Hand Bone Image Segmentation

CV-16 SOTA조
김동욱, 김세린, 김재진, 이재건, 박정욱, 황은섭

1. 프로젝트 소개

1.1 개요

프로젝트명: Hand Bone Image Segmentation

프로젝트 기간: 2024.11.13 10:00 ~ 2024.11.28 19:00

실험 환경: V100 Linux 서버 4개

Tools: Visual Studio Code, Github, Slack, smp, torchseg, mmsegmentation

커뮤니케이션 및 협업**:** Slack, Zoom, Notion, Kakaotalk

1.2 프로젝트 목표

- 사람의 Hand bone X-Ray 이미지 데이터를 받아 각 이미지를 Semantic Segmentation을 통해 각 픽셀이 어떤 뼈의 class인지를 예측하는 인공지능 만들기입니다.

1.3 Dataset

- Train 데이터 800장, Test 데이터 288장으로 구성된 Hand bone X-Ray 이미지 데이터입니다.
- 각 이미지는 2048 x 2048 사이즈를 가지고 한 사람 당 왼손, 오른손 짝이 지어져 있습니다.
- 이미지마다 총 29개의 Class가 있습니다.
- segmentation annotation은 json 파일로 제공됩니다.

2. 프로젝트 역할 분담

이름	공통	
김동욱	모델 실험, 레포트 작성	Streamlit 구현, copy_paste 증강 구현, 앙상블 실험
김세린		EDA, augmentation 실험, 모델 실험
김재진		모델 리서치, mmseg 구현
이재건		코드환경 및 streamlit 초기 설정, Notion 및 Github 협업 지도, wandb설정.
박정욱		모델 리서치, 인코더 실험, torchseg 구현

황은섭		EDA, augmenataiton 실험, loss 실험

3. 실험 내용

3.1 EDA

이번 프로젝트의 경우 이미지가 외부에 노출되면 안되기에 글로 설명을 하겠습니다.

3.1.1 각 Class의 비율

손가락 뼈는 손가락의 끝 쪽으로 갈 수록 뼈의 크기가 작아집니다.
손등 부분은 겹치는 구간이 많고 크기도 작은 편입니다. 눈으로 확인해도 구별이 어렵습니다. 이번 프로젝트에서 모델들이 제일 학습하기 어려워하는 class 입니다.
팔 뼈는 크기가 제일 크고, 큰 분산을 가집니다. X-ray를 찍을 때 사람마다 손의 위치가 다르기 때문으로 생각됩니다.

3.1.2 Multi Label

다른 뼈들도 겹치는 부분이 있지만 상대적으로 Pisiform과 Triquetrum, Trapezaid와 Trapezium가 겹치는 부분이 크고 많았습니다.

3.1.3 Meta data

dataset에 있는 meta_data 파일을 통해 분석했습니다. train dataset과 test dataset이 구별되지 않은 채로 544명의 키, 몸무게, 성별, 나이에 대한 정보가 들어있습니다.
강의에서 제공한 그래프로 나이에 따른 성별, 몸무게, 신장의 분포를 얻을 수 있었습니다.
추가로 진행한 Meta data로는 성별에 따른 손 크기 분석, 나이 분포 분석이 있습니다. 성별에 따른 손 크기 분석은 성별에 따라 손의 크기가 명확히 구분할 수 있다면 성별을 prompt로 사용할 수 있을 것 같아 진행했습니다. 남자의 손 크기가 여자보다는 컸지만, 그 차이가 크지 않아 prompt로 사용하는 것은 효과가 크지 않을 것이라 예상되었습니다. 나이는 19세부터 60대 후반까지 있었으며, 20~30대가 제일 많았습니다. 카메라와 손의 거리가 이미지마다 달라 몸무게와 신장, 성별을 활용하는 것은 어렵다고 판단했습니다.

3.1.4 데이터셋 확인

데이터를 확인하면서 발견한 특징입니다.

