

Wrap-up Report: Semantic Segmentation

Taehan Kim, Chaewon Mun, Donghwan Seo, Namgyu Youn, Jaehoon Lee, Jiwoo Jang

Abstract

Bone Segmentation은 딥러닝의 대표적인 응용 분야로, 다양한 목적으로 도움을 줄 수 있다. 이를 정확하게 예측하는 프로젝트를 경험함으로써 Segmentation에 대한 깊이 있는 학습하는 것이 주 목표이다. 또한 이를 통해서 만들어진 우수한 성능의 모델은 질병 진단, 의료 교육 등에 응용될 수 있다.

김태한	모델 탐색, 2-Stage 형태 학습 도입, CRF 후처리
문채원	YOLO 실험, WandB Sweep 환경 구축, K-fold CV 구현, Ensemble
서동환	코드 모듈화, U++ 모델 및 백본 실험, 파라미터 튜닝, Ensemble
윤남규	U3+ backbone, K-fold HOOK, TTA, Ensemble 구현 및 실험
이재훈	Data Augmentation 실험, Loss 실험, Streamlit 연결, Ensemble
장지우	EDA, Meta data 실험, Seed 실험, Streamlit 연결, ViT 실험, Ensemble

Table 1. 구성원 및 역할 소개

Detailed Figure

- Topic : Hand bone segmentation
- Organizer: NAVER Connect Foundation, Upstage AI
- Timeline : 2024-11-11 - 2024-11-28
- GPU : 4 SSH server (Nvidia V100 32GB)
- Co-operation Tools : GitHub, Notion, Slack, VSCode, WandB

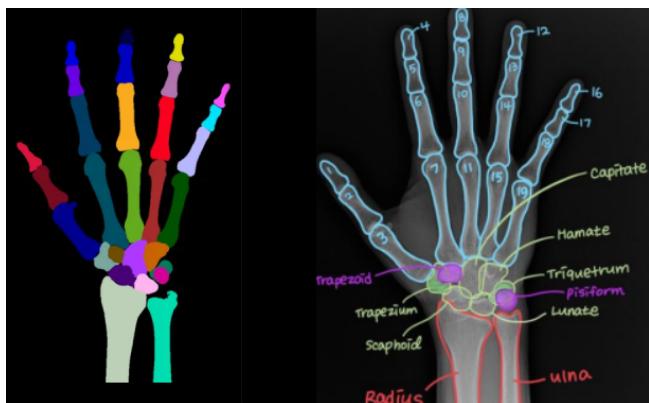


Fig. 1. Hand Bone Segmentation

Modeling

FCN. Baseline의 FCN 모델에 ResNet50 backbone을 기준으로 dice 값이 0.9392가 나오는 것을 확인했고, segmentation task 특성상 이미지 크기가 커지면 세밀한 정보를 포착하는데 유리할 것이라고 판단하여 해상도를 512에서 1024, 1536

으로 증가시켜 실험을 진행하였다. 그 결과, 이미지 크기가 커지면서 수렴 속도가 빨라지고 dice 값이 향상되는 것을 확인할 수 있었다.

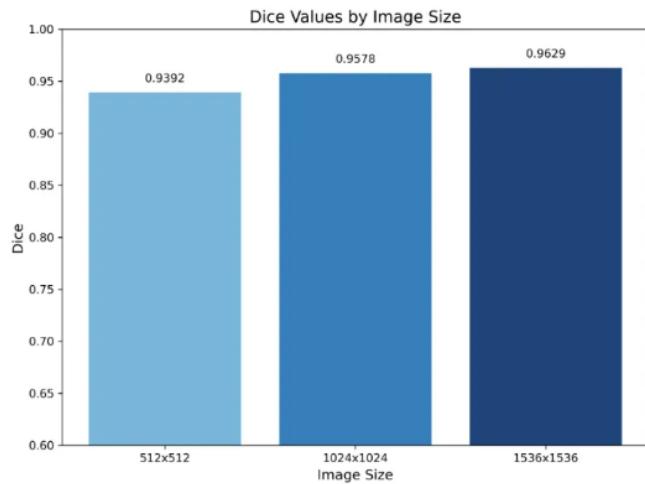


Fig. 2. Dice per image size

U-Net++. 기존 FCN 모델의 경우, 다운샘플링을 통해 공간적인 정보를 일부 잃어버리게 되고, 여기서 나온 저해상도 feature map 만을 활용하여 업샘플링을 진행한다는 점에서, 객체의 경계를 정확하게 잡아내지 못한다는 문제점이 존재한다.



Fig. 3. U-Net Architecture

따라서 위의 문제를 해결하고자, U-Net++ model을 사용하고 encoder로는 ImageNet으로 pretrained된 EfficientNet-b7을 이용하였다. U-Net++의 경우, 여러 개의 skip connection을 활용하여 다양한 스케일의 feature map으로부터 나오는 특징들을 효율적으로 결합시키는 방법을 사용한다. 이를 활용하여 객체의 경계 부분을 더 세밀하게 분류할 수 있을 것이라고 판단하였고, FCN 모델 대비 $0.9578 \rightarrow 0.9653$ 으로 약 0.7%의 유의미한 성능 향상을 보였다.

