Hand Bone Image Segmentation

CV-17

김기수 김유경 김준현 여희진 이은아 정권희

Ⅰ. 프로젝트 개요

1. 프로젝트 주제

뼈는 우리 몸의 구조와 기능에 중요한 영향을 미치기 때문에, 정확한 뼈 분할은 의료 진단 및 치료 계획을 개발하는 데 필수적임.

딥러닝 기술을 이용한 뼈 Segmentation은 많은 연구가 이루어지고 있으며, 다양한 목적으로 도움을 줄 수 있음.

모델은 29개 클래스를 분류하며, 손가락 뼈(finger-1 ~ finger-19)와 손목 및 팔뼈(Trapezium, Trapezoid, Capitate, Hamate, Scaphoid, Lunate, Triquetrum, Pisiform, Radius, Ulna)로 구성됨.

2. 프로젝트 팀 구성 및 역할

- **김기수** : 데이터 시각화, WandB 셋팅, 모델학습 시간 실험, 증강 실험, mmesegmentation 셋팅 및 코드 작성/모델 학습
- **김유경** : EDA, Loss 실험, Augmentation 실험, smp 모델 실험, 후처리, 앙상블
- **김준현** : Git, Encoder Test, Augmentation, 앙상블
- **여희진** : EDA, smp model 실험, Augmentation, 앙상블
- **이은아** : Scheduler/Optimzer 실험, Curriculum Learning, Yolo 실험
- **정권희** : 베이스라인 코드, Swin-Unet

3. 베이스라인 구축

제공받은 베이스라인 코드를 모듈화를 통해 재사용성과 유지보수를 개선, 팀의 개발 환경과 요구사항에 최적화된 구조로 재구성하는 작업을 체계적으로 진행.

모델링 진행 시 파라미터 및 변경점을 손쉽게 변경할 수 있도록 코드 내 config 수정 방식과 argparse 두가지 방식 모두 적용.

```
── code
    ├── loss.py
    ├── model.py
    ├── config.py
    ├── dataset.py
    ├── inference.py
    ├── train.py
    └── requirements.txt
```

[그림 1-1] 베이스라인 구조

4. 협업 툴

4.1 Git 활용

main을 기본 브랜치로 설정 후 각자 작업에 대한 Issue 생성 후 해당 Issue에 대한 브랜치를 만들어 main에 merge하는 방식으로 설정.

1) Branch Convention

main	개발이 완료된 산출물이 저장될 공간
feat	기능을 개발하는 브랜치, 이슈별/작업별로 생성
setting	기본 설정, 기타 설정 시
bugfix	버그를 수정하는 브랜치

[표 1-1] Branch Convention

2) Commit Convention

feat	새로운 기능 구현
fix	버그, 오류 해결, 코드 수정
add	Feat 이외의 부수적인 코드 추가, 라이브러리 추가, 새로운 View 생성
del	쓸모없는 코드, 주석 삭제
refactor	전면 수정이 있을 때 사용
remove	파일 삭제
chore	그 이외의 잡일/버전 코드 수정, 패키지 구조 변경, 파일 이동, 파일 이름 변경
docs	README나 WIKI 등의 문서 개정
merge	#이슈번호
setting	코드 세팅 관련

[표 2-1] Commit Convention

Ⅱ. 프로젝트 수행 내용

1. EDA (Exploratory Data Analysis)

1.1 Meta Data

주어진 데이터셋에 대해 메타데이터(나이, 성별, 키, 체중)의 분포 및 비율을 분석한 결과는 다음과 같음.

1) 성별 분포

성별	인원수	비율 (%)
여성	288명	52.36%
남성	262명	47.64%
총합	**550명**	**100%**

[표 2-2] 성별 통계

meta_data.xlsx 파일에 기록된 인원 수(550명)를 통해 전체 이미지가 1,100장임을 추정할 수 있으나, 이번 대회에서는 Train 이미지 800개와 Test 이미지 288개로 구성된 총 1,088개의 이미지만 제공됨. 이를 기반으로 보면, 544명이 참여한 것으로 보이며, 주어진 메타데이터가 이전 데이터셋에서 파생된 것으로 추정됨.