- X-ray를 찍으면서 위쪽 부분이나 좌측에 흰 부분이 보이는 이미지
- 손톱에 무언가가 붙어있는 손 이미지

- 손의 방향이 다른 이미지
- 반지로 보이는 무언가가 손가락에 끼어있는 이미지
- 손가락 뼈가 특이한 이미지
- 손등 부분에 흰머리 같은 자국이 보이는 이미지
- 손목 부근 뼈를 고정하기 위한 의료 도구가 있는 이미지

반지가 있거나, 의료 도구가 있는 이미지는 이상 데이터로도 분류할 수 있을 정도였습니다. 손의 방향이 다른 이미지는 수가 학습하기에 충분할 정도로 많았습니다.

3.1.5 손목, 손등, 손가락 뼈 위치 분포 시각화

손가락 뼈의 위치는 손의 방향에 따라 2가지로 나뉘었습니다. 손등과 손목은 모두 특정 부분에 위치했습니다. 여기서 얻은 정보는 Crop 실험을 진행할 때 위치를 알아내기 위해 사용했습니다.

3.2 augmentation 실험

Transformation

Default

- Model : DeepLabV3+
- BackBone : EfficientNet-B0
- scheduler: CosineAnnealingLR

	Dice Score
Default(Resize : 512)	**0.9385**
Horizontal Flip	0.9382
Rotate	0.9382
RandomBrightnessContrast	0.9373
Gaussian Noise	0.9384
HFlip + Rotate + RandomBrightnessContrast	0.9371
Grid Distort	0.5548

- Flip, Rotate 등의 증강을 추가해서 실험을 해본 결과, 큰 성능 향상을 보지는 못했습니다. 대부분의 X-Ray 이미지는 촬영 시, 유사한 형태와 위치에서 촬영을 하기에 위의 증강들이 큰 효과를 보지 못한 것으로 생각합니다.

Resize 실험

- Model : DeepLabV3+
- BackBone : EfficientNet-B0
- scheduler: CosineAnnealingLR

	Dice Score
Resize: 512	0.9385
Resize: 1024	0.9613
Resize: 1536	0.9670
Resize: 2048	**0.9687**

- 이미지 사이즈 실험 결과, 이미지 사이즈가 클 수록 성능이 좋게 나오는 것을 확인 할 수 있습니다. 해상도가 높을 수록 객체의 디테일한 정보와 경계를 더 잘 보존하고, 원본 이미지의 해상도의 변화가 없기에 Resize를 통한 정보의 손실도 없기에 2048로 Resize한 것이 가장 좋은 성능을 보인 것 같습니다.

Preprocessing

- Hand bone 부분과 각 class의 경계선 부분을 강조하기 위해 Clahe와 Sharpen 증강을 Preprocessing으로 각각 사용해 보았습니다.

	Dice Score
Default	**0.9670**
Sharpen	0.9661
Clahe	0.9640

- 실험 결과로 Sharpen과 Clahe 증강을 사용하는 것보다는 원본 이미지 자체를 사용한 것이 가장 성능이 좋았습니다.

손등 뼈를 위한 처리

	Dice Score
손등 뼈 부분만 Crop	0.9718
이미지의 전체 학습 후 손등 부분만 추가학습	0.7880
Copy and Paste	**0.9730**

20%의 확률로 손등 뼈의 GT부분만 학습하는 증강을 추가하여 학습해보기도 했습니다. valid: 0.9738 -> 0.9778, test: 0.9718 -> 0.9682로 valid에서는 성능이 증가했지만 test에서는 많은 성능 하락이 있었습니다. valid과 test의 성능 차이가 큰 이유는 valid dataset 학습에 적용했던 증강이 그대로 적용되고 있기 때문이었습니다. 여러 augmentation도 추가해보았지만, 기존의 성능보다 뛰어나지지는 않았습니다.

추가적으로 손등 뼈의 겹치는 Class들의 성능이 다른 Class보다 낮아 이 부분을 해결하기 위해 겹치는 Class에 대해 Copy and Paste를 수행하였습니다. 실험 결과로 겹치는 Class들에 대해서는 성능이 상승하였지만, Paste하는 부분에서 다른 Class의 영역을 침범하는 경우가 발생하여 다른 Class들에서는 성능이 하락하는 경우도 있었습니다. 그럼에도 불구하고 전체적인 성능에 대해서는 가장 좋은 성능을 보였습니다.