U-Net++ model의 encoder를 EfficientNet-b7에서 다른 모델로 변경했을 때의 성능을 비교해볼 때, EfficientNet-b8이나 EfficientNetv2_xl 와 같은 더 무거운 모델로 변경했을 때는 오히려 성능이 떨어지는 것을 확인할 수 있었다. 주어진 데이터 개수가 많지 않아 과적합되었을 가능성이 있으며, 파라미터 값이 해당 모델에 맞지 않아 이러한 결과가 나왔을 것으로 판단된다.

HRNet은 EfficientNet에 비해 성능이 약 0.09% 향상되었는데, 고해상도의 정보를 유지하면서 다양한 스케일의 feature map을 학습하고 이를 결합시킨다는 점에서 성능 향상이 있었을 것으로 예상된다.

Encoder	Weight	DICE score
EfficientNet-b7	ImageNet	0.9712
EfficientNet-b8	ImageNet	0.9542
EfficientNetL2	-	0.9686
ResNeXt ₃ 2x4d	ImageNet	0.9693
EfficientNet _{timm} -b6	ImageNet	0.9578
HrNet _w 64	ImageNet	0.9721

Table 2. U-Net++ model의 backbone별 성능

MaXViT. 모델 학습 기록을 보고 Encoder가 Transformer 계열인 모델이 거의 없음을 확인하여 추후 양상을 때의 성능 향상을 위해 U-Net 계열 모델들에 백본으로 ViT 계열 모델을 사용한 논문을 찾아보았고, MaxViT 모델 논문을 발견하였다. [1]

논문을 읽어보고 우리의 task에 잘 맞을 것 같다고 판단하여 U-Net, U-Net++, U-Net3+ 모델에 인코더로 MaxViT를, 디코더로 EfficientNet을 사용하여 성능을 비교해보았고 U-Net 계열에 인코더로 MaxViT를 사용하니 성능이 향상됨을 알 수 있었다. 각 모델을 비교해보니 성능은 아래와 같았다.

	Val Dice	Test Dice
U-Net	0.9712	0.9701
U-Net++	0.9703	
U-Net3+	0.9718	0.9727

Table 3. MaxViT 실험 결과

그 후 인코더뿐만 아니라 디코더에도 MaxViT를 사용하여 백본 자체를 변경해 학습해보니 리소스가 제한되어 있어 이미지 크기를 낮춰서 학습을 진행해야 했고 성능 하락을 확인할 수 있었다.

YOLO. 기존 모델 실험 과정에서 semantic segmentation 모델들은 상대적으로 단순한 구조의 손가락 뼈에서는 우수한 성능을 보였으나, 복잡하고 겹치는 구조가 많은 손목 뼈 영역에서는 정확도가 떨어지는 한계를 보였다. 따라서, 이러한 한계를 극복하기 위해 Bounding Box와 Mask 생성을 통해 개별 손목 뼈의 정확한 위치를 파악하여 겹침 영역에서의 강점을 가질 것이라는 가설을 가지고 YOLO 모델을 테스트한 결과 아래의 결과를 도출할 수 있었다.

Model	Test Dice
yolov8x-seg	0.9267
yolo11x-seg	0.9236

Table 4. Yolo segmentation model's performance

실험 결과, 단일 모델로서는 semantic segmentation 모델들보다 전체적인 성능이 낮고 손가락 뼈의 픽셀 단위 정밀도가 부족했으나, 겹치는 손목 뼈 영역에서의 높은 위치 정확도가 semantic segmentation 모델의 한계점을 보완할 수 있을 것으로 판단했다. 이에 YOLOv8x-seg 모델을 선택하여 5-fold Cross Validation을 수행했고, 최종 양상을 포함시켰을 때 Dice Score가 0.9722에서 0.9727로 0.05% 향상되어 전체적인 성능 향상에 기여했다.

Augmentation

Baseline을 기준으로 아래와 같은 다양한 증강실험을 적용해 결과를 확인해 보았다.

결과적으로 Baseline과 비교해 더 좋은 성능이 나온 데이터 증강 기법은 Elastic, CLAHE, Rotate, GridDistortion, HorizontalFilp, UnsharpMasking 기법임을 알 수 있었다.

Augmentation	Val Dice	Test Dice
base	0.9459	0.9178
Elastic	0.9469	0.9257
CLAHE	0.9331	0.9258
Sharpen	0.9337	0.8844
Rotate	0.9390	0.9222
GridDistortion	0.9452	0.9207
Emboss	0.9371	0.9000
HorizontalFilp	0.9429	0.9292
RandomBrightnessContrast	0.9462	0.9168
UnsharpMasking	0.9449	0.9241
blur	0.9385	0.8950

Table 5. 개별 증강 실험 결과

이를 토대로 경계를 설명하게 하는 CLAHE, Test 데이터 셋에 많이 분포해 있는 손목이 꺾인 이미지를 위해 Rotate, 데이터 양을 늘리기 위해 HorizontalFilp을 base 조합으로 정하였다.