2) 나이 분포

- 20-30세: 294명으로 가장 많은 비중 차지
- 10-20세: 3명으로 가장 적은 비중 차지

3) 성별별 데이터 분포

- 남성: 20대 중반에서 30대 중반의 데이터가 집중적으로 분포
- 여성: 연령대가 비교적 고르게 분포

4) Train/Test 데이터셋의 성별 비율 비교

데이터셋	남성 비율(%)	여성 비율(%)
Train	45.5%	54.5%
Test	52.8%	47.2%

[표 2-3] train/validation 별 성별 통계

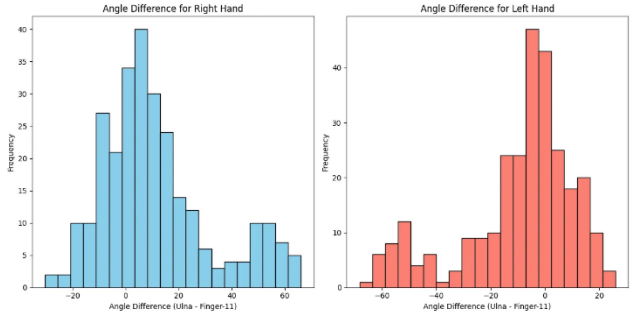
Train과 Test 데이터셋 모두 성별이 비교적 균등하게 분포되어 있음을 확인.

1.2 Image Data

1) Train 및 Test 데이터에서의 특이 이미지

- Train 및 Test 데이터를 살펴본 결과, 다음과 같이 특이 이미지를 확인할 수 있었음.
- 추가적인 강조 요소가 포함된 이미지 : 손톱,네일, 반지, 붕대 등이 강조된 소량의 이미지
- 손목이 꺾인 이미지 : Train 데이터의 약 11%, Test 데이터의 약 57%가 손목이 꺾인 이미지를 포함하고 있음.

2) 손목 회전 각도 분석



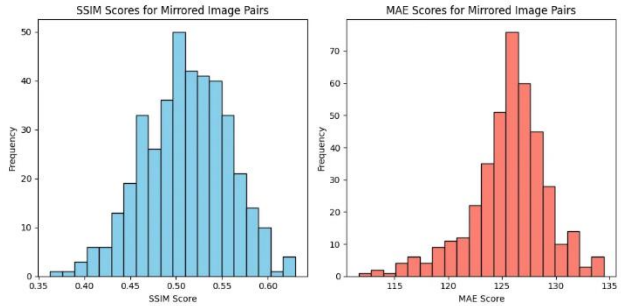
[그림 3-1] train 이미지 손목 각도 분석

손목이 꺾인 이미지를 분석하기 위해 팔목 뼈(Ulna)와 중지 손가락(Finger-11) 사이의 각도를 계산.

- 대부분의 이미지에서 양손의 회전 각도는 -20~20도 사이에 분포
- 일부 이미지는 40~60도로 회전된 경우가 확인됨.

Train 데이터와 Test 데이터에서 손목이 꺾인 이미지의 비율이 차이나는 점(Train: 11%, Test: 57%)과 이러한 이미지가 많은 점을 고려할 때, 이미지 회전(Rotate)을 활용한 데이터 증강 실험의 가능성을 제안.

3) 좌우 이미지 대칭 유사도 분석



[그림 3-2] SSIM 및 MAE

해당 데이터셋은 왼손과 오른손이 쌍을 이루는 구조로, 좌우 이미지 대칭 유사도를 확인함.

- SSIM : 평균적으로 0.5 부근에 분포
- MAE : 대부분 123~130에 분포

이를 통해, 일부 유사한 특징이 있지만 구조적 차이가 존재함을 확인했고, 이를 기반으로 Horizontal Flip을 활용한 데이터 증강 실험의 가능성을 제안.

2. 학습 최적화 실험

2.1 학습 시간 실험

주어진 baseline 코드 중 validation시 Dice 계산을 CPU로 계산하였고 이를 GPU로 변경한 후에도 validation time이 줄지 않는 문제 확인함. 문제를 해결하기 위해 batch size와 num workers, pin memory에 대한 값을 변경하여 실험을 진행함.

