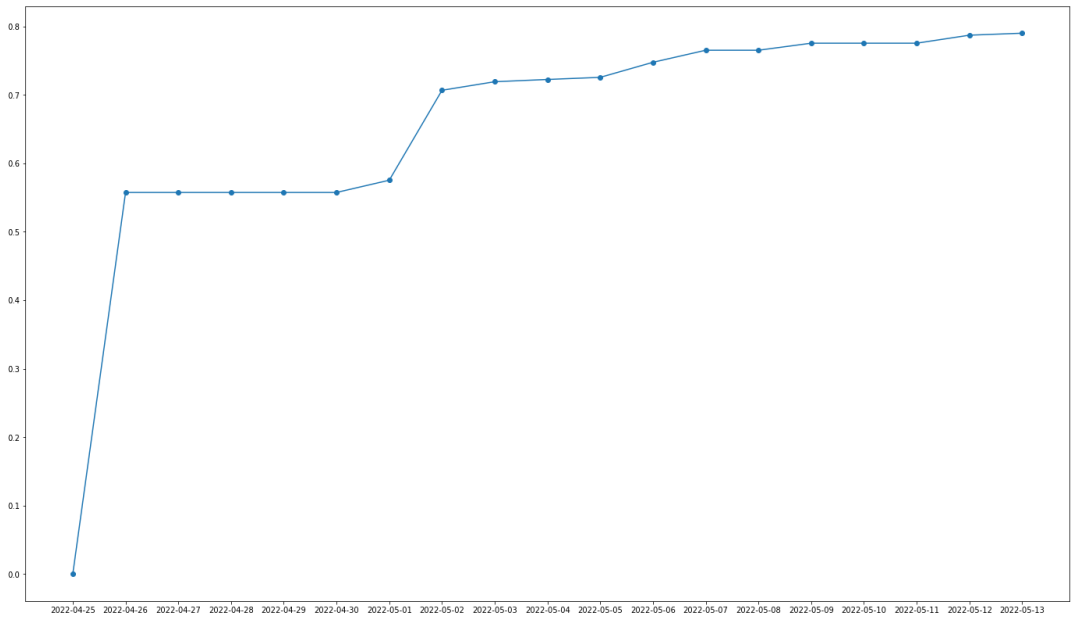
  Test time 에서 scale 의 ratio 를 [0.75, 1.0, 1.25, 1.5]로 지정하여 multi scale augmentation 을 수행한 결과, LB score 기준 0.01 의 성능 향상을 확인할 수 있었다.

○ 앙상블

최종적으로, 좋은 결과가 나왔던 모델들을 모아 앙상블을 시도하였다. 이때, hard voting 방법으로 앙상블을 진행하였는데, 각각의 pixel 에서 model 들이 예측한 class 중 가장 많이 예측한 class 를 이용하되 동점이라면 성능이 좋았던 모델부터 예측한 class 가 예측한 후보중에 있다면 이를 result 로 선택하는 방법을 택하였다.

다음과 같은 앙상블 기법을 도입한 결과, 최종적으로 LB score 0.7901 을 달성할 수 있었다.

○ 날짜 별 LB score 변화도



○ 최종 점수

Public mIoU:0.7901 (10 위) / Private mIoU:0.7230 (15 위)