이 베이스에 데이터 증강 기법을 추가해보면서 최적의 데이터 증강 기법 조합을 찾아보았다.

결과적으로 Base+GridDistortion+Elastic 조합이 가장 좋은 성능을 보인다는 것을 알 수 있었다.

Augmentation	Val Dice	Test Dice
base (= CLAHE, HorizonFilp, Rotate)	0.9316	0.9201
base + Elastic	0.9223	0.8954
base + GridDistortion	0.9334	0.9282
base + GridDistortion + Elastic	0.9426	0.9291
CLAHE + blur	0.9443	0.9232
base + UnsharpMasking	0.9187	
HorizontalFilp + GridDistortion + Elastic		0.9241

Table 6. 증강 실험 결과

A. Horizontal Flip + Grid Distortion + Elastic Transformation. 기본 U-Net++ model을 사용했을 때에 비해 증강 기법으로 HorizontalFlip, ElasticTransformation, GridDistortion 3개를 추가했을 때 0.9695 -> 0.9712로 약 0.18% 향상된 성능을 보여주었다. HorizontalFlip의 경우, 기준 데이터가 왼손, 오른손 두 종류의 데이터로 구성되어 있었으므로, 각각의 데이터의 수가 증가되는 효과로 인해 성능 향상이 있었을 것으로 판단된다.

B. CLAHE. CLAHE 증강을 적용했을 때에는 Val DICE 기준, Baseline에 비해 소폭 하락하는 결과를 보였다. 그러나 각 클래스별 DICE 값을 비교해봤을 때 손가락 뼈의 경우 더 낮아졌으나, 손목 뼈는 더 향상된다는 점을 확인할 수 있었다. 따라서 손가락 뼈는 기존의 코드로 예측한 Mask를 사용

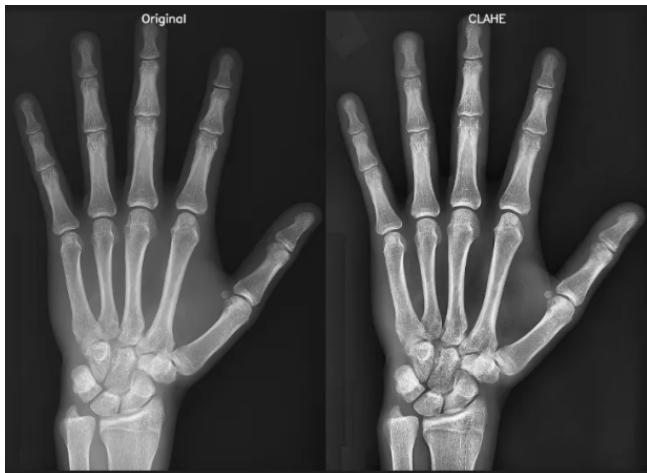


Fig. 4. Augmentation : CLAHE

하고, 손목 쪽 8개의 뼈에 대해서만 CLAHE 증강을 적용하여 예측한 Mask를 사용하여 병합시켰고 "0.9695 -> 0.9701"로 약간의 성능 향상을 가져올 수 있었다.

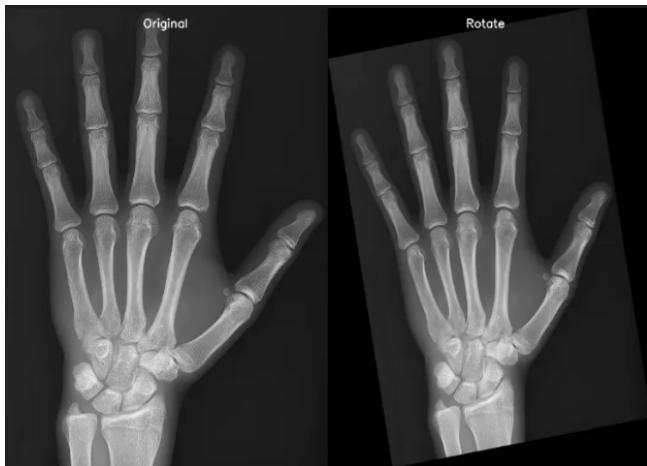


Fig. 5. Augmentation : rotation

C. (Horizontal Flip + Grid Distortion + Elastic) + Rotate. Train, Test 데이터셋에서 손이 꺾인 상태에서 촬영된 이미지가 존재하였고 이러한 이미지들에 대한 segmentation 결과가 다소 부정확하게 나온다는 것을 확인하였다. 손이 꺾인 이미지들에 대해 모델이 더 잘 포착할 수 있도록 하기 위해 rotate 증강을 추가했으나, 위의 경우처럼 객체의 크기가 작아지는 문제가 발생하여 성능이 0.9729 -> 0.9718로 하락하였다.