Batch size	num workers	pin memory	time
1	0	False	3m 10s
1	0	True	3m 5s
1	2	False	3m 30s
1	4	False	1m 53s
1	**8**	**False**	**1m 16s**
2	0	False	3m 10s
2	8	True	1m 20s
4	0	False	3m 7s
4	8	True	1m 20s
4	8	False	1m 18s

[그림 3-3] 요소 간 validation time 비교

Validation Dataloader의 batch size를 키우면 특정 클래스의 수렴속도가 매우 느린 현상도 존재하였기에 batch size 1, num workers 8이 가장 좋아보였고 이를 활용하여 다양한 실험을 하는데 시간을 줄일 수 있었음.

2.2 Loss 실험

Segmentation task는 픽셀 단위의 분류 문제로, 사용하는 Loss Function에 따라 모델 성능이 크게 달라질 수 있다는 가설을 기반으로 다양한 Loss Function에 대한 실험을 진행 (Unet++ Resnet50 backbone과 ImageNet pre-trained weight로 고정)

Loss	Val_Dice	LB_Dice
BCE Loss	0.9497	0.9492
Dice Loss	0.9436	-
IoU Loss	0.3953	-
Tversky Loss	0.2652	-
BCE + Dice Loss	**0.9558**	**0.9533**
BCE + IoU Loss	0.9421	0.9231

[표 4-1] Loss function 별 Dice 값

BCE + Dice Loss가 가장 우수한 성능을 기록했으며, 이는 BCE Loss의 픽셀별 차이를 명확히 계산하는 특성과 Dice Loss의 클래스 간 균형을 고려하는 특성이 결합된 결과로 판단됨.

2.3 Optimizer 실험

Unet++ 모델에서 ConsineAnneling를 사용하여 실험 진행

Optimizer	Val_Dice
adam	0.9495
adamw	0.9491
adamp	**0.9513**

[표 4-2] Optimizer 별 Dice 값

- AdamP: 가장 높은 val dice 0.9513을 기록하며, Adam과 AdamW보다 우수한 성능을 보임.
- Adam과 AdamW: 비슷한 성능을 보였으나, Adam이 소폭 우위를 점함.

2.4 Scheduler 실험

Unet++ 모델에서 scheduler 변경하여 실험 진행.

Scheduler	Val_Dice
x	0.9416
MultiStepLR	0.9030
CosineAnealingLR	**0.9514**
ReduceLROnPlateau	0.9504
CosineAnnealingWarmRestarts	0.9495
GradualWarmUp +ConsineAnnealing	0.9497

[표 4-3] Scheduler 별 Dice 값

- MultiStepLR: 학습 과정에서 최적점을 찾지 못하고 가장 낮은 성능을 보임.
- CosineAnnealingLR: 가장 높은 val dice 0.9514를 기록하며 학습을 극대화.
- RedisceLROnPlateau, ConsineAnnealingWarmRestarts, GradualWarmup+ConsineAnnealing: 성능이 비슷한 것을 확인.

3. 모델 실험

3.1 smp (segmentation_models_pytorch)

1) smp 내 모델 실험

SMP(segmentation models pytorch) 라이브러리 내 모든 모델들 실험 진행.

기본 설정값 : 인코더 resnet 50, 50 epoch, BCEDiceLoss, adamp, 1e-3, Consine Annealing, resize 512

Model	Val_Dice	LD_Dice
UnetPlusPlus	0.9447	-
Unet	0.9243	-
FPN	**0.9523**	**0.9461**

Linknet	0.9267	-
MAnet	0.9264	-
PAN	**0.9516**	**0.9480**
PSPNet	0.9239	-
DeepLabV3	0.9483	-
DeepLabV3Plus	**0.9536**	**0.9500**
UperNet	**0.9553**	**0.9502**

[표 5-1] 모델 별 Dice 값

DeepLabV3와 UperNet 모델의 성능이 우수했으므로, 이후 UperNet을 기준으로 추가 Encoder 실험 진행.

