Semantic Segmentation Wrap-Up Report

부스트캠프 **AI Tech 5**기 **CV-10** 형셋누나하나

김보경(T5033), 김정주(T5052), 양재민(T5123), 임준표(T5175), 정남교(T5188)

1. 프로젝트 개요

a. 개요

의료분야에서 Segmentation task는 진단 및 치료 계획을 개발하는 데 필수적이다. Bone Segmentation은 뼈의 형태나 위치가 변형되거나 부러지거나 골절 등이 있을 경우, 문제를 정확하게 파악하여 적절한 치료를 시행할 수 있다. 또한 수술 계획을 세우거나 의료 장비에 필요한 정보를 제공하고 교육 목적으로도 사용될 수 있다. 이번 프로젝트를 통해 뼈를 정확하게 Segmentation하는 모델을 개발함으로써 의료 분야에 다양한 목적으로 도움이 되고자 했다.

b. 환경

- 팀 구성 및 컴퓨팅 환경 : 5인 1팀, 인당 V100 GPU 및 AI Stages 서버
- 개발 환경 : VSCode, Jupyter Notebook
- 개발 언어 : Python, PyTorch
- 협업 툴 및 기타 : Slack, Notion, Github, wandb

c. 데이터셋 구성

3-channel 2048 * 2048 size의 Images(.png) & Labels(.json)

사람 별로 양손 촬영 이미지 2장이 존재, 총 29개의 Class Label 구성

“finger-1”, “finger-2”, “finger-3”, “finger-4”, “finger-5”, “finger-6”, “finger-7”,

“finger-8”, “finger-9”, “finger-10”, “finger-11”, “finger-12”, “finger-13”, “finger-14”,

“finger-15”, “finger-16”, “finger-17”, “finger-18”, “finger-19”, “Trapezium”,

“Trapezoid”, “Capitate”, “Hamate”, “Scaphoid”, “Lunate”, “Triquetrum”,

“Pisiform”, “Radius”, “Ulna”

Train : 800개의 Images, Label 존재 O

Test : 300개의 Images, Label 존재 X

2. 프로젝트 팀 구성 및 역할

공통	**Data EDA, Baseline Test**
김보경	Baseline code 리팩토링, sml Model 구현 및 실험, Loss weight 실험, Optuna 구현, inference ensemble 구현
김정주	Augmentation 실험, Model 설계/성능 측정, Loss 설계/실험, Visualization 구축
양재민	Augmentation 실험, smp Model 구현 및 실험 Scratch Model 구현, 실험
임준표	Augmentation 실험, Model 구현 및 실험, inference augmentation, 학습 속도 가속화
정남교	kfold, ensemble 구현, smp Model 구현 및 실험, augmentation 실험

3. 프로젝트 수행 절차 및 방법

a. Data EDA

본 Competition에서 사용하게 될 dataset의 성질을 파악하여 그 특성에 최적화된 Segmentation 모델을 설계하기 위해 Data EDA를 수행하고자 하였다. 특히 Augmentation 기법 중 Normalize의 argument로 입력될 dataset의 평균값과 표준편차를 파악하여 적절히 활용하고자 계획하였다.

b. Augmentation Ablation Study

1) Augmentation 조합

어떠한 Test dataset에 대해서도 Robust한 Segmentator를 설계하기 위해서는 Augmentation이 필수적이며, 본 Competition과 같이 Train dataset이 부족한 상황에서 Generalization 성능을 향상시키기 위해서는 적절한 Augmentation 조합을 찾아 적용하는 것이 필요하다. 따라서 Resize를 포함해 다양한 Spatial-Level Transform과 Pixel-Level Transform을 실험하며 본 Competition dataset에 최적화된 Augmentation 조합을 찾고자 한다.