Research for Optimal Loss Function

Optimal loss를 찾기 위해서 같은 환경에서 loss function을 바꾸면서 실험해 보았다. 실험을 하면서 DICE + BCE 조합의 loss가 가장 성능이 높게 나옴을 알 수 있었고 각각 DICE와 BCE에 label smoothing을 적용하면 어떨까 하여 실험해보았지만 적용하지 않은 버전이 더 좋은 성능을 보였다. 따라서 최종 loss 조합을 dice + bce로 결정했다.

Loss	Trapezoid Val Dice	Pisiform Val Dice	Val Dice
DICE Top4	0.9192	0.9094	0.9589
tversky	0.9193	0.911	0.9681
DICE + BCE	0.9187	0.9090	0.9695
focal+BCE	0.9151	0.8803	0.9655
tversky+BCE	0.9167	0.9076	0.9662
DICE +BCE	0.8600	0.8860	0.9485

Table 7. Loss 실험 결과

Research for Optimal SEED Value

Validation score가 높은 경우에도 불구하고 test(public) score는 하락하는 경우가 많다는 문제점을 발견하여, 두 점수의 상관관계가 가장 높은 seed를 찾고자 실험하였다. 결론적으로는 123이 가장 public score와 상관관계가 높은 시드 값임을 확인하여 123을 사용하였다.

	Val DICE	Test DICE
GridDistortion	0.9452	0.9207
CLAH E	0.9331	0.9258
HorizonFlip	0.9429	0.9292
Sharpen	0.9337	0.8844

Table 8. SEED = 21 (base)

	Val DICE	Test DICE
GridDistortion	0.9468	0.9238
CLAH E	0.9460	0.9220
HorizonFlip	0.9427	0.9168
Sharpen	0.9364	0.8861

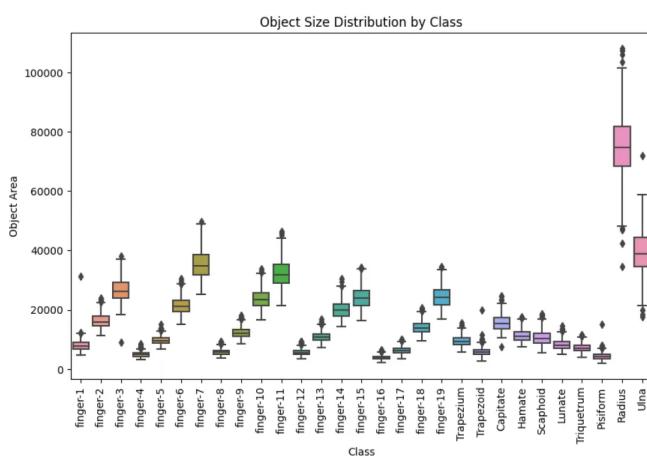
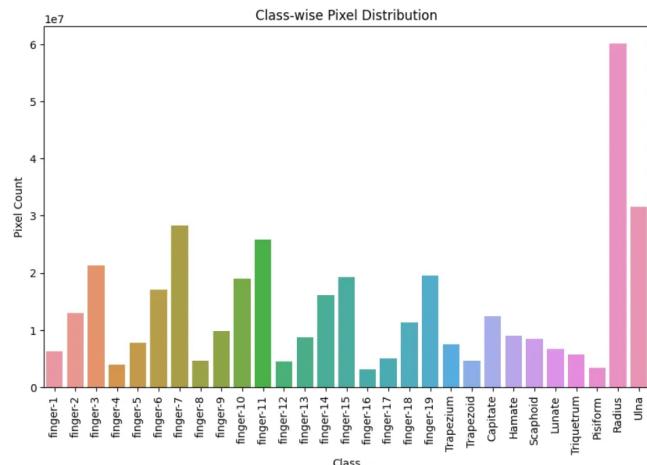
Table 9. SEED = 123

	Val DICE	Test DICE
GridDistortion	0.9477	0.9229
CLAH E	0.9486	0.9271
HorizonFlip	0.9406	0.9214
Sharpen	0.9331	0.8852