2) Encoder 실험

1) 모델 실험을 바탕으로 UperNet 기준으로 실험 진행.

Encoder	Val_Dice	LD_Dice
mobilenet_v2	0.9513	-
efficientnet-b0	0.9515	-
vgg19	0.9545	-
resnet152	0.9540	-
hrnet_w64	**0.9553**	**0.9534**
timm-resnest14d	0.9519	-

[표 5-2] Encoder 별 Dice 값 1

Encoder로 hrnet_64를 활용할 때 가장 우수한 성능을 보였으며, 해당 모델을 바탕으로 추후 실험 계획 수립.

Encoder	LD_Dice (Private)	LD_Dice (Public)
Efficientnet-b0	0.9453	0.9459
Efficientnet-b7	0.9489	0.9514

[표 5-3] Encoder 별 Dice 값 2

b7의 성능이 public 기준 0.0036 높게 나온 것을 확인할 수 있었음. 하지만, b7의 경우 모델 사이지가 커 batch size 2로 설정해야 서버 환경에서 학습을 진행할 수 있었음.

따라서, 추후에 이미지 사이즈를 키워 학습시킬 때 제한사항이 생겨 Efficientnet-b0을 사용.

3.2 MMSegmatation

1) Segformer(mit_b3/ImageNet1k)

Image_size	Augmentation	Loss	Epoch	Val_Dice	LD_Dice
1024	CLAHE	CE	100	0.9714	0.9685

[표 5-4] Segformer 실험 결과

Segformer은 Multi-scale feature map을 사용하여 transformer의 낮은 작은 물체 검출 성능과 큰 이미지의 높은 연산량의 단점을 개선한 모델.

Radius, Ulna와 같은 큰 크기의 객체와 trapezoid, pisiform과 같은 작은 객체가 존재했으면 2048x2048의 크기의 이미지의 input에 적합하다고 생각하여 모델을 선택함.

3.3 Swin-Unet

EDA 결과, 다음과 같은 이유로 Swin-Unet이 효과적일 것이라 판단 후 진행함.

- Swin Transformer 기반의 Hierarchical 구조가 2048크기의 데이터셋에 적합
- U-Net의 skip connection을 활용하여 low-level feature를 보존하면서, high-level feature와 결합하여 예측의 디테일을 강화
- Swin-Unet은 멀티스케일 윈도우 attention을 활용해, 손 뼈 이미지에 있는 다양한 크기와 형태의 객체를 효과적으로 처리
- Transformer 기반의 모델은 CNN에 비해 더 풍부한 특징 표현을 학습하기에 추후 앙상블에 유리

smp, MMSegmentation 라이브러리에 Swin-Unet을 구현할 자료들이 부족하여 논문과 공식 깃허

브를 바탕으로 직접 모델을 구현함.

모델을 구성하는 checkpoint의 window size가 7이므로 그 가중치를 활용하기 위해 이미지 사이즈를 1120으로 resize하여 점진적으로 가까운 사이즈로 키우는 것을 목표 설정.

Model	Augmentation	LD_Dice (Private)	LD_Dice (Public)
Swin-Unet	-	0.9462	0.9498
Swin-Unet	CLAHE	0.9499	0.9522

[표 6-1] Swin-Unet 실험 결과

증강없이 Swin Unet을 실험하였을 때, 리더보드 기준 Public 0.9462, Private 0.9498 정도로 예상보다 저조한 성적을 보임.

추가로 CLAHE 증강을 적용해본 결과, 리더보드 기준 Public 0.9499, Private 0.9522 정도로 약 3% 정도 증가.

3.4 YOLO-Seg

Model	Val_Dice
YOLOv8x-seg	**0.9421**
YOLOv9e-seg	0.9380
YOLO11l-seg	0.9392

[표 6-2] YOLO 버전 별 실험 결과

실험 결과, YOLOv8x-seg가 가장 우수한 성능을 보임.

4. 증강 실험

4.1 Resize

원본 이미지의 해상도가 (2048, 2048)이므로 해상도에 따라 정보 손실 가능성이 존재하고 성능 차이가 발생할 것이라는 가설을 세우고 Resize 실험 진행.