3.3 Loss 실험

아래 표에서 default 모델은 아래와 같이 설정했습니다.

- 모델: DeepLabV3+
- 인코더: efficient-b0
- Augmentation: resize 512, ch 1
- scheduler: CosineAnnealingLR
- epoch: 40

	BCE Loss	Focal + Dice	Focal0.7 + Dice0.3	BCE + Dice
valid	0.9423	0.9489	0.9498	0.9442
test	0.9384	0.9449	0.9456	0.9379

가장 성능이 많이 오른 Combine Loss는 Focal Loss에 weight를 0.7, Dice에 0.3을 주었을 때 였습니다. 이때 학습에서 성능이 오르는 속도도 확연하게 빨랐습니다.

IoU Loss도 있었습니다. DICE와 비교해서 class가 위치하는 것보단 위치하지 않는 것에 좀 더 초점을 맞추는 것으로 알고 있어서 DICE보다 성능이 떨어질 것으로 예상하였고, 시간 관계상 실험은 하지 않았습니다.

3.4 threshold 분석

test 과정에서 threshold를 다르게 하면 성능이 얼마나 달라지는지 확인해보았습니다.
default threshold는 0.5입니다.

threshold	0.01	0.4	0.5	0.6	0.9	0.9999
test	0.9678	0.9724	0.9725	0.9725	0.9719	0.9587

threshold를 0.4와 0.6으로 test 결과를 보았을 때, 성능의 차이가 거의 없었습니다. 이를 이상하게 여겨 0.9의 threshold로 test를 해보았는데, 약간의 성능이 하락만 모습이었습니다. threshold가 0.01, 0.9999일 때도 확인해보면 threshold가 극단적임에도 성능이 상당히 높은 모습입니다.

threshold를 조정해도 성능의 차이가 미미한 원인을 알기 위해서 모델이 1개의 class에 대해 예측한 결과를 sigmoid 함수를 사용하기 전의 상태로 txt 파일을 얻어 살펴보았습니다. 약 4.5 값일 때 sigmoid의 값이 0.99가 되는데, class의 경계 부분은 대부분 -5 ~ +5의 값을 가지고 있었고 그 외의 부분은 -5보다 더

작거나 +5 보다 더 큰 값을 가지고 있었습니다. 생각보다 sigmoid 이후 0.1 ~ 0.9의 값을 가지는 픽셀들이 매우 적다는 것과, 이 픽셀들이 경계선 부근에만 존재한다는 것을 알 수 있었습니다.

txt 파일을 한 눈에 볼 수 있도록 시각화를 했더니 해당 class 부분만 나타나는 것이 아닌 경계선 부분이 두드러진 모든 손 뼈가 나타났습니다. 한 class에 대해서 예측한 경계선 부근은 다른 class 결과과에서는 매우 낮은 값이 나오는 것을 알았습니다. 이를 통해 모델이 class를 예측하는 과정에서 class마다 독립적으로 예측하는 것이 아니고 상호관계가 있다는 것을 경험할 수 있었습니다.

3.5 앙상블 실험

Hard Voting vs. Soft Voting

Model	**Backbone**	**Image Size**	**Dice**
DeepLabV3+	EfficientNet-B3	2048	0.9730
DeepLabV3+	Xception-41	1536	0.9690
DeepLabV3+	Xception-71	1536	0.9675
Hard Voting			0.9727
Soft Voting			**0.9735**

Soft Voting과 Hard Voting을 비교한 결과입니다. Hard Voting의 경우 오히려 단일 모델 기준 가장 성능이 좋았던 결과보다 떨어졌고, Soft Voting은 모든 단일 모델의 결과보다 향상된 것을 확인 할 수 있었습니다. 그렇기에 최종 앙상블 시에 **Soft Voting**을 사용하기로 결정 하였습니다.