Table 10. SEED = 42

Meta Data Research

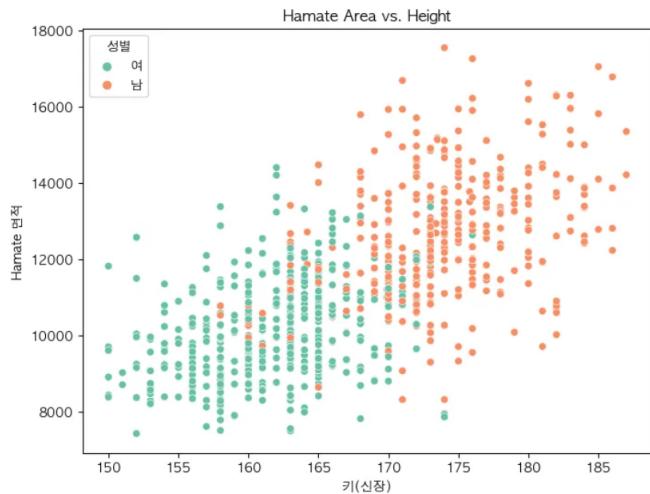
실험 과정에서 모델의 성능이 wrist-bone 부위에서 현저히 저하됨을 발견하였다. 이에 meta data와의 상관관계를 분석하고, 성별, 키, 몸무게 등의 특징을 피처로 추가하여 모델을 재학습시켰다. 그러나 성능 면에서 예상치 못한 변동이 관찰되었으며, meta data를 단순 concatenation 방식으로 입력할 경우 모델이 meta data에 과도하게 의존하는 문제가 발생했다. 이에 임베딩 피처로 meta data를 통합하는 방안을 고려하였으나, 현재 segmentation 과제의 특성상 데이터 간 관계가 예측 성능에 유의미한 영향을 미치지 못한다고 판단하여 meta data 실험을 중단하게 되었다.



Data Cleansing

Train image, test image를 관찰한 결과 아래의 경우가 train image에만 존재하며, 학습 중에 성능이 떨어지는 경우임이 확인되었다. 인식되기 어려운 경우를 부위별로 분류한 내용은 아래와 같다.

- Case 1 : 손가락 끝 마디 : 매니큐어, 손톱 (N = 13)
- Case 2 : 손가락 3번째 마디 : 반지 (N = 1)
- Case 3 : 손가락 끝 마디 : 여백이 부족해 끝 마디가 일부 짤린 경우 (N = 3)
- Case 4 : 손 전체 : 이미지가 훼손되어 noise(실선)가 있는 경우 (N = 2)



- Case 5 : Radius, Ulna : 보형물 (N = 2)

이 중에서 3, 4, 5는 극단적인 Outlier로 판단하였고, 이를 제거한 뒤에 학습해보는 A/B test를 실행했다. 하지만 모델이 학습할 sample size가 감소하면서 성능이 하락하는 현상이 하락되었다. A(원본 데이터)/B(Outlier를 제거한 경우) test 결과는 아래와 같다.

Model	Backbone	DICE (A)	DICE (B)
U-Net3+	MaxViT	0.9718	0.9716
U-Net3+	ResNet	0.9712	0.9606
U-Net3+	EfficientNet-b5	0.9705	0.9697

Table 11. Result of the A/B test

Ensemble

프로젝트가 마무리가 가까워지면서, 다양한 경우를 조합해보면서 성능을 끌어올릴 필요가 있었다. 따라서 Hard Voting, Soft Voting, Weight Voting의 세 가지 앙상블 방법을 구현하고 다양한 데이터셋과 환경에서 그 성능을 평가하였다. 각 앙상블 기법은 서로 다른 예측 메커니즘을 통해 모델의 최종 분류 결과를 결정하는데, Hard Voting은 다수결 원칙을, Soft Voting은 확률값의 평균을, Weight Voting은 가중치 기반 접근법을 사용한다. 실험 결과, 다양한 모델을 조합할 수록 조합된 모델의 단점이 보완되면서 성능이 개선되는 현상을 확인했다.

개인회고 : 김태한

나는 어떤 방식으로 모델을 개선했는가?.

- smp와 torchvision.models 라이브러리에서 지원하지 않는 UNet3+ 아키텍처를 논문과 git 오픈소스를 활용하여 프로젝트에 적용할 수 있도록 했다. 학습된 모델에서 비교적 성능이 떨어지는 손등뼈 Class의 문제를 확인하고 이를 해결하기 위해 2-Stage 학습, 추론 형태를 도입하였다.

내가 한 행동의 결과로 어떤 지점을 달성하고, 어떤 깨달음을 얻었는가?.

- 프로젝트에서 UNet3+ 아키텍처를 활용할 수 있도록 하여 팀원들의 다양한 모델 탐색을 도울 수 있었다.
- 대회 중반에 성능 향상이 정체되었을 때, 모델의 예측 및 Validation 성능을 분석하였고 2-Stage 형태의 방법론을 제시하고 도입하여 최종 모델의 성능을 올렸다.

전과 비교하여 새롭게 시도한 변화는 무엇이고, 어떤 효과가 있었는가?.

- 기존과 다르게 단순한 모델 수정이 아닌 2-Stage 형태 도입, CRF 후처리 등 다양한 방법론을 통해 모델의 성능 향상을 이끌어 내기 위해 시도하였다. 그 결과, 대회 중반 성능 향상이 정체된 상황에서 추가적인 성능 향상을 이끌어 낼 수 있었다.

마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?.

- 개발 환경의 한계라고 생각하여 입력 이미지 사이즈를 원본인 2048이 아닌 1024로 적용하여 프로젝트를 진행한 점이 아쉬웠다. 모델 구조 변경 및 Mixed Precision 등을 활용하여 입력 이미지를 원본 데이터의 손실 없이 2048으로 학습을 진행했다면 더 좋은 성능을 이끌어 낼 수 있었을 것 같다.