2) TTA

앙상블 기법인 Test time augmentation을 inference 단계에서 적용하여 예측결과의 편향을 억제한다. batch size, threshold를 바꾸거나 Augmentation을 적용하여 모델의 정확도를 높이고자 한다.

c. Model 설계/구축

Semantic Segmentation task에서 SOTA 성능을 보인 Network에 대해 논문을 리서치하여 모델의 Architecture와 동작 매커니즘을 정확히 파악한 후, 직접 설계하여 실험을 수행하였다. SMP(Segmentation_Models_PyTorch) 라이브러리와 torchvision 패키지 및 timm 패키지를 활용하였으며, pre-trained weight를 import하여 AI 모델을 구축하였다. 또한 scratch부터 직접 구현한 모델을 학습해 본 Competition에서 사용하는 dataset에 최적화된 network를 구축하는 등 다양한 방법으로 모델을 설계/구축하여 실험을 수행하고자 계획하였다.

d. Loss and Optimizer

1) Loss function

Loss function의 경우 본 Competition의 task인 Pixel-Level Multi-Label Classification에 맞게 Baseline Code로 제공된 BCEWithLogitsLoss(이하 BCELoss)와 더불어 추가적으로 DiceLoss 및 IoULoss를 구현해 학습에 사용하였다. 특히 단일 Loss로 학습하는 경우 이외에 여러개의 Loss를 Combine하여 Hybrid Loss를 구현해 학습을 시도해 보았으며, 그 과정에서 Hybrid Loss에 대한 하이퍼 파라미터인 weight를 변경하며 최적의 Combined Loss 조합을 찾기 위해 실험을 진행하였다.

2) Optimizer

DL 모델 학습 시 가장 좋은 성능을 보이는 Optimizer로 잘 알려져 있는 Adam을 포함해 AdamP를 사용하여 학습을 수행했다. 실험을 진행하며 Adam과 AdamP를 번갈아 사용하며 성능을 측정해 더 좋은 성능을 보이는 Optimizer를 선택하고자 하였다.

e. Ensemble

1) Group K-Fold

본 데이터 셋은 폴더 별로 한 인물의 양손 데이터가 주어졌는데, 이러한 두 손이 train/val 데이터 셋에 나눠서 분포할 경우 Data Leakage 현상이 우려된다. 따라서, 폴더(인물)를 group으로 둔 Group K-Fold로 데이터 셋을 분할하여 학습/검증이 진행되어 각 fold별 pt 파일이 저장되도록 구현하였다.

이후 K-Fold Inference에는 각각의 모델이 예측한 값을 Hard-Voting 방식으로 Ensemble하도록 구현하였다.

2) Inference Ensemble

다양한 모델들의 output.csv 대해 ensemble을 수행하여 Dice Score를 높이고자 하였다. 한 픽셀에 대해서 여러 클래스로 분류될 수 있는 만큼 모델별로 Pixel-wise Multi-Label Classification을 수행한 결과를 sum한 결과에 threshold값을 넘은 값들만 output으로 출력하는 Hard-Voting 방식의 Ensemble 코드를 작성하였다.

sum을 수행하기 위해서 inference 시 rle로 마스킹 된 output에 대해 mask map으로 다시 decode를 수행하였다. threshold 값은 (모델들의 수 * 0.7)를 default로 사용하였다.

f. Visualization

본 Competition에서 수행한 다양한 실험의 결과로 출력된 Output Prediction Mask를 시각화하였다. 추가적인 성능 보완과 Augmentation 조합 실험 및 모델 Architecture 개선 등을 수행하기 위해 Streamlit 라이브러리를 활용해 간단한 웹앱 플랫폼을 제작하였다. Visualization에 활용할 요소로는 Input Image와 Ground Truth Mask, Output Prediction Mask, Compare Loss Map을 선택하였으며, 다양한 실험을 통해 설계한 AI 모델의 예측 결과를 시각적으로 확인하고 추후 성능 개선의 지표로 활용하고자 하였다.

4. 수행 결과

a. Data EDA

Train dataset에 특징을 파악하기 위해 검수를 진행한 결과, 일반적인 왼손/오른손 Image를 제외하고 네일 혹은 매니큐어가 존재하는 경우, 반지를 착용한 경우, 수술을 통해 보형물이 존재하는 경우, 붕대를 착용한 경우 등 다양한 Noisy Image가 존재함을 확인하였다. 더불어 손목이 일직선으로 곧게 뻗어있는 일반적인 경우를 제외하고 안쪽으로 회전되어 촬영된 Noisy Image가 존재함을 확인하였다.