Resolution	Model	Val. Dice	LB Dice
512 x 512	DeepLabv3Plus (resnet350/ imagenet)	0.9536	0.9500
1024 x 1024	DeepLabV3Plus (resnet152/Imagenet)	0.9690	0.9658
1536 x 1536	**DeepLabV3Plus (resnet152/Imagenet)**	**0.9723**	**0.9689**

[표 6-3] 해상도 별 실험 결과

실험 결과, 이미지 사이즈가 클수록 좋은 성능을 보임.

4.2 CLAHE

Image Data EDA에서 손등뼈가 겹치는 이슈를 개선하기 위한 방법으로 CLAHE 적용

CLAHE	Val_Dice
baseline	0.9459
clip_limit=2, tile_grid_size=(8, 8)	0.9150
clip_limit=3.5, tile_grid_size=(8, 8)	**0.9491**
clip_limit=4.5, tile_grid_size=(8, 8)	0.9420
clip_limit=[3.0, 3.0] tile_grid_size=(8, 8)	0.9369

[표 6-4] CLAHE 실험 결과

전체적인 성능 0.0048 향상. 상대적으로 낮은 score을 보였던 손등뼈 중 하나인 Pisiform의 성능이 0.05 향상

4.3 Horizontal Flip

한 사람의 양 손을 쌍으로 가지고 있었기에 Horizontal Flip을 적용.

	Val_Dice	LD_Dice
baseline	0.9442	0.9210
Horizontal Flip	**0.9526**	**0.9418**

[표 6-5] Horizontal flip 실험 결과

실험 결과, 0.0208정도 유의미한 성능 향상

4.4 Canny

뼈를 검출하는 task는 경계선을 가지고 충분히 학습할 수 있어 오히려 경계선 내부의 픽셀은 학습하는데 방해할 것이라는 가설을 세움.

	Val_Dice	LD_Dice
baseline	0.9442	0.9210
Canny	**0.9296**	**0.8860**

[표 7-1] Canny 실험 결과

이에, Canny Edge 검출 기법을 적용하여 학습하였으나 성능이 감소하는 결과를 보임.

4.5 Gray Scale

X-ray 이미지에서 1 channel gray scale image로 변환하는 것이 효과적이기에 Gray Scale 적용.

4.6 Index별 손목 Rotate 실험

손목 회전 데이터를 시각적으로 분석한 결과, 왼손은 왼쪽으로, 오른손을 오른쪽으로 회전되는 특징을 확인.

이를 기반으로, 손목이 돌아가지 않은 데이터에 대해 손 방향별 Rotate 증강 실험을 진행.

- 왼손 : 왼쪽으로 회전 → limit = (10, 30)
- 오른손 : 오른쪽으로 회전 → limit = (-30, -10)

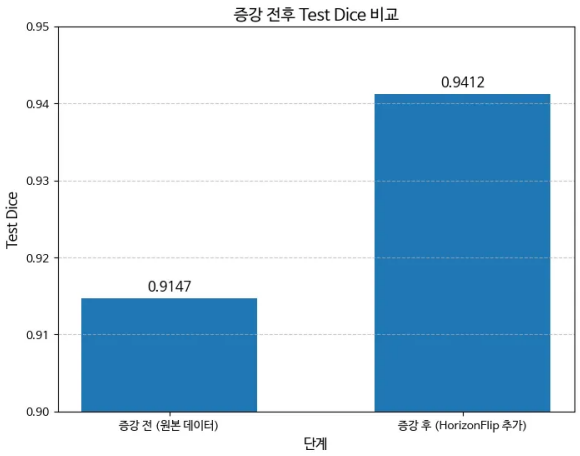
Model	Loss	Augmentation	ImageSize	Val_Dice
DeepLabV3Plus(resnet152/Imagenet)	BCEDiceLoss	-	1024	0.9690
DeepLabV3Plus(resnet152/Imagenet)	**BCEDiceLoss**	**Rotate (p=0.3)**	**1024**	**0.9670**

실험 결과, Rotate 증강 적용 후, 0.2% 성능 소폭 하락 확인하였음.