한계/교훈을 바탕으로 다음 프로젝트에서 시도해볼 것은 무엇인가?.

- Mixed Precision, Gradient Checkpointing 등 최적화 기법들을 활용하여 최대한 가진 데이터의 정보를 학습에 활용하는 것을 목표로 진행할 것이다.

개인회고 : 문채원

나는 어떤 방식으로 모델을 개선했는가?.

- 기존 train.py에 포함되어 있던 Loss function을 모듈화하여 Dice, Focal, DiceTopK, Tversky 등 다양한 loss 실험이 가능하도록 개선했다.
- Instance-Semantic Ensemble을 위해 Yolo 모델을 실험하였고, SwinUnet, SwinUNETR 등 다양한 모델 아키텍처와 이미지 크기, 데이터 증강 기법에 대한 실험을 진행했다.
- WandB Sweep을 통해 learning rate, weight decay 등의 하이퍼파라미터 최적화와 K-Fold Cross Validation을 위한 데이터 분할 코드를 구현했다.

내가 한 행동의 결과로 어떤 지점을 달성하고, 어떤 깨달음을 얻었는가?.

- 클래스별 성능 지표의 WandB 로깅 시스템 개선과 worst sample 시각화 기능 구현을 통해 라벨링 오류를 발견하여 Data Cleaning까지 수행할 수 있었다.
- 이 과정에서 모델 실험 외에도 데이터 품질 관리와 체계적인 결과 분석이 폴드별 성능 분석과 안정적인 프로젝트 진행의 기반이 됨을 확인할 수 있었다.

전과 비교하여 새롭게 시도한 변화는 무엇이고, 어떤 효과가 있었는가?.

- 협업 툴 중 GitHub를 활용하여 브랜치 생성, 이슈 트래킹, PR 및 코드 리뷰 과정을 체계적으로 관리할 수 있었다.
- Notion에 실험 기록 및 양상을 기록을 상세히 문서화함으로써 실험계획과 최종 우선순위를 정하여 제출기회를 전략적으로 활용할 수 있었다.

마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?.

- 다양한 실험을 진행하는 과정에서 모델 간 성능 차이의 원인을 더 깊이 있게 분석하지 못한 점이 아쉽다.
- MMSegmentation Library를 활용하고 싶었으나 시간 제약으로 인해 시도해보지 못한 점이 아쉽다.

한계/교훈을 바탕으로 다음 프로젝트에서 시도해볼 것은 무엇인가?.

- 각 모델의 공식 논문과 참고 문헌을 통해 아키텍처의 기본적인 동작 원리를 이해하고, 각 모델의 한계점을 파악 후에 개선하기 위한 세부 실험들을 단계적으로 설계하는 과정으로 진행하고자 한다.

개인회고 : 서동환

나는 어떤 방식으로 모델을 개선했는가?.

- 기존 FCN이 경계를 뚜렷하게 포착하지 못한다고 생각하여 UNet 계열의 모델을 사용하여 성능을 개선하였으며 backbone으로 Efficientnet-b7, b8, v2-xl, HRNet 등 다양한 모델을 사용하여 성능을 비교해보았다. 데이터 증강도 기존 base model 기준으로 성능이 올랐던 Horizontal flip, Grid distortion, CLAHE, UnsharpMasking과 같은 기법들을 다양하게 조합해 적용해보았고 optimizer나 학습률과 같은 파라미터를 바꿔보며 모델을 개선했다.

내가 한 행동의 결과로 어떤 지점을 달성하고, 어떤 깨달음을 얻었는가?.

- 초반에 코드 모듈화와 해상도 실험, optimizer 및 학습률 실험을 통해 실험이 원활하게 진행될 수 있도록 했고 UNet++ 모델을 활용하여 전반적으로 높은 모델 성능을 달성할 수 있었다. 다양한 데이터 증강 조합을 적용하여 손목 뼈와 같은 특정 부분을 잘 잡아내지 못하는 문제점을 어느 정도 해결하였다.

전과 비교하여 새롭게 시도한 변화는 무엇이고, 어떤 효과가 있었는가?.

- WandB를 활용하여 진행 중인 실험에 대한 validation dice 뿐만 아니라 worst sample이나 클래스 별 dice 값 등 다양한 정보를 모든 팀원들이 실시간으로 확인할 수 있어 좋았던 것 같다.

마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?.

- UNet++, UNet3+ 와 같이 초반에 성능이 잘 나왔던 UNet 계열의 모델만을 활용하여 실험을 진행해서 마지막에 앙상블할 때 성능이 생각보다 많이 오르지는 않았던 것 같다. YOLO의 경우 단일 모델 기준으로는 성능이 높지 않았으나 앙상블 했을 때는 성능이 많이 올랐는데, 초반에 모델 실험을 더 다양하게 해봤어도 좋았을 것 같다.

한계/교훈을 바탕으로 다음 프로젝트에서 시도해볼 것은 무엇인가?.