또한 Train dataset에 대해 OpenCV와 Numpy 라이브러리를 사용하여 Input Image의 전체 pixel value에 대해 channel 기준 Global Mean값과 Global Standard Deviation값을 구하였다. 계산 결과 아래와 같은 값이 도출되었으며, Input Image가 Gray Scale이기 때문에 3개의 channel(RGB)에 대한 Global Mean값과 Global Standard Deviation값이 동일하게 계산되었음을 알 수 있었다.

Global Mean = [0.12099939, 0.12099939, 0.12099939]

Global Std = [0.0363468, 0.0363468, 0.0363468]

b. Augmentation Ablation Study

1) Augmentation 조합

a) Resize

Input Image size가 2048 * 2048이기 때문에, 모델의 높은 정확도를 유지하되 학습 파라미터 수를 줄이기 위해 512 * 512 혹은 1024 * 1024로 Resize하고자 하였다. 다만 Input Image size가 High-Resolution일수록 detail한 boundary를 잘 포착하며 Low-Resolution일수록 semantic한 정보를 잘 포착할 수 있으므로, 512 * 512로 Resize하여 학습한 모델과 1024 * 1024로 Resize하여 학습한 모델을 Inference 단계에서 Ensemble하여 모델로 하여금 Robustness를 잘 확보하도록 계획하였다.

b) ColorJitter

ColorJitter 기법은 Brightness/Contrast/Saturation값을 변경하여 Input Image의 RGB-channel값을 변경하는 기법이다. 본 Competition에서 사용하는 data의 경우 Gray Scale이지만 ColorJitter를 통해 AI 모델의 Robustness를 확보할 수 있을 것이라 생각하여 dataset에 적용하였다. brightness=[0.8, 1.2], contrast=[0.8, 1.2], saturation=[0.8, 1.2], hue=[-0.2, 0.2]로 argument를 설정해 실험한 결과, ColorJitter를 적용하기 전보다 Avg Dice가 약 2%정도 향상함을 확인하였다.

c) Sharpen

입력 이미지를 기존보다 더 양각화하여 물체의 경계선을 두드러지게 증강해주는 기법을 뜻한다. Hand bone data는 class끼리 겹치는 부분을 Segmentation해야 하기 때문에 경계선을 강조해준다면 성능 향상이 될 것이라고 생각하였다. Albumentation에서 설정한 default parameter를 사용했을 때 성능이 향상됨을 확인하였다. 좀 더 경계를 두드러지게 하기 위해 Albumentation demo를 사용하여 parameter를 조정하여 실험해 보았으나 성능 향상은 보이지 않았다.

d) RandomBrightnessContrast

RandomBrightnessContrast는 밝기와 대비를 조정하여 Input Image를 변형시키는 증강 기법으로, brightness_limit과 contrast_limit등의 argument를 통해 밝기와 대비를 조정하는 방식이다. brightness_limit=[-0.2, 0.2], contrast_limit=[-0.2, 0.2]로 설정하여 실험한 결과 좋은 성능을 보였다. 특히 Sharpen 기법과 함께 사용했을 때 Avg Dice가 약 2%정도 향상함을 확인하였다.

e) GridDropout

GridDropout 기법은 Input Image에 대해 사전에 정의한 box size와 box 갯수를 기준으로 random하게 Erasing하는 기법으로, Semantic Segmentation에서 일반적으로 좋은 성능을 보이는 것으로 알려져 있다. ratio=0.25, random_offset=True, holes_number_x=4, holes_number_y=4로 설정하여 실험을 수행한 결과, Avg Dice가 약 1%정도 향상함을 확인하였다.

f) GridDistortion

GridDistortion은 그리드 왜곡을 적용해 data를 증강하는 기법이다. Albumentation demo를 사용해서 data에 적용해본 결과 다양한 input image에 robust한 모델을 만들수 있을 것으로 예상되어 GridDistorstion을 사용하였다. 확률값은 0.5를 사용했으며 큰 폭으로 성능이 하락하였다.

g) CropNonEmptyIfExists

물체가 있는 부분만 확대하여 배경을 지우는 효과가 있는 증강 기법이다. EDA결과 input 이미지들이 손이 그려진 위치가 각자 다 다른 상태이고, 특히 손목의 위치가 다양함을 확인하여 손목의 위치를 고정시켜주기 위해 사용하였다. 예상과 다르게 큰 성능향상은 없었다.