최종 **Ensemble**

Ensemble Model	**BackBone**	**Image-size**	**Dice**
DeepLabV3+	ConvNext-Small	2048	0.9711
UNet++	InceptionNext-Small	2048	0.9729
UNet++	InceptionNext-Tiny	2048	
DeepLabV3+	Xception-41	1536	0.9690
DeepLabV3+	EfficientNet-B3	2048	0.9730
UNet++(Decoder Channel:[1024, 512, 256, 128, 64])	EfficientNet-B3	2048	0.9737
UNet++(Decoder Channel:[2048, 1024, 512, 256, 128])	EfficientNet-B3	2048	0.9734
Soft Voting Result(Threshold 0.5)			**0.9754**

UNet++모델과 DeepLabV3+ 모델을 기반으로 Soft Voting 앙상블을 진행하였습니다. **BackBone**, **Image Size**, **Decoder Channel** 등을 바꾼 모델들을 앙상블하여 Feature 추출, 이미지의 정보를 얼마나 세부적으로 학습하는지 등의 다양성을 주었고자 하였습니다. 이를 통해 최종적으로 0.9754라는 기존의 단일 모델들보다 더 나은 성능을 낼 수 있었습니다.

Ensemble Threshold 실험

Threshold	**Dice**
0.3	0.9747
0.5	**0.9754**
0.6	0.9752

추가적으로 앙상블 시에 Threshold값의 변화를 주어 실험을 진행하였습니다. Threshold를 0.5를 기준으로 0.5보다 낮은 0.3과 높은 0.6으로 실험을 진행하였을 때, 기존 Threshold(0.5)값으로 했을 때보다 성능이 하락한 것을 확인할 수 있습니다. 결론적으로 **Threshold 0.5**가 TP, FP, FN간의 균형을 가장 적절히 조절한 임계값임을 확인 할 수 있습니다.

3.6 모델 리서치

본 프로젝트처럼 X-ray나 고해상도의 이미지 데이터를 취급하는 대회에서 상위권이 주로 사용했던 모델 및 솔루션을 조사했습니다.

3.6.1 Segment multi-organ functional tissue units

본 대회는 2022년 캐글에서 진행됐던 세포를 Segment하는 대회로 세포 단면 이미지를 다루고 3000 by 3000 또는 4500 by 4500 해상도의 데이터로 구성되어있습니다. 상위권을 달성한 유저들이 사용한 모델과 인코더는 아래와 같습니다.

- 모델: SegFormer, Unet, Unet++
- 인코더: Efficientnet, Convnext, Coat, Mit

주요 특징으로 인코더로 Efficientnet-b7, Convnext-large와 같은 대형 모델을 사용했습니다. 또한 앙상블할 모델을 확보하기 위해 인코더를 중점으로 단일 모델로 다양한 인코더를 학습했습니다.

3.6.2 Track healthy organs in medical scans to improve cancer treatment

본 대회는 2022년 캐글에서 진행됐던 장기를 Segment하는 대회로 MRI 이미지를 다루고 최대 360 by 310 해상도의 데이터로 구성되어있습니다. 상위권을 달성한 유저들이 사용한 모델과 인코더는 아래와 같습니다.

- 모델: Unet, UperNet
- 인코더: Efficientnet, Swin, Convnext

주요 특징으로 앞에서 소개한 대회와 유사하게 대형 모델을 사용했고 모델을 1개로 고정하고 다양한 인코더로 학습해 앙상블을 했습니다.

3.6.3 Segment vasculature in 3D scans of human kidney

본 대회는 2024년 캐글에서 진행됐던 혈관을 Segment하는 대회로 CT 이미지를 다루고 최소 2000 by 2000 해상도의 데이터로 구성되어있습니다. 상위권을 달성한 유저들이 사용한 모델과 인코더는 아래와 같습니다.

- 모델: Unet, Unet++,
- 인코더: Efficientnet, Convnext, Maxvit, Seresnext

주요 특징으로 Unet 시리즈의 모델을 주로 사용하고 성능 향상을 위해 SCSE 기법을 사용했습니다. SCSE는 디코더에서 업샘플링 또는 Skip connection을 할 때 중요한 feature를 강조해 성능을 개선해주는 역할을 합니다.