- 가장 높은 성능을 보였던 모델 뿐만 아니라 다른 모델들에 대한 실험도 다양하게 진행하며 결과를 같이 비교해보면 좋을 것 같다. 또한 Soft voting과 가중치를 활용한 앙상블 결과를 충분히 확인하지 못한 점이 아쉬웠는데 이러한 방법들을 보다 체계적으로 분석하고 적용해보는 것이 필요할 것 같다.

개인회고 : 윤남규

나는 어떤 방식으로 모델을 개선했는가?.

- U-Net 3+ model의 backbone으로 ResNet보다 후속 연구인 EfficientNet을 구현해 성능 개선을 시도했습니다. 이를 통해서 연산량을 최적화해 GPU 사용량을 기존의 30GB에서 8GB 정도 감소 시켰습니다. 또한 모델이 학습 중 Validation 부분에서 CPU 병목 현상이 발생하는 것을 확인하고, 이를 해결해 학습 속도를 개선 했습니다. 또한 프로젝트 후반부에는 다양한 앙상블을 이용해 성능 개선에 집중했습니다.

내가 한 행동의 결과로 어떤 지점을 달성하고, 어떤 깨달음을 얻었는가?.

- 프로젝트 초기에는 gdown, tmux, GitHub Source Control/Label 같은 도구들을 팀원들에게 소개해줌으로써, 초기 작업 환경을 체계적으로 구축하는 것에 집중했습니다. 후반부로 갈수록 다양한 기능을 구현하고, 이전에 구현된 코드의 기능적 오류를 확인하고 이를 개선하고자 노력했습니다.

전과 비교하여 새롭게 시도한 변화는 무엇이고, 어떤 효과가 있었는가?.

- 이전보다 코드를 간결하고 체계적으로 구조화하는데 중점을 두었습니다. 예를 들어서 config 같은 정보를 class 등을 활용해 체계적으로 모듈화하고, main() 함수의 코드를 기능 별로 구분해 함수화 함으로써 구조를 개선했습니다. 또한 Ensemble 코드에서 이미지를 불러오는 과정 때문에 연산량에 과부하가 생긴다는 점을 해결하기 위해서 chunk를 도입하기도 했습니다.

마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?.

- U-Net 3+ model의 Backbone을 ResNet 대신에 EfficientNet을 사용하는 경우, 연산량이 감소하면서 GPU 사용량이 8GB 정도 감소했음을 확인했지만, 이를 활용하기 위해 image size를 조절하는 시도만 해봤다는 점이 아쉽습니다. 단일 모델의 성능이 떨어지더라도, 모델 양상을 시도한 경우 다른 모델들과 다양성을 이용해서 성능 개선이 이루어지는 경우가 많은데, 이 점을 간과하고 U-Net 3+ model 성능이 떨어진다는 이유로 너무 빨리 포기했다는 점이 아쉽습니다.
- Masking Error를 마지막까지 해결하지 못해서 TTA를 프로젝트에 적용하지 못했다는 점이 아쉽습니다. 이를 해결했다면 성능 개선이 있지 않았을까 하는 아쉬움이 강하게 남습니다.

한계/교훈을 바탕으로 다음 프로젝트에서 시도해볼 것은 무엇인가?.

- 이전에 다른 분들이 구현했던 내용에 새로운 기능을 추가하고 구조를 체계적으로 다듬는 것을 통해서, 보다 완성도 있게 구현하는 경험을 많이 해보는 것을 목표로 하고 있습니다.

개인회고 : 이재훈

나는 어떤 방식으로 모델을 개선했는가?.

- 여러가지 데이터 증강 기법들을 실험해 보면서 모델을 개선할 수 있는 데이터 증강 기법 조합을 찾았습니다.
- 학습에 알맞는 loss 함수 조합을 찾기 위해 여러 loss 함수를 실험해보고 여러 조합을 실험하면서 가장 적합한 loss 함수 조합을 찾았습니다.
- 다른 팀원들이 많은 모델 실험을 해줘서 결과물을 통해 많은 양상을 실험을 진행했습니다.

내가 한 행동의 결과로 어떤 지점을 달성하고, 어떤 깨달음을 얻었는가?.

- train 데이터와 test 데이터를 비교해 test 데이터와 비슷한 느낌을 낼 수 있도록 데이터 증강 기법을 선택했고 이러한 방법이 test 성능에서의 향상으로 이루어졌습니다.
- 베이스 라인의 loss 함수보다 더 좋은 loss 함수가 있을 것이고 여러 loss 함수를 섞으면 더 좋은 성능을 낼 수 있다고 생각해 실험을 진행했습니다.
- 베이스 라인 loss 함수보다 2가지의 loss 함수를 섞어 사용하는 것이 성능 향상에 도움이 되었습니다. 추가적으로 label smoothing을 적용하는 실험을 해보면서 더 좋은 성능을 기대했지만 아쉽게도 성능 하락을 보여 사용하지 않게 되었습니다.