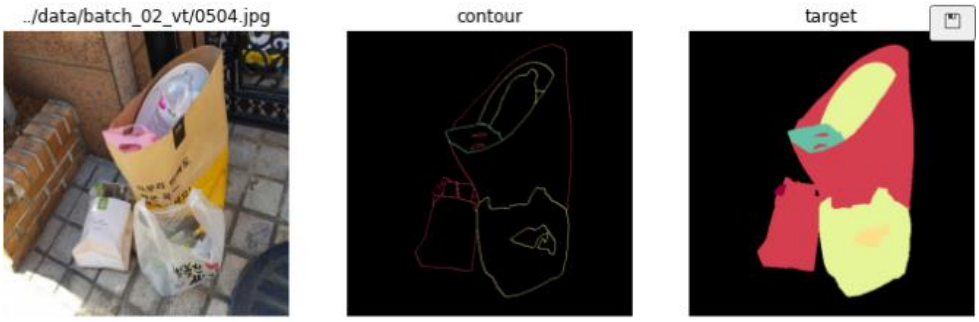
자체 평가	

의견	○ 잘한 점 ● issue 기반으로 Github 을 통하여 활발한 협업을 진행한 것 ● 초반부에 다양한 모델들을 팀원별로 나누어 실험을 진행해본 것 ● Wandb 를 사용하여 wandb 웹에 로깅하여 팀원 모두의 학습과정을 기록한 것 ● pseudo labeling 같은 추가적인 학습방법을 통하여 성능향상을 해보려 한 것 ● 다양한 transformer 기반 외부 모델들을 사용해 보았다는 점 (lawin, semask) ● 성능하락과는 별개로 팀원 모두 즐거웠다는 점 ○ 시도했으나 잘 되지 않았던 점 ● 그전부터 해보고 싶던 multi-label stratified group k fold 를 나름의 가설과 함께 성능을 검증해보았으나 유의미하지 않았던 점 ● data 를 살펴보며 잘못된 annotation 을 발견하였고, 추가적으로 결측치를 변경하거나 삭제하여 다시 학습시켜 보았으나, 성능이 오히려 떨어졌던 점 ● mmsegmentation 에서 제공하는 모델 외 외부 pre-trained 된 모델을 불러왔음에도 모델을 일반화 하지 못한점 ○ 아쉬운 점 ● transformer 기반 모델의 성능에 대한 맹신으로 다양한 backbone 기반 모델들을 실험하지 못했고 이는 private LB 에서 큰 폭의 성능하락으로 이루어졌다. 다양한 모델을 통해 일반화 하는 것은 기본 이나, 간과하고 넘어갔다는 점이 아쉽다. ○ 프로젝트를 통하여 배운 점, 시사 점 ● 항상 느끼지만, 제대로 된 협업이 모든 프로젝트의 초석이 된다는 것 ● 다양한 모델/환경 하에서의 실험이 앙상블 시 성능의 향상을 크게 불러온다는 것

개인 회고

캠퍼명	송민수_T3113
P Stage 동안 진행한 일	• **EDA 진행** 개인적으로 이미지들을 살펴봄과 함께 data 의 분포가 어떤지, 다른 추가적으로 봐야할 사항이 있는지 살펴보기 위하여 EDA 를 진행하였다. 첫 번째로, 각각의 category 가 얼마나 분포되어 있는지를 파악하였다. paper 가 9481 장으로 가장 많았으며, battery 가 70 장으로 class 간의 imbalance 문제가 상당하다는 것을 알게되었고, 추가적으로 crop&paste 와 같은 방법도 시도할 수 있을 것이라고 생각하였다.

두 번째로, area 의 분포에 대하여 알아보았다. 단순히 area 를 나열해 보았을 때, 1~10 까지 아주 작은 area 들이 존재한다라는 것을 확인하였고, 이를 결측치로 여기고 뺀다면 성능향상이 존재하지 않을까 생각하였다.
세 번째로, train data 를 semantic segmentation 형태로 시각화 하였다. 이는 실제로 어떻게 annotation 이 되어 있는지 확인할 수 있었고, 추가적으로 결측치를 확인하고 여러가지 가설을 생각해 볼 수 있었다.



- **ocrnet**
  제일 처음 mmsegmentation 에 대한 실험으로 ocrnet 을 선택하여 진행하였다. 그러나 Adam 을 사용하지 않고 SGD 를 사용한 점, pre-trained 된 model 을 가져왔지만 오류로 적용되지 않은 점 등등 여러가지 요인으로 인하여 validation 점수가 2%대로 매우 저조하였다. 이는 다른 캠퍼님의 같은 모델로 나온 성능으로 판단할 수 있었다. 여기서 여러가지 optimizer 나 scheduler 등을 고쳐서 성능을 내 볼 수도 있었지만, lawin 의 실험을 진행하여 더는 진행하지 못하였다.
- **lawin**
  두 번째로 2022 년에 나온 lawin 이라는 모델을 통하여 실험을 진행하였다. lawin 을 사용한 이유는 transformer 구조를 사용하여 큰 성능향상을 기대할 수 있고, 또한 lawinASPP 라는 Spatial Pyramid Pooling 에서 착안한 decoder 로 여러 scale 에서의 특징을 이용하여 조금 더 정확한 segmentation 이 가능하도록 하였기 때문이다. mmsegmentation 에는 lawin 이 없었기 때문에, 해당 github 에 있는 issue 에서의 log 를 참조하여 구현하였고, swinL 을 backbone 으로 하였을 때, LB 점수 0.76 이라는 높은 점수를 얻을 수 있었다.
- **mlflow 연결**
  실험을 하던 중간에, 이를 기록하고 전체적인 학습결과를 알고싶어서, 여러 logger 중 mlflow 를 적용시켜보기로 하였다. mmdetection 에서는 mlflowhook 이 있어서 바로 사용할 수는 있지만, 이는 기본 포트 5000 과 서버에서 특정 ip 로만 접속이 가능하였다. 따라서, mmcv 에서 mlflow.set_registry_uri()를 통하여 localhost 와 함께 특정 포트를 열어주어서 로컬 데스크탑에서도 확인이 가능하게 하였다.
  추가적으로, 다른 ip 에서 실험을 진행한 뒤 logging 하는 방법도 알았으나, mmcv 에서의 내용수정과 더불어 고려할 사항이 많아서 중간에 wandb 로 넘어가기로 하였다.
- **mislabeling 추가 가설 실험**