대회에서 사용했던 모델 리서치를 하면서 두 가지 가설을 만들었습니다. 본 프로젝트의 이미지가 고해상도인 것을 고려해 인코더로 대형 모델을 사용하는 것이 일반적인 상황과 비교해 더 효과적일 것이라는 첫 번째 가설을 세웠습니다. 또한 SOTA를 기록하는 모델을 사용하지 않고 Unet 같은 비교적 최신이 아닌 모델을 사용해도 준수한 성능을 기록할 것이라는 두 번째 가설을 세웠습니다.

4. 자체 평가

4.1 잘한 점들

- Github를 통해 코드 공유가 원활히 되었습니다.
- 또한 Notion을 통해 각자의 의견들을 제시 및 공유하고, 실험 결과에 대한 내용도 공유하여 같은 실험을 반복하는 일이 없었습니다.
- Semantic Segmentation 관련 논문을 서칭 및 리뷰하며, 아이디어를 얻으려고 시도하였습니다.
- EDA, 리서치, 증강 실험 등 역할을 적절히 분담해 효율적으로 실험을 하였습니다.
- upstage 리더보드에 신경쓰지 않는 방향으로 프로젝트를 진행했는데, 최종 순위 기준 3위로 상위권을 달성했습니다.
- 모델을 일부 개량해보고자 하는 시도가 있었고, 활발한 의견 공유가 이루어졌습니다.

4.2 시도했으나 잘 되지 않았던 것들

- 손등 부분의 성능을 올리기 위해 여러 시도를 해보았지만, 만족할 만한 성능 향상을 이뤄내지 못했습니다.
- 논문 리뷰를 통해 알게 된 PerPixel Contrastive Loss를 시간 관계 상 시도해 보지 못했습니다.
- 프로젝트 막바지에 CRF를 진행하려 했지만, 제대로 구현하지 못하였습니다.
- 다양한 인코더를 실험하기 위해 torchseg 라이브러리를 사용했지만, 에러를 모두 해결하지 못해 일부 인코더만 사용했습니다.

4.3 아쉬웠던 점들

- PerPixel Contrastive Loss, CRF 등 여러 아이디어가 있었지만, 모두 수행해보지 못한 점이 아쉬운 것 같습니다.
- 여러 증강을 시도했지만, 만족할 만한 성능 향상을 이뤄낸 증강을 찾지 못한 점이 아쉽습니다.
- Streamlit을 통해 Ground Truth와 예측 결과를 비교하려 하였지만, 픽셀 별로 구별되기에 눈으로

확인하기 어려워 제대로 사용하지 못한 점이 아쉬운 것 같습니다.

- 메모리가 부족한 부분에 대해서 auto mixed precision 말고 더 알아볼 생각을 못한 것이 아쉽습니다.
- 증강으로 성능 향상을 이룬 팀들을 보고 난 후 베이스라인 코드를 꼼꼼히 보지 못한 것 같아 아쉽습니다.

4.4 프로젝트를 통해 배운 점 또는 시사점

- 협업 Tool을 좀 더 잘 쓰게 되었습니다.
- Segmentation 을 통해 좀더 다양한 모델을 사용할 수 있게 되었습니다.
- 모델이 상대적으로 정확히 분류하지 못 하는 클래스에 대해 이미지를 확인하면서 어떻게 접근해야할지 심도 깊게 고찰하는 시간을 가질 수 있었습니다.

개인 회고록

이재건

초기 세팅에 많은 노력을 기울였습니다. 처음 세팅이 잘 이루어져야 이후 빌드 과정이 수월하고 협업도 원활하다고 생각했기 때문입니다. 하지만 여전히 부족한 점이 있었습니다. 예를 들어, 설정(config)을 하나의 파일에 모두 담다 보니 혼란이 생기고 오류가 자주 발생했습니다. 또한, 코드를 모듈화하지 않아 다른 코드와의 충돌을 방지하려는 과정에서 많은 문제가 있었습니다. 처음부터 코드를 모듈화하고 체계적으로 구성했다면 더 효율적이지 않았을까 하는 아쉬움이 남아있습니다. 그럼에도 불구하고, 세그멘테이션 관련 코딩, GitHub와 Notion을 활용한 협업, 그리고 오류를 통해 배운 점들을 통해 한 단계 더 성장할 수 있었다고 생각합니다.