h) Emboss

Emboss는 이미지에 새로운 입체적 효과를 부여하여 이미지의 강조된 경계와 돋보이는 텍스처를 생성하는 증강 기법이다. Hand bone data는 class끼리 겹치는 부분을 Segmentation해야 하기 때문에 입체적 효과를 부여한다면 성능이 올라갈 것이라고 생각하였다. 예상과 다르게 약간의 성능 하락을 보였다.

i) Invert

Invert는 이미지의 색상값을 반전시키는 증강 기법이다. data가 grayscale이기 때문에 흰색은 검정색으로, 검정색은 흰색으로 색상 값이 반전된다. 손가락 끝마디 부분 class의 dice가 낮게 나와서 흰 바탕에 검은 픽셀로 바뀐다면 경계를 더 정확히 인지할 수 있을 것이라는 가설을 세우고 실험하였다. 실험 결과, 성능의 차이가 미미했다.

j) Unsharp Mask

Unsharp Mask는 이미지의 고주파 성분을 강조하고 낮은 주파수 성분을 억제하여 이미지의 선명도를 증가시키는 증강 기법이다. Hand bone data는 class끼리 겹치는 부분을 Segmentation해야 하기 때문에 경계선을 강조해준다면 성능 향상이 될 것이라고 생각하였다. 실험 결과, 약간의 성능 향상을 보였다.

k) GaussianNoise

GaussianNoise는 Noise를 추가해서 모델의 Generalization performance 향상시키고 overfitting을 방지하는 증강 기법이다. GaussianNoise를 사용하여 모델을 robust하게 만들기 위해서 사용하였다. 실험 결과, 성능의 차이가 미미했다.

l) ElasticTransform

ElasticTransform은 이미지의 픽셀을 변형하여 이미지 전체적으로 탄성이 있는 효과를 주는 증강 기법이다. 이 효과로 인해 data의 다양성을 확보하고 Generalization performance를 향상시킬 수 있을 것이라 생각하였다. 단독으로 사용하였을 때는 성능이 떨어졌지만 Horizontal Flip과 함께 사용했을때 약간의 성능 향상이 있었다.

2) TTA

a) 테스트 과정에서 batch size를 줄여 1장씩 예측값을 내는 것이 도움이 될 것 같아서 batch size를 모델을 학습시켰던 4가 아닌 1, 2, 8로 학습시켜 보았다. 또한, threshold를 증가시키면, 예측값이 좀더 엄격해져 더 정확하게 segmentation할 것이라고 예상하였다. 그러나 두 가지 경우 모두 큰 차이가 없었다.

c. **Model** 설계**/**구축

1) SMP

a) MANet

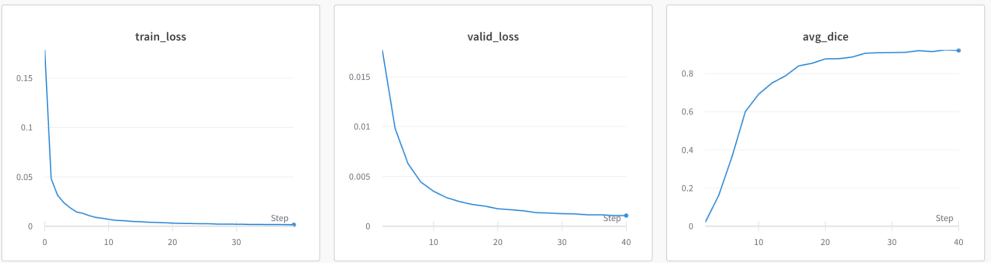
MANet(Multi-Scale Attention Net)은 attention mechanism에 기반한 PAB(Point-wise Attention Block)와 MFAB(Multi-scale Fusion Attention Block)를 사용하여 pixel 간 contextual

dependencies를 잘 학습하는 모델이다. smp 라이브러리 내 구현된 MAnet을 활용하여 실험을 진행하였다.

본 모델은 다른 모델들과 다르게 수렴이 매우 늦어 lr를 기존의 5배인 5e-4로 진행해야했다. 실험 결과, avg dice 0.95 이상의 준수한 성능을 보였다.