	하준 캠퍼님이 모든 data 를 살펴보았을 때, 컵홀더가 general trash 로 구분되어 있다던가와 같은 mislabeling 된 annotation 들이 존재하였고, 이에 따라 이를 삭제하거나, annotation 을 변경하여 실험을 진행하였었다. 그러나, 이 중 plastic bag 내부의 object 를 plastic bag 로 수정하여 실험을 진행하였었는데, 본인은 그렇게 실험을 진행한다면, model 이 내부의 object 를 잡더라도 plastic bag 라고 잘못 예측할 수 있다고 판단하여, plastic bag 에 관한 내용을 삭제하고 실험을 진행해 보았다. 그 결과 모두 수정하였을 때 보다 2%정도의 성능 향상을 이룰 수 있었다.
느낀점	• **Semantic Segmentation task 경험** 이번 task 에서는 실제 Semantic Segmentation 분야의 task 를 직접 경험해보고, 모델 학습 및 평가를 진행함으로써, Semantic Segmentation 학습의 전반적인 파이프라인을 겪어보았다. 이러한 경험을 살려서 이후에 Kaggle 이나 Daycon 에서도 이와 관련된 Competition 을 진행해 볼 수 있을 것 같다. • **적은 모델 활용** 실제로 대회가 끝나고 나니, 본인이 실험했던 model 은 두 종류밖에 존재하지 않았다. 물론 두 모델로 여러가지 실험을 진행하였었지만, 다양한 모델을 활용하였다면 이후에 더 다양한 output 으로 앙상블을 할 수 있지 않았을까 생각한다.

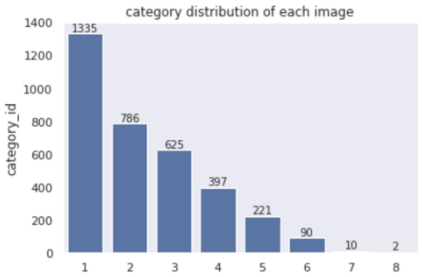
캠퍼명	**심준교_T3124**
P Stage 동안 진행한 일	- BEiT 실험 진행 및 논문 발표 mmsegmentation 을 통해 BEiT-UperNet 모델 실험을 진행하였고, 피어세션에서 BEiT 논문을 발표하였다. - Inference&Ensemble 코드 작성 mmsegmentation 으로 학습한 config 파일과 모델을 불러와 실제 AI Stages 에 제출하기 위한 inference 파일을 작성하였다. 그 과정에서 valid set, test set 각각 원본 이미지와 모델이 예측한 결과를 시각화할 수 있도록 하여 에러 분석을 진행하였다. 또한 제출한 csv 파일을 가져와 앙상블할 수 있는 코드를 작성하였고, 역시 시각화 코드를 추가하여 앙상블 시 결과가 얼마나 바뀌었는지를 확인할 수 있도록 하였다. - labeling 오류 확인 투명한 비닐봉투 안에 있는 물체들을 개별 클래스로 분류하는 데이터들이 train set 에 존재하는 경우가 꽤 있었다. 따라서 실제 test set 의 정답에는 해당 경우를 어떻게 분류하였는지 알아보기 위해 제출 결과를 달리해보며 labeling 오류에 해당하는 케이스를 분류하였다.

