재활용 품목 분류를 위한
Semantic Segmentation

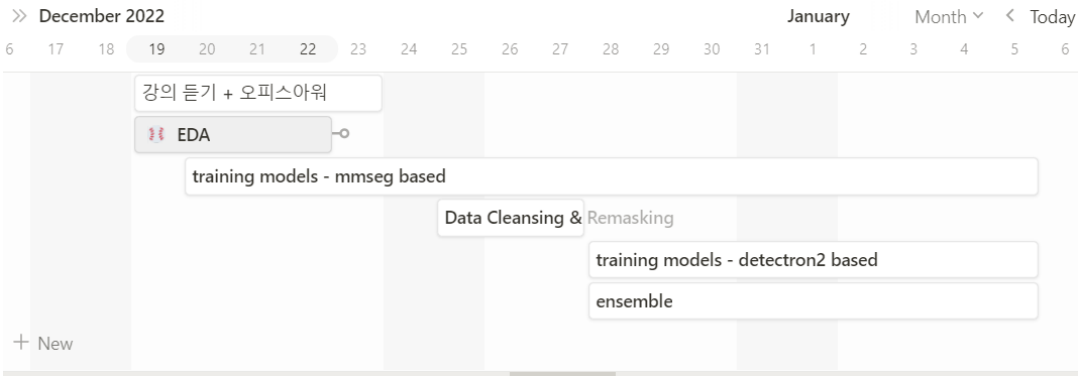
cv-16 비전 길잡이

1. Objective of the project

바야흐로 대량 생산, 대량 소비의 시대. 우리는 많은 물건이 대량으로 생산되고, 소비되는 시대를 살고 있다. 하지만 이러한 문화는 '쓰레기 대란', '매립지 부족'과 같은 여러 사회 문제를 낳고 있다.

따라서 우리는 사진에서 쓰레기를 Segmentation하는 모델을 만들어 이러한 문제점을 해결해 보기 위한 프로젝트를 진행하였다. 문제 해결을 위한 데이터셋으로는 배경, 일반 쓰레기, 플라스틱, 종이, 유리 등 11종류의 쓰레기가 찍힌 사진 데이터셋이 제공되었다.

2. 프로젝트 기간: 12월 19일 ~ 1월 6일



<그림 1. 주요 프로젝트 타임라인>

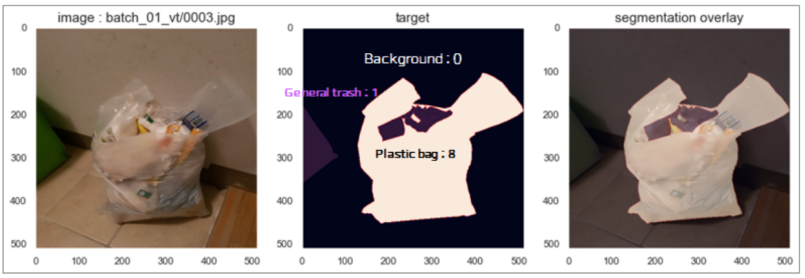
3. 팀 역할

- 박민지: SeMask 모델 학습, 하이퍼 파라미터 튜닝, pseudo-labeling, ensemble test
- 민 기: 다양한 모델(Upernet convNeXt, mask2former_swin 등) 학습, wandb sweep, CVAT 구현, stratified kfold by Area, data versioning,
- 최동혁: EVA, DiNAT, annotation manual, class_weights, ViT-Adapter fold-4
- 장지훈: ViT-Adapter 실험, data split & merge 구현, wandb logger 커스텀
- 유영준: 다양한 모델(HorNet, Swin Upernet, Senformer 등) 실험, 하이퍼파라미터(batch size, scheduler 등) 비교 실험, data 버전 비교 실험, ensemble 테스트

4. 프로젝트 수행 절차 및 방법

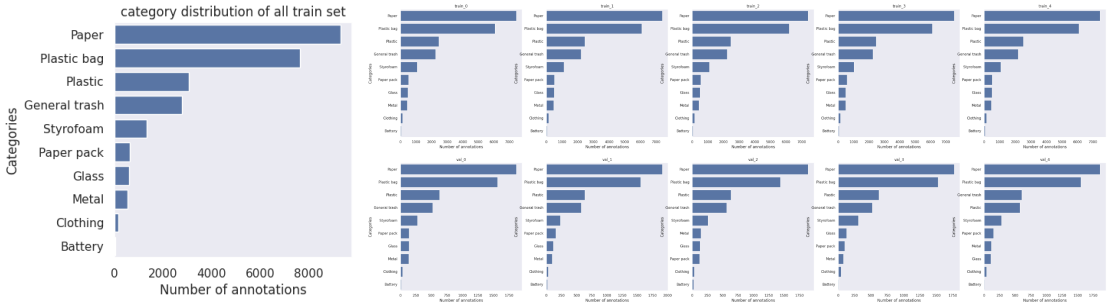
4-1. Data

- 이미지 크기 : (512, 512)
- 이미지 개수 : 학습 데이터 3272장, 평가 데이터 819장
- 11 클래스 : Background, General trash, Paper, Paper pack, Metal, Glass, Plastic, Styrofoam, Plastic bag, Battery, Clothing