4.7 Curriculum Learning

커리큘럼 러닝을 도입하여 손목 회전 데이터에 대한 모델의 학습 성능을 개선하고자 실험을 진행.

모델이 학습하기 쉬운 데이터로 시작해, 점진적으로 학습 난이도가 높은 데이터를 추가하도록 설계.



[그림 7-1] Curriculum Learning 실험 결과

데이터 증강 : 손목 회전 데이터에 HorizonFlip 증강 기법을 적용하여 데이터를 추가 생성.

- 첫 번째 단계 : 원본 데이터만 사용하여 모델의 기본 학습 진행
- 두 번째 단계 : 원본 학습 모델을 기반으로, HorizonFlip 증강된 손목 회전 데이터만을 추가 학습.

커리큘럼 러닝과 데이터 증강을 결합한 학습 전략은 손목 회전 데이터에 대한 성능 개선에 효과적임을 확인.

5. 후처리

Inference 를 시각화한 결과, 29개의 class가 각각 하나의 Contour로 Segmentation 되어야 하지만, 일부 class가 여러개의 Contour로 분리되는 경우를 확인함.

이를 해결하기 위해, 불필요한 영역을 제거하는 Negative Sample Masking 기법을 도입함

Polygon으로 영역을 연결 후, 가장 면적이 큰 Contour만을 선택하고 나머지는 제거함.

적용 전 LB Dice	적용 후 LB Dice	변경된 row 수	변경된 row 비율
0.9658	0.9660 (+0.0002)	686	8%
0.9688	0.9692 (+0.0004)	281	3.4%

[표 8-1] 후처리 결과

후처리 적용 후 약 0.02%-0.04% 성능 향상 확인함.

6. 앙상블 실험

앙상블은 Hard Voting, Soft Voting 방법으로 진행.

6.1 다른 Fold 앙상블

K-Fold Cross Validation의 장점을 활용하여 K개의 Fold에 대해 각기 다른 증강 실험을 진행한 후 앙상블 수행.

모델은 DeepLabV3+ (resnet152, imagenet)로 진행.

K-fold	Augmentation	Val_Dice	LB_Dice
fold0	x	0.9690	0.9658
fold1	HorizontalFlip	0.9692	-
fold2	CLAHE	0.9717	0.9681
fold3	CLAHE, HorizontalFlip	0.9696	-
fold4	Sharpen	0.9674	-
5-Fold Ensemble			**0.9698**

[표 8-2] Fold 실험 결과

실험 결과, 각 Fold의 Val_Dice 값보다 높은 LB_Dice(0.9698) 값을 관찰.

6.2 같은 실험, 다른 Epoch 앙상블

Epoch	fold	voting	LD_Val
100	0		0.9685
80, 85, 90, 95, 100	**0**	**hard**	**0.9687**

[표 8-3] Epoch 별 실험 결과

Segformer (mit_b3/ImageNet1k) 모델을 사용해 이미지 크기 1024로 실험한 결과, Epoch 100에서 가장 높은 Val_Dice를 기록. 이와 비교해 Val_Dice 상위 5개 Epoch의 결과를 앙상블한 결과는 0.0002 더 높은 LD_Dice를 달성.

6.3 같은 모델, 다른 증강 앙상블

RGB로 학습한 모델과 Gray로 학습하 모델을 Soft Voting으로 앙상블 진행.

실험 결과, 한 종류의 모델로 LD_Dice 0.9697의 스코어를 얻음.

6.4 각 클래스 별 다른 모델 앙상블

29개의 클래스 각각에 대해 Validation Dice 성능이 가장 높은 모델들의 에측 결과를 선택적으로 결합하여 최종 앙상블 결과를 생성하는 방법.

클래스별로 가장 적합한 모델의 예측을 활용하

여 전반적인 성능을 극대화하고자 함.