느낀점	- 데이터의 중요성 trainset 와 validset 을 다르게 구성하거나, mislabeling 된 annotation 을 수정하였을 때 큰 폭은 아니었지만 성능 향상이 일어났다. 앞선 대회에서도 느꼈던 annotation 의 일관성과 데이터 분포의 중요성을 다시 한번 느낄 수 있었다. - Self-supervised learning BEiT 모델에 대해 공부하면서 self-supervised learning 을 통해 사전 학습하고 이를 downstream task 에 적용한다는 것이 매우 인상적이었다. 일관성 있게 라벨링 된 데이터도 중요하지만, self-supervied learning 은 사전 학습 시 라벨이 필요없기 때문에 앞으로도 계속 핫한 주제일 것이라는 생각이 들었다. 다만 BEiT 이후에 나온 MAE(Masked Autoencoder) 모델을 mmsegmentation 에서 적용해 봤을 때 성능이 생각보다 잘 나오지 않아 아쉬웠다. - 리더보드형 대회 세 번의 P-stage 에서 캐글, 데이콘과 같이 리더보드형 대회를 경험하며 각 task 뿐만 아니라 대회 전반에 관한 경험을 하고 팁을 얻을 수 있었다. 다음에 데이콘에 참여하면 예전보다는 더 좋은 성적을 거둘 수 있을 것 같다는 생각이 들었다.

캠퍼명	**이창진_3169**
P Stage 동안 진행한 일	- 자체 baseline 을 구축하고 smp 라이브러리로 실험 진행. 이전 Pstage 에서 소개됐던 victoresque 템플릿을 커스터마이징하여 coco dataset 에 맞게 dataloader 수정 및 복잡한 상속 구조를 간소화하여 빠른 실험을 할 수 있게 하였다 - U-Net, Deeplabv3, U-Net++ 등을 사용하여 실험. - mmsegmentation 라이브러리를 사용하여 upernet, segformer 등 다양한 모델들을 실험
느낀점	• 복잡한 segmentation 를 밑바닥부터 직접 구현하면서 더 깊이 있는 이해를 할 수 있었다. 데이터 전처리, 모델링, augmentation, 등 end-to-end 로 경험할 수 있어서 좋았다. • Image classification 같은 상대적으로 쉬운 task 는 자체 베이스라인으로 좋은 성능을 낼 수 있었다. 하지만 object detection, segmentation 과 같은 상대적으로 훨씬 복잡한 task 는 자체 베이스라인이 좋은 성능을 내지 못하였다. 많은 실험을 진행했음에도 불구하고 끝내 mAP 가 60 을 넘기지 못한게 많이 아쉬웠다. 이유를 추측하면 수백줄이 넘는 코드에서

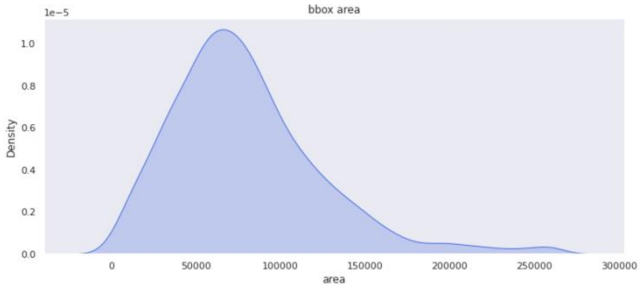
	단 한줄이라도 로직에러가 발생하면 전체 로직이 잘못되기 때문에 미처 찾아내지 못했던 에러들이 좋은 성능을 내지 못하게 한게 아닌가 생각한다. 이를 해결하기 위해서는 다른 좋은 라이브러리들의 소스 코드를 심도있게 분석하고 좋은 practice 를 많이 참고해야 할 것 같다. • mmsegmentation 라이브러리로 실험했을때는 epoch 1 번만에 mAP50 이 넘었다. 단지 모델링만이 중요한것이 아니고 전체 파이프라인을 구성하는 코드들도 성능에 지대한 영향을 미칠 수 있다는것을 배웠다.

캠퍼명	**김하준_T3066**
P Stage 동안 진행한 일	**데이터 시각화** 해당 task 는 데이터 클렌징, 같은 카테고리 내에서도 다양한 물체의 형태 등을 파악하기 위하여 각 상황에 맞는 데이터 시각화가 필수라 생각하였다. 이에 따라 아래의 4 가지 방법으로 시각화 하였다. 각 방법은 상황별로 활용되었다. 1) 원하는 indicies 의 이미지와 해당 mask 출력 - 빠른 시각화 2) 이미지 별 segmentation mask 크기 오름차순, 내림차순 출력 - outlier 제거에 사용 3) 카테고리 별 출력, (해당 카테고리의 mask 크기 오름차순, 내림차순 출력) - 각 카테고리 등장 물체 형태 파악 4) 각 Image 별 segmentation mask annotation 개별 출력 - 데이터 클랜징을 위해 annotation id 파악을 위해 사용 **Multi-label stratified group k fold** 해당 task 는 이미지 별 segmentation mask 의 크기가 큰 차이가 있고, 등장하는 카테고리 및 물체의 수 차이가 크다. 이에 따라 데이터를 다음의 기준으로 나누었다. 1) 카테고리 2) image 별 마스크 크기 3) image 별 등장하는 카테고리 수 처음 계획은 각 image 별로 등장한 모든 annotation 의 수를 고려하고자 하였으나, 해당 경우 경우의 수가 너무 많아지며 동시에 여러 mask 가 겹쳐지는 현상으로 인해 동일 물체에 대해 여러 마스크가 생성될 수 있는 상황을 고려하여 한 이미지에서 등장한 동일 카테고리 annotation 들에 대해서는 하나의 물체로 처리하였다 .