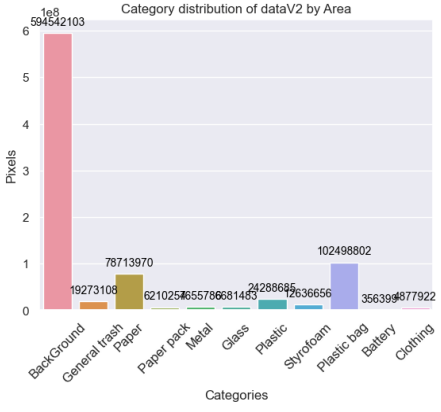
<그림 2. 학습 데이터 및 segmentation 예시>

4-2. EDA



<그림 3. train set 클래스 분포와 k-fold가 적용된 train, validation set의 클래스 분포>

전체 train set에 대한 클래스 분포는 그림 3과 같이 불균형이 매우 심하다. 특히 clothing, battery 클래스는 매우 적은 수를 보인다. 이러한 분포를 train set과 validation set에 동일하게 적용하여 평가 지표로 삼을 수 있도록 Stratified Group K-Fold (K=5) 를 사용하여 동일한 분포를 갖는 5쌍의 train, validation set을 구성하였다.



<그림 4. train set 클래스별 픽셀 개수>

대회 초반에는 그림 3과 같이 클래스 개수 분포를 이용하여 k-fold를 적용하였으나, segmentation의 task 특성상 픽셀별로 loss를 계산하므로 그림 4의 클래스별 픽셀 개수를 고려하여 k-fold를 새로 적용하였다.

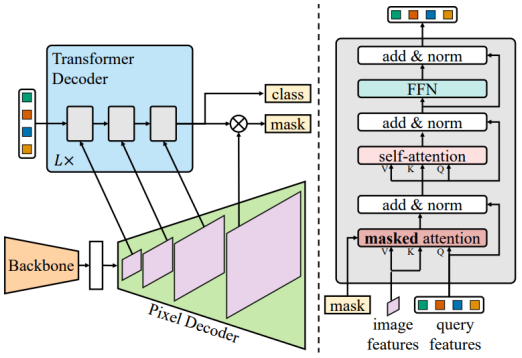
4-3. Model

Method	Backbone	Data Versioning			최종 제출
		data v1	data v2	data v3	
Mask2former	swinT			O	O
Mask2former	beit-adapter	O	O		O
SeMask	swinT		O		O
Upernet	Hornet		O		O
Upernet	swinT		O		O
Upernet	convNeXt	O	O		
Upernet	Beit (v1/v2)	O			

<표 1. 사용한 모델 도표>

4-3-1. Mask2Former

Mask2Former는 per-pixel classification을 활용하지 않고 mask 혹은 경계선 단위로 학습시켜 pixel 단위로 학습시킨 경우보다 오차가 줄어드는 장점이 있는 특징을 가지고 있다. Mask2Former는 Pixel Decoder에서 Deformable DETR을 활용하고, Transformer Decoder에서 DETR을 그대로 사용하되, Masked attention을 추가하여 Mask에 대한 정보를 class 정보와 함께 loss 계산에 반영하여 기존 Transformer 모델 보다 성능을 높일 수 있었다. 최근 SOTA 모델들은 Mask2Former 구조를 활용하는 경우가 많다보니 기본 Mask2Former를 앙상블 모델 중 하나로 선정되었다.