Ⅲ. 결론

1. 최종 모델

① DeepLabV3Plus (Resnet152)	kfold1: 0.9677(LD)
	kfold2: 0.9677(LD)
	kfold2: 0.9689(LD)
	kfold3: 0.9696(Val)
② Uper (HRNet-w64)	kfold0: 0.9608(LD)
③ Unet (efficientnet-b0)	kfold0: 0.9680(LD)

[표 9-1] 최종 모델

2. 앙상블 결과

Candidates	LB score
① + ②	0.9707
① + ② + ③	**0.9709**

[표 9-2] 앙상블 결과 1

Class별 최고점을 포함한 모델들을 soft voting 진행

최종적으로 앙상블을 통하여 단일모델 리더보드 최고점이었던 0.9689 대비 0.002 성능 향상

No.	Class	14_unet++	75_unper_30e	75_unper_40e	83_DeepLabV3	91_fold2	97_unet	103_DLv3
1	finger-1	0.963	0.9659	0.9651	0.9659	0.9674	0.9638	0.9666
2	finger-2	0.979	0.9817	0.9818	0.9809	0.9811	0.9808	0.9817
3	finger-3	0.9825	0.9849	0.9853	0.9858	0.9854	0.9847	0.986
4	finger-4	0.967	0.9709	0.971	0.9723	0.9753	0.971	0.9747
5	finger-5	0.9722	0.9739	0.9742	0.9737	0.9766	0.9747	0.9757
6	finger-6	0.984	0.9853	0.9853	0.9854	0.9864	0.9846	0.9862
7	finger-7	0.9813	0.9823	0.9816	0.9836	0.9841	0.9819	0.9843
8	finger-8	0.9724	0.9751	0.9758	0.9655	0.9767	0.9739	0.9769
9	finger-9	0.9765	0.9788	0.9786	0.9727	0.9805	0.9772	0.98
10	finger-10	0.9844	0.9867	0.9867	0.9861	0.9874	0.9861	0.9875
11	finger-11	0.9754	0.9769	0.9766	0.9771	0.9779	0.9766	0.9784
12	finger-12	0.9652	0.9751	0.9749	0.97	0.9754	0.9731	0.9755
13	finger-13	0.9747	0.9781	0.9776	0.9748	0.9789	0.9771	0.9784
14	finger-14	0.9808	0.9848	0.9847	0.9847	0.9864	0.9833	0.9862
15	finger-15	0.9752	0.9793	0.9787	0.9793	0.9801	0.9785	0.9808
16	finger-16	0.9661	0.9697	0.9693	0.9678	0.9704	0.9673	0.9705
17	finger-17	0.966	0.9684	0.9678	0.9667	0.9708	0.9684	0.97
18	finger-18	0.9819	0.9825	0.9825	0.9826	0.9838	0.982	0.9837
19	finger-19	0.9822	0.9842	0.9845	0.9841	0.9842	0.9833	0.9839
20	Trapezium	0.9352	0.9482	0.9495	0.9487	0.9543	0.9475	0.9536
21	Trapezoid	0.8934	0.9144	0.918	0.9187	0.9216	0.9207	0.9228
22	Capitate	0.9526	0.9668	0.9658	0.9645	0.9667	0.9629	0.9678
23	Hamate	0.9412	0.9532	0.9533	0.9532	0.9495	0.9502	0.9507
24	Scaphoid	0.9639	0.9689	0.9708	0.9712	0.9705	0.9724	0.9734
25	Lunate	0.946	0.9622	0.9607	0.9604	0.9592	0.9599	0.9619
26	Triquetrum	0.9382	0.9527	0.951	0.95	0.9551	0.9553	0.9585
27	Pisiform	0.9081	0.9083	0.9053	0.8985	0.9144	0.9138	0.9237
28	Radius	0.9897	0.9894	0.9901	0.9897	0.9899	0.9894	0.99
29	Ulna	0.9891	0.9882	0.9882	0.9872	0.9884	0.988	0.9881
Val-dice avg		0.9651	0.9702	0.9702	0.969	0.9717	0.9699	0.9723
LD dice		0.9614	0.9608	-	0.9658	0.9681	0.968	0.9689

[표 9-3] 모델 및 클래스 별 성능 비교