그 결과 각 이미지 당 등장한 카테고리의 수의 분포는 다음과 같다. 물론 이 방법은 정확하지 않지만 아예 고려하지 않은 것보다는 나을 것이라 생각하였다.

또한 각 이미지 별 mask 크기를 구하기 위하여 이미지에 등장한 모든 annotation 의 area 크기를 더했다. 하지만 이 결과는 기대와 다르게 이미지 크기를 넘어서기도 하였다. 이는 각 마스크 끼리 겹치는 경우가 있기 때문이다. 이에 따라 한 이미지 내에서 겹치는 부분을 제거하고 area 를 다시구했다. 결과는 아래 분포와 같다.

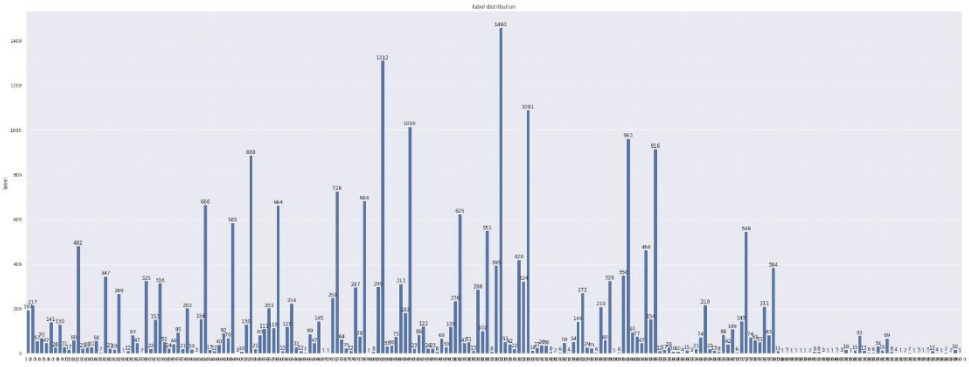


전체 area 크기를 기준으로 3 등분 하여, Small, Medium, Larg 처리하였다.

(총 카테고리 수 [1, 10]) 10 * (이미지 별 등장 카테고리 수 [1, 8]) 8 * (mask 크기 [1, 3]) 3 = 240 개의 label 을 생성하였다. 이 때 각 label 로 최종 label 을 만들기 위한 수식은 아래와 같다.

```
for a,b,c in zip(df["category_count"], df["area_size"], df["category_id"]):
    label.append( 30*(a-1) + 10*(b-1) + c)
```

이로 인해 생성되는 240 개 label 에 대한 분포는 아래와 같다.



최종적으로 데이터가 잘 나눠졌는지 확인하기 위해 본 카테고리 별 분포는 다음과 같다.

	General trash	Paper	Paper pack	Metal	Glass	Plastic	Styrofoam	Plastic bag	Battery	Clothing
training set	10.79%	35.20%	2.54%	2.29%	2.32%	11.88%	5.11%	28.90%	0.26%	0.71%
train - fold4	10.79%	35.96%	2.57%	2.29%	2.48%	11.51%	5.07%	28.36%	0.22%	0.75%
val - fold4	10.77%	32.21%	2.39%	2.28%	1.69%	13.35%	5.27%	31.06%	0.41%	0.57%
train - fold4	10.86%	34.76%	2.61%	2.36%	2.19%	12.30%	4.98%	29.01%	0.25%	0.68%
val - fold4	10.50%	36.97%	2.23%	1.99%	2.85%	10.21%	5.62%	28.46%	0.31%	0.86%
train - fold4	10.75%	34.56%	2.54%	2.24%	2.27%	12.19%	5.26%	29.19%	0.25%	0.74%
val - fold4	10.96%	37.86%	2.52%	2.48%	2.53%	10.58%	4.48%	27.71%	0.30%	0.59%
train - fold4	10.87%	35.01%	2.58%	2.23%	2.23%	11.75%	4.98%	29.39%	0.29%	0.68%
val - fold4	10.49%	35.99%	2.35%	2.52%	2.68%	12.40%	5.62%	26.97%	0.13%	0.85%
train - fold4	10.68%	35.75%	2.37%	2.32%	2.44%	11.64%	5.25%	28.55%	0.29%	0.72%
val - fold4	11.25%	33.03%	3.20%	2.17%	1.84%	12.84%	4.54%	30.30%	0.15%	0.69%