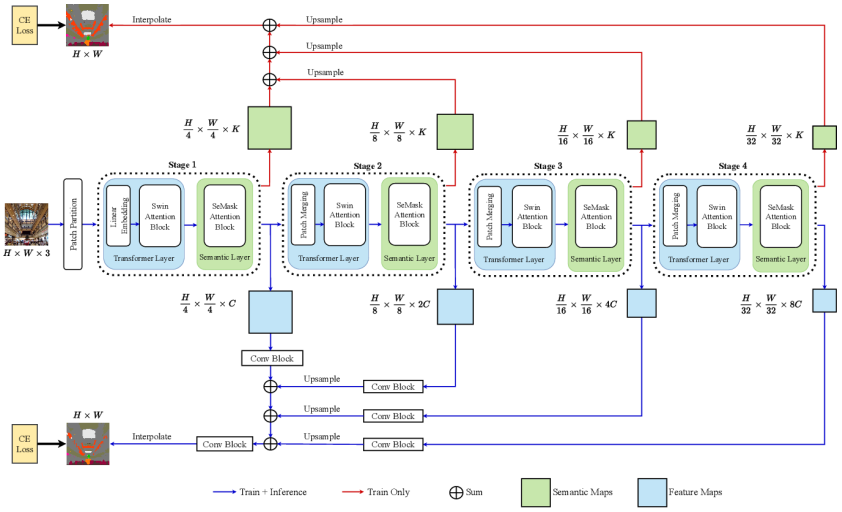


<그림 5. Mask2Former>

4-3-2. SeMask

SeMask 모델은 Encoder 파트에서 Swin 백본을 활용한 Transformer Block과 Attention 계산할 때 Segmentation에 대한 feature map을 추가로 추출하는 Semantic Layer로 구성되어 있는 것이 특징이다. 기존 Transformer 계열 Segmentation 모델에서는 Transformer layer에서 classification과 segmentation을 같이 학습했다면 SeMask는 2단계를 거쳐 segmentation와 관련된 QKV를 뽑아 Segmentation 정보를 뽑아내기 위한 점수를 따로 계산한다. 또한 각 stage 별 Encoder에서 나온 Semantic Map들을 뽑아 계층적으로 합성하는 방식을 적용하면서 low level feature부터 high level feature까지 정보를 뽑아낼 수 있기 때문에 더 정확한 Segmentation을 뽑아낼 수 있다는 장점이 있다.

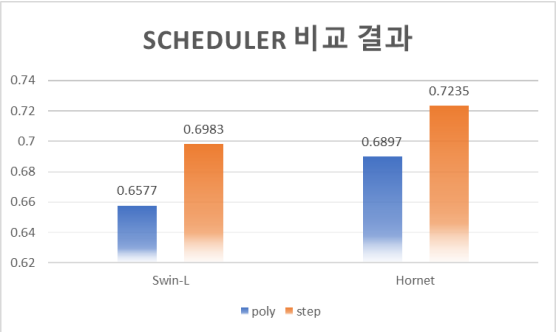
이와 같은 장점들로 인해 SeMask는 ViT-adapter 대비(571M) 절반의 파라미터 수(227M)로 비슷한 성능을 뽑아내기에 앙상블 모델 중 하나로 선정되었다.



<그림 6. SeMask>

4-4. Optimization

4-4-1. Scheduler

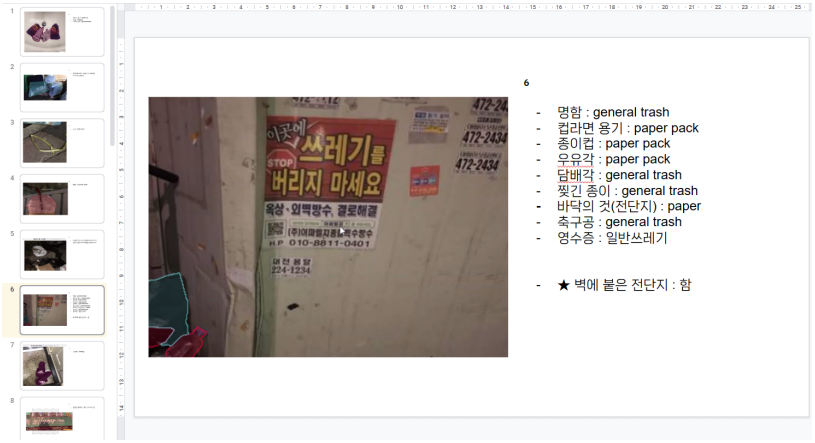


<그림 7. scheduler 비교 결과>

Scheduler의 경우 대부분의 모델에서 learning rate를 초반에 증가시켰다가 천천히 감소시키는 poly가 기본으로 사용되고 있었으나, 일부 모델의 경우 step을 통해 특정 에폭에서 learning rate를 조절해주었을 때 좋은 성능을 보여 변경하여 사용하였다.

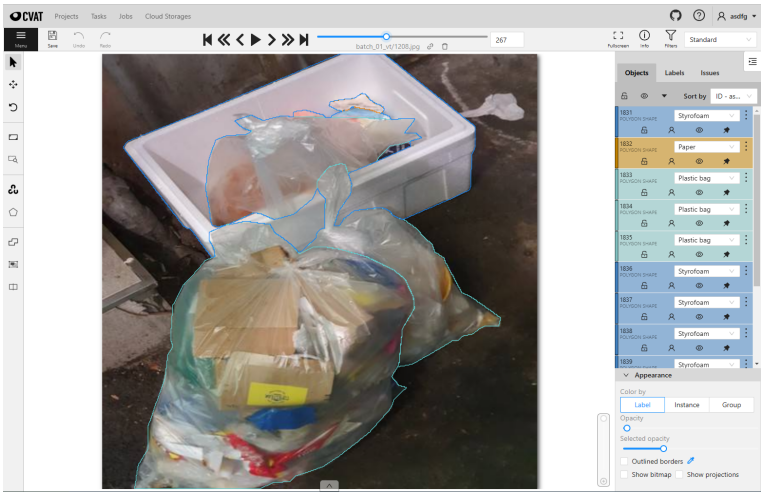
4-5. etc (★)