전과 비교하여 새롭게 시도한 변화는 무엇이고, 어떤 효과가 있었는가?.

- 협업 툴로 Github, WandB, Notion 등을 잘 사용하게 되면서 말하지 않아도 팀원들이 무엇을 하고 있고 어떤 점이 문제인지 등을 빠르게 파악할 수 있었고 그로 인해 내가 해야 할 일을 빠르게 파악해 바로 진행할 수 있게 되었다.

마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?.

- 데이터 증강 실험을 진행할 때 Albeumentation을 사용하는 방법만 고려한 것이 아쉬웠다. 다른 팀이 진행한 방법을 보니 오프라인 데이터 증강 기법을 사용했어도 좋은 성능 향상을 보일 수 있을 것 같았다.

한계/교훈을 바탕으로 다음 프로젝트에서 시도해볼 것은 무엇인가?.

- M 실험을 진행하면서 좀 더 넓은 시야를 가지고 실험해야겠다는 생각이 들었다.

개인회고 : 장지우

나는 어떤 방식으로 모델을 개선했는가?.

- 추후 양상블에서의 성능 향상을 위해 태한님께서 구현해주신 U3+ 모델의 코드를 변경해 백본 인코더로 maxvit를 사용하여 U3+에서의 최고 성능과 양상블에서의 성능 향상을 확인할 수 있었습니다.
- validation score와 public score의 상관 관계가 낮은 문제점을 확인한 후 seed를 바꿔가며 실험하였고 가장 높은 상관 관계를 보이는 seed를 찾아 validation score로 성능을 비교할 수 있었습니다.
- 모델 실험 결과를 바탕으로 여러 모델을 soft, hard, weight, fusion 양상을 하였습니다.

내가 한 행동의 결과로 어떤 지점을 달성하고, 어떤 깨달음을 얻었는가?.

- U3+ 백본 인코더로 vit 계열 모델을 사용함으로써 모델의 다양성이 높아져 양상블에서의 성능이 향상됨을 확인할 수 있었고, 단일 모델 부분에서도 최고 성능을 확인할 수 있었습니다. 이로 인해 모델 다양성이 중요하다는 걸 깨달을 수 있었습니다.
- seed를 조절하여 validation set과 test set 사이의 일관성을 맞춰놓아서 validation score만으로 모델들을 비교할 수 있었으며, validation set 설정이 중요함을 깨달을 수 있었습니다.

전과 비교하여 새롭게 시도한 변화는 무엇이고, 어떤 효과가 있었는가?.

- 협업 툴로 Github를 적극적으로 사용하며 팀원들이 만든 이슈를 함께 고민하고 해당 이슈를 해결, 검증한 코드를 더 쉽고 편하게 옮리고 확인할 수 있었습니다.
- sota 모델을 들고 오는 것이 아닌 논문을 찾아보며 task에 맞는 모델을 선택할 수 있었으며 높은 성능을 확인하며 검증할 수 있었습니다.

마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?.

- 서버 간의 파일 이동에 대해서 미리 생각해두지 않아서 가중치 파일 이동에 많은 시간이 걸려 시간이 많이 남지 않았을 때 원하는 실험을 할 수 없었습니다.
- U-Net 계열 모델들을 주로 실험하였고, 다른 모델들은 거의 실험하지 못하여 모델 다양성이 부족해 양상블 때 큰 성능 향상은 볼 수 없어 아쉬웠습니다.
- Meta data를 사용하면 성능이 높아질 것이라는 단순한 생각으로 EDA, 다중 피처 학습을 진행하였는데 segmentation task에는 도움이 되지 않았습니다. 이로 인해 시간을 꽤 소모하여 아쉬웠고, 단순한 생각만으로는 실험, 검증을 세울 수 없으며 ‘왜, 어떻게’라는 질문을 항상 해야겠다고 다짐했습니다.

한계/교훈을 바탕으로 다음 프로젝트에서 시도해볼 것은 무엇인가?.

- 서버 간의 파일 이동에 대해 익히고 더 빠르고 편리한 파일 이동을 구현할 것입니다.
- 초반에 모델을 다양하게 실험하고 성능이 높게 나오지 않아도 추후 양상블을 통해 검증할 것입니다.
- ‘왜, 어떻게?’라는 질문을 스스로 많이 하며 실험, 검증할 것입니다.

References

- [1] Abdul Rehman Khan, Asifullah Khan, "MAXVIT-UNET: MULTI-AXIS ATTENTION FOR MEDICAL IMAGE SEGMENTATION" (arXiv:2305.08396v5, 2023)
- [2] Manuel Cossio MMed, MEng, "Augmenting Medical Imaging: A Comprehensive Catalogue of 65 Techniques for Enhanced Data Analysis" (arXiv:2303.01178, 2023)
- [3] Liang-Chieh Chen, George Papandreou, Senior Member, IEEE, Iasonas Kokkinos, Member, IEEE, Kevin Murphy, and Alan L. Yuille, Fellow, "DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs" (arXiv:1606.00915v2, 2017)