그러나 성능은 기대와 다르게 일정 수준 이상으로 나오지 않았다.
사실 segmentation 대회는 픽셀별로 classification 을 하는 것이기 때문에 한 이미지 당 mask 의 크기와 카테고리 등장 개수가 성능에 정말 유의미한 영향이 있는지는 모른다. 단지 유사한 분포를 최대한으로 맞춰줘서 안정적인 성능을 기대한 실험 정도로 이해해도 좋을 것 같다.

김하준_T3066	0.6129 → 0.5841
김하준_T3066	0.6091 → 0.5822

나의 경우 public private set 에서 모두 약간의 성능 향상이 있었으나 팀원들의 실험 결과 성능이 하락되었다고 확인되어 실제로 사용하지는 않았다.

데이터 전수 검사 및 의사결정
Augmentation 을 진행하기 위하여 데이터를 살펴보다가 annotation 이 일정하게 되지 않은 사례를 너무 많이 발견하여 전수조사를 하게 되었다. Wrap up report 에 있는 내용이기에 간단하게만 다뤄보자면, 확실한 miss labeling 외 annotation 기준이 달라 비교가 필요한 데이터셋들은 submission file 에 수정을 가해 LB 성적을 확인하였다. 위 과정을 걸쳐 annotation 을 수정하고 제출하였으나 오히려 성능이 하락하였다. 다음의 두가지 이유를 의심해볼 수 있다.

1) Test dataset 에도 annotation guide line 이 일정치 않아 노이즈가 많은 경우
2) 파악에 실패한 정교한 annotation guide line 이 따로 존재하는 경우

Pseudo Labelling
학습한 모델로 inference 한 Test 이미지의 label 을 재활용하여 학습한 후 결과를 확인했다. 유의미한 성능 향상이 있었다. 다만 각 단일 모델별로 inference 한 label 을 재학습에 사용하였는데 ensemble 하여 얻은 최고의 성능 label 로 재학습을 해보았으면 어떨까 싶다.

BEiT Large 실험
BEiT 는 Bert 의 학습 방법을 이미지에 적용하여 self-supervised-learning 을 한 모델이다. 기대 만큼이나 성능이 잘 나와줬다. 간단히 논문을 살펴보고 팀원의 리뷰까지 들어볼 수 있어 모델에 대한 어느정도 이해도 있다. 다만 좀더 정확하게 bert 기반 학습이 segmentation 에 어떻게 잘 적용되었는지에 대해서는 좀 더 살펴볼 필요가 있다. 아쉬운

	점으로는 Bert Large 가 base 모델보다 성능이 나빴다는 점이다. parameter 개수가 2 배 많아 성능의 향상을 기대했는데 base 와 동일한 hyperparameter 세팅에서도 성능이 상당히 낮았다. 학습을 더 오래 해보기에는 이미 오버 피팅 현상이 나타났었다. 첫 대회때도 그랬지만 학습하고자 하는 문제의 복잡도에 따라 모델의 크기를 결정하는 것이 중요한 것 같다. Segmentation task 는 상당히 복잡한 문제라고 생각 했었지만 어디까지나 내 추론이기에 좀 더 확인이 필요할 것 같다.
느낀점	대회에 적용해볼 수 있는 기술은 너무나 많은데 주어진 시간은 부족하다. 구현 능력을 키워야 하고, 우선적으로 기존 대회에서 유의미했던 실험들을 먼저 해보는 것이 중요한 것 같다. 가설을 충분히 세우고 검증하는 과정이 이번에 별로 효과적이지는 않았지만, 덕분에 절차대로 실험해볼 수 있었고, 앞으로도 계속 고수하고 싶은 방법이다. 기존 사람들이 하지 않았던 방법의 시도도 때로는 필요하지만, 이는 해볼 수 있는 것을 모두 해보고 난 후에 하는 것이 효율적일 것 같다. 첫 대회, 두 번째 대회에서 더 진취적으로 해보고 싶다는 아쉬움이 많이 남았었는데, 이번 대회는 성능은 아쉽지만 여러가지 구현해볼 수 있었다는 점에서 즐거웠다.