4-5-1. Data Cleansing



<그림 8. Annotation 메뉴얼>

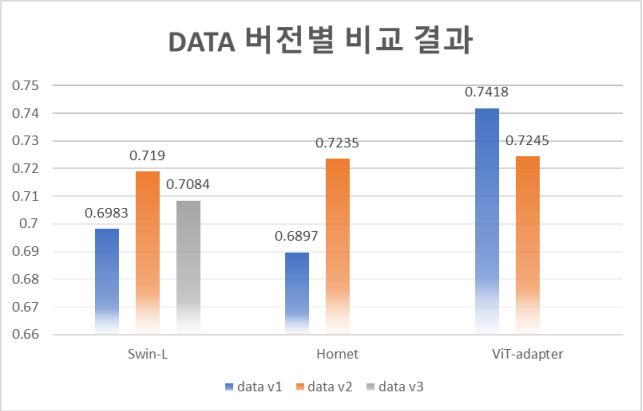
Annotaion 데이터의 경계선이 틀리거나 라벨링의 일관성이 부족하다고 판단하여 전 팀원이 train과 validation 데이터 3272장에 대해 데이터를 전수조사, 팀원과 토의하여 생성한 메뉴얼 대로 DataCleansing을 진행하였다.



<그림 9.cvat 메뉴얼>

선택한 annotation tool은 CVAT으로 제공된 데이터인 coco dataset을 지원한다. 또한 로컬에서 동작하기 때문에 private data의 유출에 대한 걱정이 없기 때문에 선택하였다. Organization 그룹을 기준으로 Maintainer가 프로젝트를 생성 후 task를 생성, 이후 worker들에게 각 task을 부여할 수 있기 때문에 팀원들이 독립된 파일에 접근하여 충돌 없이 annotation 작업이 가능하다. 상세한 구현 내용은 notion으로 정리하였다[1].

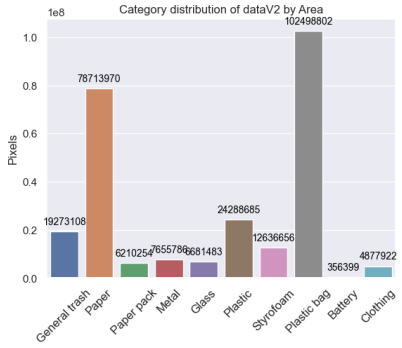
4-5-2. Data Versioning



<그림 10. Data 버전별 비교 실험 mIoU 결과>

Data versioning을 통해 Data에 수정이 가해질 때마다 성능 향상 폭을 측정하여 사용하였다. data v1은 초기 데이터셋을 클래스 개수를 이용해 stratified group k-fold를 진행한 데이터이며 가장 기초가 되는 버전이다. Data Cleansing 진행 후 클래스 개수를 이용해 stratified group k-fold를 진행한 버전이 data v2, 클래스별 픽셀 개수를 이용해 stratified group k-fold를 진행한 버전이 data v3이다. 대부분의 모델에 대해서 v2, v3를 이용했을 때 성능이 오르는 것을 확인하였다. 다만 ViT-adapter의 경우만 성능이 data v1에 비해 감소하여서, 우리가 진행한 Data Cleansing이 adapter 모델에서 특징을 추출하는 데 방해한 부분이 있다고 판단하여 ViT-adapter에 대해서만 data v1으로 진행했다.

4-5-3. Class Weight



<그림 11. 전체 클래스별 픽셀 분포>

class_weight

파일 수정 보기 삽입 서식 데이터 도구 확장 프로그램 도움말 3일 전에 마지막으로 수정했습니다.

F15

	A	B	C	D	E	F
1		클래스별픽셀	전체 픽셀 / 클래스별 픽셀	exp(logC~)	클래스별 픽셀 / 전체 픽셀	
2	background	594542103	1.442681963	1.172537967	0.6931534641	1.172537967
3	general trash	19273108	44.50424747	5.198662921	0.0224697654	5.198662921
4	paper	78713970	10.89686072	2.821592382	0.09176954955	2.821592382
5	paper pack	6210254	138.115956	8.501510501	0.007240293078	8.501510501
6	metal	7655786	112.0375058	7.76295788	0.008925582494	7.76295788
7	glass	6681483	128.3749683	8.235716357	0.00778680602	8.235716357
8	plastic	24288685	35.31418716	4.701817868	0.02831723113	4.701817868
9	styrofoam	12636656	67.8767522	6.244624751	0.01473258469	6.244624751
10	plastic bag	102498802	8.368245787	2.515909649	0.1194993581	2.515909649
11	battery	356399	2406.671085	29.41241135	0.0004155117026	29.41241135
12	clothing	4877922	175.8402795	9.441533042	0.005686979131	9.441533042

<그림 12. 픽셀 수를 기반으로 여러 버전의 class별 weight 계산>

```
=B2/$B$13        =exp(-log(E2))        =exp(log(V2)) * sqrt(E2) * 0.2
```