김동욱

이전 프로젝트에 비해 코드 관리 측면에서 많이 배웠습니다. 이전과 달리 단순히 Github에 commit만 날리는 것이 아닌 Issue와 Pull Request를 적극 활용하여, 서로의 코드를 공유 및 점검하는 과정을 통해 Github를 이전보다는 좀 더 잘 사용할 수 있게 된 것 같습니다. 또한 이런 저런 문제들이 있었지만, 모듈화를 통해 코드를 작성해 본 경험도 좋았습니다.

또한 하나의 라이브러리만 사용하는 것이 아닌 직접 좋은 모델을 찾아보고, 그걸 구현하기 위한 라이브러리나 방법을 찾아보는 경험도 좋았습니다. 그리고 특정 문제를 해결하기 위해 각자 다양한 의견을 내어 여러 방법으로 실험을 진행해본 점도 협업 관점에서 좋은 경험을 한 것 같습니다.

아쉬운 점으로는 대회 막바지쯤에 가서 증강 관련 코드가 잘 못 되었다는 점을 깨달았다는 것입니다. 이 때문에 증강 실험이 제대로 된 것 같지 않아 찜찜함이 남아있습니다. 또한 CRF도 적용시켜보려 했지만 제대로 되지 못한 점 또한 아쉬움으로 남는 것 같습니다. 또한 초기 Streamlit을 구현하였을 때, 팀원들의 needs를 잘못 파악하여 다른 것을 구현한 적이 있습니다. 이로 인해 기존의 계획했던 기간 보다 더 늦게 완성된 점도 아쉬운 것 같습니다.

황은섭

이번 프로젝트에서 저의 목적은 모든 팀원이 활용할 수 있는 방법을 찾고 도움을 주는 것이었습니다. 베이스라인의 코드가 불편한 점이 있으면 곧바로 수정했습니다. 덕분에 전보다는 좀 편하게 베이스라인을 사용할 수 있었을거라 믿습니다. 이전 프로젝트에서 하던대로 근거를 기반으로 가설을 세우고 실험을 하고, 결과를 면밀하게 예측하려고 했습니다. 실험을 여러 번 진행했는데 결과가 큰 효과를 보이지는 못했었습니다. 실험을 더 끈질기게 했으면 좋은 결과를 얻을 수도 있다는 생각이 자꾸 듭니다. 앞으로는 실험을 쉽게 포기하지 않을겁니다. 정말 더 이상 할 것이 없다고 판단되었을 때 다른 실험을 넘어가고 싶습니다. 모델 개선을 위해 제가 시도한 대부분의 방식이 모델 개선으로 이어지지 않았습니다. 채널 1개를 유지하며 학습한 것, fp16 계산, 원본 이미지의 크기로 학습, 무거운 인코더를 사용하기. 정도가 있었습니다.

AMP로 학습하면 세부적인 계산이 힘드니 LoRA로 파인튜닝을 해서 성능을 더욱 올려보고 싶었습니다. 하지만 decoder에는 LoRA를 적용할 수 없었으며 총 10%의 파라미터를 학습했지만 성능의 향상이 보이지 않았습니다. LoRA에서 지원하지 않는 block들이 있어 segmentation에서는 활용하기가 힘들다는 점을 알았습니다. 메모리 사용량도 줄어들 줄 알았지만 그렇지 않았습니다. 또 Crop을 해서 손등 부분에만 집중하여 뛰어난 성능을 가지는 모델을 얻고 싶었습니다. 하지만 모든 뼈에 대해 학습한 모델과 손등 부분에 대해 성능이 비슷했습니다. 이렇게 얻은 손등 모델을 기존의 모델과 앙상블을 했을 때 성능이 오르긴 했지만 모든 손뼈에 대해 학습한 다른 모델과 앙상블 하는 것에 비해 효과적이지 않았습니다.