캠퍼명	유승리_T3129
P Stage 동안 진행한 일	● **mmsegmentation을 이용한 UPerNet 실험 수행** 우선, mmsegmentation 에서 제공하는 UPerNet 에 대한 실험을 수행하였다. 처음에는 backbone 으로 ImageNet pretrained Swin-L 을 불러올 때 key error 가 발생하였으나, swin2mmseg.py 를 이용하여 convert 를 수행한 후에는 정상적으로 실험을 진행할 수 있었다. 실험 시에는 제공된 train / val / train_all set 을 사용했으며, optimizer 로는 AdamW(lr: 6e-5)를 사용했다. lr scheduler 로는 poly 와 cosine restart 를 실험해보았는데, poly 를 사용했을 때의 성능이 더 좋았다. loss 로는 주로 cross entropy 를 사용했으나, cross entropy : dice loss 를 1:3 과 3:1 로 조합한 hybrid loss 로도 실험해보았다. 하지만 성능은 일반 cross entropy 가 가장 좋았다. 여러 가지 augmentation 도 적용해 보았는데, 성능이 오히려 하락하는 모습을 보였다. 마지막 날에 pseudo labeling 을 적용한 결과, LB mIoU 가 처음으로 0.77 이상을 기록하였다. ● **mmsegmentation에서 제공하지 않는 SeMask-FPN 추가 및 실험 수행**

	UPerNet 실험 중, 조사 과정에서 mmsegmentation 에서 제공하지 않는 SeMask-FPN 의 mmdetection 구현 소스코드를 발견했다. (https://github.com/Picsart-AI-Research/SeMask-Segmentation/tree/main/SeMask-FPN) 하지만 이를 서버에서 실행해보니 많은 오류에 부딪히게 되었고, SeMask 용 가상 환경을 따로 생성하고 mmseg 관련 코드들을 수정하는 등 며칠 간의 시도 끝에 train 시 발생하는 오류를 해결하였다. 그러나 여전히 inference 시에도 dataset 이 계속 shuffle 되는 등의 다양한 문제가 발생하여 다시 며칠 동안 트러블슈팅을 진행하였다. 이렇게 오류 해결 후에 남은 기간 동안 UPerNet 과 비슷하게 실험을 진행하였는데, SeMask-FPN 의 경우에는 lr 로 1e-4, lr scheduler 로 cosine restart 를 사용했을 때의 성능이 더 좋았다. 또한, 마찬가지로 마지막 날에 pseudo labeling 을 적용한 결과, LB mIoU 가 0.77~0.78 이상을 기록하였다. ● **TTA 적용** train 시 config 의 MultiScaleFlipAug 에서 여러 ratio 를 주면 transformer 에서 에러가 발생하는 문제점이 있어서 test 시에도 시도하지 않았으나, 대회 마지막 날에 시도하였을 때 에러 없이 성공하였다. multi-scale 의 ratio 로는 [0.75, 1.0, 1.25, 1.5]를 주었으며, SeMask-FPN 결과 중 LB mIoU 가 가장 높았던 full / pseudo labeling 모델에 대해 적용한 결과, LB mIoU 기준 0.01 의 상승을 확인할 수 있었다.
느낀 점	● **배운 점 및 좋았던 점** - mmsegmentation 에서 제공하지 않는 모델인 SeMask-FPN 을 적용하려 하다 보니 train, inference 과정에서 많은 에러들에 부딪혔는데, 포기하지 않고 해냈다. - 새로 추가한 SeMask-FPN 의 최종 private LB mIoU 가 팀 제출 결과 중 가장 높았고, 오히려 public LB mIoU 보다 상승했다. (0.7146 -> 0.7251) - git flow 를 이전 대회 때보다 잘 사용했다.

	● **한계점 및 아쉬운 점** - SeMask-FPN 을 적용해보기 위해 트러블슈팅을 진행하다가 시간이 많이 지체되었다. - 최종 private LB 결과, SeMask-FPN 을 제외하고는 점수와 등수가 크게 떨어졌다. - pseudo labeling 시, 성능이 가장 좋았던 모델의 inference 결과를 pseudo labeling 에 사용하지 못했다. - MultiScaleFlipAug 가 안되는 줄만 알고 다시 시도해보지 않았다가 마지막 날이 되어서야 시도했다. - 일정 상의 이유로 copy paste 등의 다양한 augmentation 을 적용해보지 못했다.