다양한 class_weights 계산 방식

보다 효율적인 학습을 위해 class_weight를 계산하여 모델 학습에 활용하였다. 이때 클래스별 픽셀 수 비율 정보를 유지하면서 일부 클래스가 지나치게 큰 값을 가지지 않게 하는 동시에 클래스별 학습/추론 난이도까지 고려될 수 있도록 가중평균 및 log() 및 exp()등의 함수를 활용하여 다음과 같은 다양한 버전의 class_weights들을 테스트하였다.

```
#class_weight=[1.172537967, 5.198662921, 2.821592382, 8.501510501, 7.76295788, 8.235716357, 4.701817868, 6.244624751, 2.515909649, 29.41241135, 9.441533042] + [0.1]
#class_weight=[0.1141056336, 9.00117906, 1.18494101, 4.341233462, 3.495662065, 4.817374539, 5.049535857, 1.860116676, 0.9740785703, 2.451909869, 2.371106075] + [0.1]
#class_weight=[0.1034721337, 7.162501468, 0.8808323956, 1.375910059, 3.742064271, 5.805553932, 3.899901354, 1.378751588, 1.033751852, 2.287765275, 5.305349843] + [0.1]
class_weight=[0.01, 3.112177639, 1.689145238, 5.089426124, 4.64729187, 4.930308558, 2.814741532, 3.738342302, 1.506148383, 17.6077292, 5.652170271] + [0.1]
#class_weight=[1.0] * num_classes + [0.1]
```

class별 난이도는 각 epoch의 validation score를 확인하여, class별 accuracy를 활용하여 계산에 적용하였다. 실험을 통해 내린 결론은, 학습 과정에서 매 iter epoch마다 모델의 클래스별 성능이 변하기 때문에, 결국 매 iter마다 난이도를 새롭게 계산하여 class_weights를 업데이트 하는 것이 최적이라고 생각되었으나 라이브러리(mmsegmentation) 코드 수정의 어려움 및 대회 종료 일정을 고려하여 일정 학습 구간이 지나면 모델 checkpoint를 저장하고 학습을 멈춘 뒤 새로운 class_weight를 config에 적용하고 모델을 load_from하여 이어서 학습시키는 방식으로 class_weight를 수동으로 업데이트 하는 방식을 시도하였다.

5. 프로젝트 수행 결과

5-1. Evaluation

Model	mIoU
upernet_hornet_large[2]	0.7310
semask[3]	0.7353
mask2former_swin[4]	0.7433
ViT-adapter[5]	0.7418

<표 2. 단일 모델 k-fold 중 최고 결과>

Model	mIoU
upernet_hornet_large	0.7356
semask	0.7419
mask2former_swin	0.7580
ViT-adapter	0.7552

<표 3. 단일 모델 k-fold 앙상블 결과>

5-2. Ensemble

swin	adapter	semask	adapter_ dataV2	upernet_ beit	swin_dat aV3	hornet	mIoU
✔	✔	✔					0.7716
✔	✔	✔	✔	✔	✔		0.7810
✔	✔	✔	✔	✔	✔	✔	**0.7828**

<표 4. 상위 3개 앙상블 결과>

- 모든 조합의 모델을 다 포함시켰을 때 가장 높은 점수를 기록하였다.
- 모델마다 잘 예측하는 클래스가 다른 부분으로 인해 점수가 상승하였다고 판단된다.

6. 자체 평가 의견

지난 3주간의 대회 전체 일정 가운데 여러 도전적인 상황들이 발생하였지만 팀원 모두가 협력하여 함께 이겨내고 좋은 결과를 빚어낼 수 있었다. Annotation error가 많고 노이즈가 심한 데이터셋이 주어진 상황을 극복하고자 data annotation 전수 검사 및 수정이라는 어려운 과업를 팀원 모두가 합심하여 완수해낼 수 있었다. 모델 학습에 상당한 시간을 필요로 하는 segmentation task의 제약사항을 이겨내고 ViT-Adapter, SeMask, Mask2Former, EVA, BEIT, SwinV2, DiNAT, HorNet 등 현재 사용가능한 다양한 SOTA 모델들을 mmsegmentation, detectron2 등 복수의 라이브러리를 세팅하여 학습 완료하였다. 타이트한 일정 속에서 각 팀원들끼리 유휴 자원을 고려하고 스케줄을 배분하여 학습을 진행하였고, 전체 모델의 5-folds를 확보하여 ensemble하는 실험을 완료, 높은 테스트 점수까지 얻을 수 있었다.

7. Reference

[1] https://heavy-edge-701.notion.site/CVAT-516e44b823f34280aed3b50d4aaebcab

[2] Rao, Yongming, et al. "Hornet: Efficient high-order spatial interactions with recursive gated convolutions." *arXiv preprint arXiv:2207.14284* (2022).

[3] Jain, Jitesh, et al. "Semask: Semantically masked transformers for semantic segmentation." *arXiv preprint arXiv:2112.12782* (2021).

[4] Cheng, Bowen, et al. "Masked-attention mask transformer for universal image segmentation." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022.

[5] Chen, Zhe, et al. "Vision Transformer Adapter for Dense Predictions." *arXiv preprint arXiv:2205.08534* (2022).