깃허브를 사용하는 방법을 좀 더 익혔습니다. 다만 코드 협업은 처음이다 보니 다른 작업을 하는 과정에서 제 코드가 에러를 일으킨 적이 있었습니다. 프로젝트의 후반에는 노션이나 코드 협업에 좀 미흡했던 것 같습니다.

박정욱

이전 프로젝트까지는 Github에 미숙하고 어려움을 겪어 100% 활용하지 못했지만 이번에 프로젝트를 진행하면서 convention, issue, pull request, templete 등 여러 개념들과 기능을 알게 됐고 팀원들 덕분에 많이 사용할 수 있었습니다. 미지의 영역이었던 github에 한 걸음 더 가까워진 기분이 들었습니다. 또한 팀원들이 만든 초기 코드 세팅을 바탕으로 원하는 기능을 추가 및 수정을 하면서 협업 경험을 쌓았습니다.

새로운 라이브러리를 적용하면서 다양한 에러가 발생해 팀원들이 실험하는데 문제가 되지 않게끔 최대한 모든 에러를 잡아보려 했지만 생각했던 것보다 경우의 수가 너무 많아서 모두 해결하지 못했습니다. 예상했던 대로 실험 도중 에러가 발생해 정상적으로 실험하지 못한 팀원이 나왔고, 이 부분에서 수월하게 진행되지 못해 아쉬웠습니다. 다음 프로젝트에서는 새로운 기능을 추가하기 전 테스트를 여러 번 진행하고 이에 대한 가이드라인을 작성해 협업에 혼란을 주지 않도록 노력하고 싶습니다.

손등 클래스의 성능 개선을 위해 클래스 수를 조절하거나 Crop을 추가하면서 여러 실험을 진행했지만 결과적으로 유의미한 기록은 내지 못했습니다. 하지만 실험을 진행하면서 이미지에서 손등 부분을 분석하고 모델이 어떻게 예측을 하고 있을지 고민해보면서, CV의 특성에 맞게 고찰을 진행한 것 같아 기억에 많이 남았습니다.

김세린

라이브러리에만 의존하기 보다는 모델이 예측하기 어려워하는 부분인 손목에 집중해 유의미한 모델 개량을 해보고자 노력했습니다. 손목뼈를 crop한 후 resize한 상태의 input이 주어진다는 점을 고려하여, 인코더의 stride를 줄이고 디코더의 dilation rate을 default 값보다 작은 값들로 구성했습니다. 이러한 조정은 모델이 국소적인 부위에 더 집중하도록 만들 것이라는 가설에서 시작했고, 실험 결과 crop된 부분에 한해서는 성능 향상을 관찰할 수 있었습니다. 그러나 모델이 하나의 파이프라인으로 2-branch 형식으로 동작할 수 있도록 설계하지 못했고, 더 다양한 실험을 시도해보지 못한 것이 아쉽습니다.

이번 프로젝트에서는 팀원들의 도움을 받아 pull request와 branch 생성 후 작업하는 것에 대해 익힐 수 있었습니다. 모두가 적극적으로 깃허브와 관련된 이슈를 함께 해결해주기 위해 도와주고, 또 협업 툴로서 적극 활용하는 모습이 좋았습니다.

김재진

새롭게 팀이 구성되면서 초기에 시간이 많이 들어갈것이라 생각했던 요소들에서 시간이 지연되지 않아 빠르게 시작할 수 있었다고 생각합니다. 하지만 프로젝트 측면에서는 개인적으로는 아쉬움이 많이 남았습니다. 한 라이브러리의 사용이 원활하지 않아 많은 시간이 소요되었고 현재 학습된 모델에 대해 분석할 시간이 부족해 명확한 근거에 의한 실험이 부족했다고 생각합니다. 일정이 정해진 상황에서 예상보다 지연이 되면 어떤 행동을 해야하는지 깊게 생각해봐야 할 문제인거 같습니다.