캠퍼명	전영우_T3192
P Stage 동안 진행한 일	**Baseline 실험** 대회를 시작하며 제공된 간단한 baseline 코드를 이용하여 실험을 진행했다. FCN, Deeplabv3 등 실험을 통해 가장 간단한 베이스라인 모델의 성능이 어느정도 나오는지 확인할 수 있었고, 이를 mmsegmentation 모델 실험 시 참고할 수 있었다. 또한 baseline 코드를 이용해 augmentation 에 대한 사전 실험도 진행하였는데, Image classification 이나 Object detection 에 비해 다양한 augmentation 적용 시 생각만큼 성능이 크게 오르지 않고 오히려 몇몇 augmentation 에 대해서는 성능이 하락하는 것을 확인할 수 있었다. 이를 통해 대회 진행 중 적용할 augmentation 에 대한 인사이트를 얻고, EDA 에 대한 중요성을 팀원들과 공유하였다. **데이터 파이프라인 생성** 대회에서 기본으로 주어진 데이터셋은 COCO format 의 json 파일로, mmsegmentation 에서 이를 활용하려면 적절하게 format 을 바꿔주는 과정이 필요했다. 토론 게시판에 있는 글을 참고하여 우리 팀에 맞게 코드를 작성하고, mmsegmentation 의 데이터셋 config 파일을 작성하였다. **Backbone/Scheduler/Loss 실험** 지난 대회 동안 조작변인/통제변인을 명확히 하여 실험을 진행하지 못했던 것이

	아쉬워서 이번 대회에서는 조금 늦더라도 이를 명확히 하여 차근차근 실험을 진행하였다. Backbone 으로는 resnet, convnext, swin-L 등을, Scheduler 로는 Poly, StepLR, CosineAnnealing 등을, Loss 로는 CE, Focal, Dice, 여러 개를 동시에 사용하는 Multi-loss 등을 실험해보았다. **knet 실험** 2020 년 발표된 논문의 모델인 knet 을 위주로 실험을 진행하였다. Validation score 상에서는 mIoU 값이 0.76 이상이 나올 정도로 좋은 성능을 보였지만, test score 에서는 그러지 못했다. 이에 k-fold, seed ensemble 등 다양한 시도를 해보았지만 명확한 해결책을 찾지는 못했다. 이에 모델 특성 때문일지도 모른다는 생각을 하였다.
느낀점	**배운 점 및 좋았던 점** - 이슈 기반 github 협업을 가장 활발하게 했던 대회였던 것 같다. 진행하면서 다양한 어려움에 직면하며 이를 해결하는 과정에서 실제로 협업이 어떤 식으로 이루어져야 할 지 배울 수 있었다. - 이번 대회에서는 통제변인/조작변인을 명확히 설정하여 하나씩 실험을 진행하였다. 이에 실험의 방향성을 잡을 수 있었고, 각 요소가 성능에 미치는 영향을 알 수 있었다. - Data cleansing, Pseudo labeling, TTA 등을 활용하여 모델 성능을 끌어올리려는 시도를 하였고, 실제로 해당 기법들이 어느정도의 영향을 주는지 파악할 수 있었다. **한계점 및 아쉬운 점** - 데이터를 많이 만져보지 못 한 것이 아쉽다. 다른 팀원 분이 data cleansing, pseudo labeling 등을 진행하는 과정에서 더 많은 도움을 드리지 못 해 아쉽다. 다음에는 직접 데이터를 만지는 경험을 중점적으로 시도할 것이다. - knet 모델의 mIoU 가 validation set 과 test set 에서 다른 모델들보다 큰 차이가 나는 이유를 명확히 찾지 못했다. 이를 찾으려 다양한 시도를 해보았지만, 그 차이가 완화되었을 뿐 비슷한 수준으로 align 하지는 못했다. BEiT, SEMASK 등의 모델들은 비교적 align 이 잘 되었는데, 모델의 특성 때문일지도 모르겠다는 생각을 해본다. - Image Classification 에서 focal loss 와 dice loss 등을 함께 사용하는 multi-loss 가 좋은 성능을 보였다고 하는데, 이를 이번 segmentation 대회에서 시도해보았지만 성능의 향상이 없었던 것이 아쉽다.