8. 개인 회고

8-1. 박민지

[마지막 대회형 프로젝트**]** 지난 level-2에서 3가지의 대회를 진행하였고 이번 대회는 그 마지막을 장식하였다. 처음 Object Detection에서는 SOTA 모델 선정 여부와 추출하는 모델의 특성에 따른 앙상블 효과가 어떤지 확인해 볼 수 있는 계기가 되었다면, OCR 추출 데이터 제작 챌린지에서는 깃헙을 통해 OCR 학습시키는 방법을 참고 삼아 50만장 이상의 데이터셋을 사전 학습 시킨 후 그 다음 데이터셋에 대해 Fine-tuning을 시킨 방식이 성능을 끌어낼 수 있다는 걸 확인해 볼 수 있는 계기가 되었다.
그리하여 마지막 대회에서 Data와 모델 둘다 검토하여 지난 대회들의 경험을 최대한 살리고자 하였으며, Data Driven Approach와 Model Driven Approach를 진행할 수 있었다.

[데이터 전수 검사를 통한 **Data Driven Approach]** 데이터를 검토하는 데 빠진 labeling이나 잘못 표기된 annotation이 있어 5일동안 데이터를 전부 확인하여 재작업을 하여 데이터 버저닝을 진행하였다. 그리고 해당 과정을 통해 서로 다른 모델이더라도 리더보드 점수가 0.05~0.1 향상하였고, public에서 private 점수 또한 향상되는 것을 확인할 수 있었다.

[SOTA 모델을 학습하기 위한 **Detectron2**와 **MMSegmetation]** SOTA 모델들을 학습하기 위해 Detectron2와 MMSegmentation을 삽질하면서 배워야 했다. 특히 Mask2Former 계열은 Detectron2에서 구현되었기 때문에 서버에 맞는 버전을 설치해야 했고, 버전 차이로 인해 구현되지 않은best checkpointer hook을 별도로 작성해야 했다. 해당 과정을 통해 어떤 라이브러리더라도 돌아가는 코드에 대해 분석할 수 있다는 자신감이 생기게 되었다.

[모델이 꺾였다 생각했지만 정작 꺾인 건 나의 믿음**]** 하지만 진행하면서 내가 돌린 모델이 정작 성능을 올리지 못해 막판 스퍼트를 내지 못한 점이 아쉬웠다. 모자란 시간 내에 5-fold를 다 돌려 앙상블을 해야 했기에 iter수가 2~3만으로 정해야 했다. 앙상블 효과가 꽤 괜찮게 나왔다는 점에서 위안을 삼아야 했지만, 스페셜 피어세션 시간에 나와 같은 모델을 사용한 다른 캠퍼님들의 이야기를 들어보니 iter 수를 8만 이상으로 학습시켰다고 한다. 모델이 꺾였다 생각하여 낙심했지만, 돌아보니 꺾인 건 모델에 대한 나의 믿음이었다. 실험을 여러개 나눠서 진행해서 분석하거나 모델에 대한 믿음이 굳건해야 했다.

[포스트 부스트캠프**]** 이제 부스트캠프가 40일 정도 남은 시점에서 부스트캠프 끝나고 앞으로 뭘 할지 고민하고 있다. 가장 큰 계획은 취직 혹은 대학원이겠지만, 중간 단계로 대회에 참가해서 입상하고, 부스트캠프를 통해서 배운 점들을 노션에 기록하여 호스팅 페이지 만들기로 했다.

8-2. 최동혁

- 대회 초반 팀에서 SOTA 모델들을 분석하여 어떤 모델이 가장 적합하며 최고의 성능을 보여줄 수 있을지, 그리고 다른 모델들과 ensemble까지 진행했을 때 어느 정도의 결과까지 예상 가능한지 분석 및 테스트가 완료된 상태에서 대회 중반을 맞이할 수 있었고, 덕분에 data annotation 등 새로운 방법론을 적극적으로 실행해 볼 수 있었다.

- Segmentation task의 중요성만큼, 이미 다양한 방법론과 그에 기반한 모델들이 제시되어 있었고 벤치마크 점수가 다소 낮더라도 dilated attention 등 새로운 시도를 한 모델들(EVA, DiNAT 등) 위주로 본 대회에 적용해보며 많은 배움을 얻을 수 있었다.

- 다만 다양한 모델을 적용하는 과정에서 라이브러리 버전 충돌이 거의 매번 발생하였고 여러 모델을 새로 세팅할 때마다 트러블슈팅 과정을 거쳐야 했기에 그 부분이 스트레스로 작용하며 상당한 시간이 소요되었다. 그렇지만 그 과정에서 mmsegmentation, mmcv 라이브러리 내부의 실행 코드들과 모듈에 좀 더 익숙해질 수 있었던 좋은 경험이었다.

- 다양한 오픈소스 딥러닝 라이브러리를 활용해 모델을 학습시키는 작업에 익숙해질수록 해당 소스 코드를 구현한 숙련된 개발자, 연구자 분들에 대한 경외심이 높아지고 배움의 깊이가 많이 부족함을 느끼게 되었다. 특히 AI 모델링 부분의 구현 외에도 전체 파이프라인이 작동되도록 하는 다양한 core 모듈, api 등 아직 제대로 파악하지도 못한 복잡한 구조를 마주할 때 향후 공부 방향에 대한 고민들이 깊어지게 되었는데, 지난 대회들(object detection, OCR) 그리고 이번 segmentation 대회까지 진행하며 가지게 된 결론은 결국 현 단계에서 내가 어렵다고 느끼는 것은 결국 같은 레벨에서 학습을 진행하는 다른 분들도 어려움을 느낄 것이기에 조바심을 느끼지 말고 기본기에 충실할 필요가 있다는 생각을 하게 되었다. 그런 배경에서 본 프로젝트에서 사용하는 모델의 구조와 코드 구현부를 좀 더 살펴보고자 노력하였고, 다음 최종 프로젝트 (segmentation 경량 모델링)까지 그런 노력을 이어가 보고자 한다.

- AI 프로젝트를 진행함에 있어 개발 일정과 성능 구현에만 매진하기보단 더욱 폭넓은 시각으로 혹여 발생할지 모를 상황에 대비해 안전한 계획을 설정하여 프로젝트를 진행해야 한다는 점을 배울 수 있었던 부분이 뜻깊었다.

- 운이 좋아서 좋은 팀원들이 모여 협업하는 기회를 가질 수 있었고, 각자 최선을 다해 협업하며 매번 좋은 퍼포먼스를 기록하는 과정을 바라보며 AI 지식 외에도 좋은 팀의 힘, 강력한 시너지까지 함께 배울 수 있었던 점이 가장 감사하다고 생각한다.

8-3. 유영준

이번 프로젝트에서 나의 목표

- Semantic Segmentation을 수행하기 위한 과정 전반에 익숙해지기
- 다양한 SOTA 모델들을 이용해 최고 성능 이끌어내기

학습목표를 달성하기 위해 한 일

- Detection 대회 때 mmcv 프레임워크를 사용해 본 경험을 기반으로, mmsegmentation 내에서 다양한 비교 실험을 진행하였다.
- 가장 좋은 한 모델만 집중 실험하기보다는, 최종적으로 ensemble에서 좋은 성능을 기대하기 위해 다양한 SOTA 모델들에 대한 실험을 진행하였다.

모델 개선 방법

- 대부분의 모델에서 default로 learning rate가 초반에 증가하였다가 천천히 감소하는 poly scheduler를 사용하고 있었는데, 이는 학습을 천천히 오래 진행할 때 효과가 좋다고 판단하였다. 따라서, StepLR 을 이용하여 특정 epoch에서 learning rate를 떨어뜨리는 방법을 통해 성능을 향상시켰다.
- batch size가 클수록 noise가 줄어들 것이라고 판단하여, GPU 메모리 허용 범위 내에서 최대한의 batch size를 이용하였고, batch size가 너무 작게 허용되는 경우에는 Gradient Accumulation을 이용하여 큰 batch size를 갖는 효과를 내도록 하였다.
- Segmentation 데이터 특성상 annotation이 정확히 되어있지 않은 부분이 많다고 판단하여, 팀원들과 분담해서 Data Cleansing 작업을 진행했다.

행동의 결과로 달성한 지점, 얻은 깨달음

- 논문이나 프레임워크에서 기본으로 구현해 놓은 모델의 하이퍼파라미터들은 해당 저자들이 많은 실험을 통해 최적으로 구현해 놓은 것이라고 생각하고 크게 바꿀 생각을 하지 않았었다. 하지만 이번 대회에서 다양한 실험을 해보면서 task나 data에 따라서 customizing을 통해 성능을 많이 향상시킬 수 있음을 알았다.

전과 비교해서, 내가 새롭게 시도한 변화와 그 효과

- Data Cleansing을 통해 점수를 올릴 수 있었는데, segmentation task에 있어서 annotation의 중요성을 깨달았다. 데이터를 직접 전수조사해 본 결과, 성능에 있어서 annotation에서 어떤 점을 고려해야 할지, 일관성을 가지도록 하는 게 얼마나 중요한지 등을 깨닫는 계기가 되었다.

마주한 한계와 아쉬웠던 점

- 성능이 좋은 모델일수록 모델 크기가 커서 학습 시간이 매우 오래 걸렸고, 그로 인해 큰 모델에 대해 다양한 실험을 할 수 없었던 점이 아쉬웠다.
- Data Augmentation이나 Ensemble 등 당연히 점수가 오를 것이라고 생각한 부분들에 대해 기대에 달성하지 못한 부분들이 있었다.

한계/교훈을 바탕으로 다음 프로젝트에서 스스로 새롭게 시도해볼 것

- 시간적인 한계가 있을 때, 작은 모델에 대한 실험 결과를 일반화하여 큰 모델에도 적용할 수 있도록 고려해보는 시도가 필요할 것 같다.
- 다음 프로젝트에서는 하이퍼파라미터는 고정하고, 모델 구조의 변화를 통해 목표를 달성하고자 하므로, 그 동안 많이 고려하지는 않았던 모델 구조 수정에 대한 공부가 많이 필요할 것 같다.

8-4. 민기

이번 프로젝트에서 나의 목표

- Data cleansing을 직접 시도하여 성능 향상 시도

학습목표를 달성하기 위해 한 일

- Wandb sweep을 사용한 학습 파라미터 탐색
- 아래와 같은 기준으로 annotation tool 탐색, CVAT 선정
    - 제공된 annotation data format인 coco dataset을 지원하는 tool
    - 로컬에서 동작되어 데이터 유출의 문제가 없는 tool
- CVAT 구축 (7.Reference [1] 참조)
- Data cleansing 진행
- Data 변경점에 따른 Versioning 진행
- 제출 횟수의 제한으로 인해 출력 결과물을 직접 확인하여 앙상블 효과 진단

모델 개선 방법

- 제공된 데이터셋에 픽셀 단위로 클래스가 저장되어 annotation data의 클래스 개수가 실제 클래스 분포와는 차이가 있다는 것을 알게되어 클래스가 차지하는 영역 크기의 합으로 stratified k-fold 진행

행동의 결과로 달성한 지점, 얻은 깨달음

- Data cleansing이 private data와 다를 시 오히려 점수가 하락할 수 있다.
- 클래스의 분포가 적다고 학습이 안되는 것이 아니다. Segmentation Task에서는 object의 경계선이 명확한지의 여부가 더 중요하다
- Head pretrained model이 아닌 backbone pretrained model이 성능이 더 높게 나올 때

전과 비교해서, 내가 새롭게 시도한 변화와 그 효과

- 이전 대회까지 수행한 대회 진행과정에서 data에 대한 탐색이 소홀하다고 생각하여 이번 대회에서는 data에 집중하려고 노력하였다
- Versioning을 통해 팀원들이 혼동 없이 동일한 조건에서 실험할 수 있도록 하였다
- 이를 통해 모델별로 나타나는 효과가 다른 것을 깨닫게 되었으며 최종 제출물에서는 각 모델의 최적의 version으로 학습을 진행하였다

마주한 한계와 아쉬웠던 점

- data cleansing을 통해 data를 확실히 알고 진행할 수 있었으나 너무나 많은 시간을 소비하여 모델이나 augmentation, hyper parameter tuning에 대한 실험이 부족하였다.
- 팀원와 토의한 끝에 기준을 정하고 cleansing을 진행하였지만, 프로젝트의 목표를 다시 되집어 보고 진행했어야 한다.

8-5. 장지훈

이번 프로젝트에서 나의 목표

- SOTA 모델을 이용하여 mIoU 점수를 극대화 시켜보는 것을 목표로 함

학습목표를 달성하기 위해 한 일

- 학습에 mask 이미지가 사용되므로 coco format에 들어있는 segmentation 정보를 이미지로 바꿔주는 작업을 수행
- 주어진 서버 환경에서 sota 모델이 돌아갈 수 있도록 환경 설정
- 학습이 오래걸려서 밖에서도 실험 결과를 확인할 수 있게 wandb 설정도 해줌
- 이미지 검수 및 수정 작업을 하기 위해서 coco format을 나누고 합치는 코드를 작성

모델 개선 방법

- 이미지 검수 및 수정을 통해서 이미지의 노이즈를 제거해 줌
- 쓰레기 데이터셋과 비슷한 정보를 가진 coco 데이터셋으로 학습된 pre-trained를 이용하여 학습함
- PhotoMetric 어그먼테이션을 사용하여 밝기, 대비, 색상에 일반화를 해줌
- batch size가 너무 작아서 Gradient Accumulation 기법을 사용함

행동의 결과로 달성한 지점, 얻은 깨달음

- 큰 모델일수록 strong-aug를 넣어주면 오히려 성능이 하락하는 경우가 더 많음
- 데이터 수가 적은 클래스가 학습을 덜 할 줄 알았는데 오히려 형태가 확실해서 인지하는순간부터 점수가 높게 나옴 (ex: 배터리)

전과 비교해서, 내가 새롭게 시도한 변화와 그 효과

- Wandb에서 segmentation 예측 사진을 볼 수 있도록 구현함
- config.py를 만들어서 체계적으로 실험을 기록할 수 있게 구현함
- Gradient accumulation를 이용하여 batch size를 키우는 효과를 만듬

마주한 한계와 아쉬웠던 점

- 성능이 너무 큰 모델을 가지고 학습을 해서 여러 실험을 해보지 못함
- Augmentation에 너무 민감해서 aug에 대한 확신이 줄어듬
- Wandb에서 2번 validation 되는 문제를 해결하지 못함 (mmseg 라이브러리 문제)
- 여러 fold 데이터를 학습시켜야했기에 시간 관리때문에 더 많이 학습하지 못한게 너무 아쉬움 (1등팀은 iter를 160000번 이상 학습시킴)

한계/교훈을 바탕으로 다음 프로젝트에서 스스로 새롭게 시도해볼 것

- 대회에서는 데이터 검수만 하고 수정말고 삭제하는 것으로 대체해야 함
  -> 대회 test 데이터가 무슨 기준으로 라벨링 되어있는지 몰라서 도박 요소가 큼
- 실험을 더욱 체계적으로 기록할 수 있게 좀 더 자동화를 추가해야 함