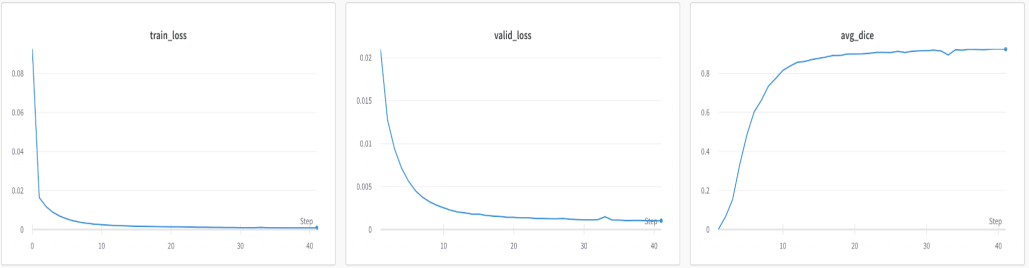
b) FPN

다양한 scale의 feature map을 활용하는 FPN(Feature Pyramid Net)을 활용하면 다양한 객체를 탐지할 수 있을 것이라 생각하여 사용하였다. SMP 라이브러리 내 구현된 FPN을 활용하여 실험을 진행하였다. Input Image size를 512 * 512로 Resize하여 별도의 Augmentation 없이 학습을 수행했으며, 40 epoch으로 학습한 결과는 아래의 그림과 같다. 전반적으로 Loss의 경우 우하향하며 Dice의 경우 우상향하는 등 준수한 성능을 보였다.



c) PSPNet

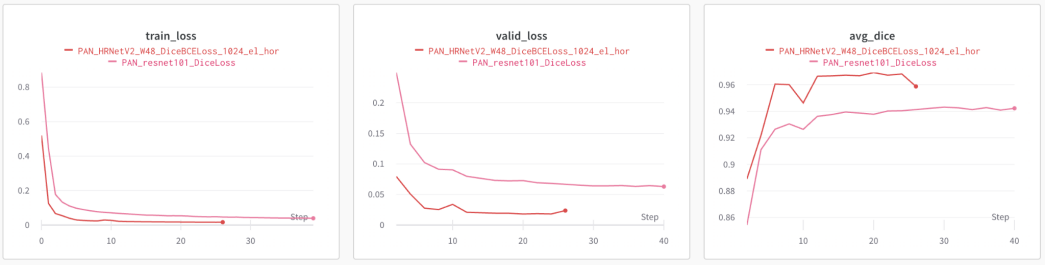
PSPNet은 4개의 level로 구성된 Pyramid Pooling Module을 사용해서 다양한 크기의 context 정보를 활용하는 모델이다. 겹쳐있는 class들을 효과적으로 masking 할 것이라고 생각하여 사용하였다. smp 라이브러리 내 구현된 PSPNet을 활용하여 실험을 진행하였다.



d) PAN

Pyramid Attention Network(PAN)는 global 문맥 정보를 파악하기 위해 기존 dilated convolution을 사용했던 모델들과 달리 attention mechanism과 spatial pyramid를 활용한

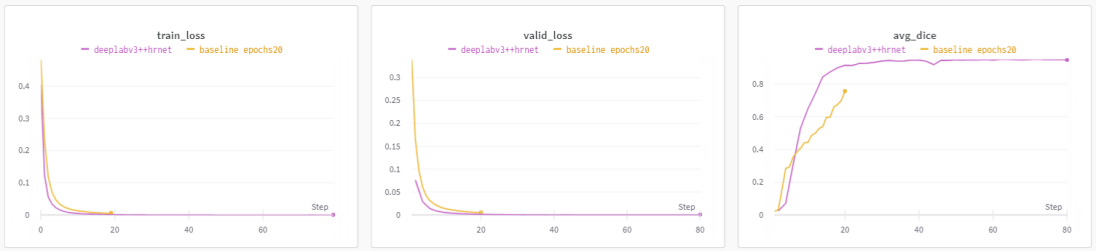
모델이다. smp 라이브러리 내 구현된 PAN을 활용하여 실험을 진행하였고, encoder(backbone)로 resnet101과 hrnet_w48(timm)을 사용하여 실험을 하였다.



High resolution 정보를 유지하는 hrnet을 backbone으로 사용했을때가 avg dice 0.96을 넘기며 resnet101을 사용했을 때보다 더 높은 성능을 보였다.

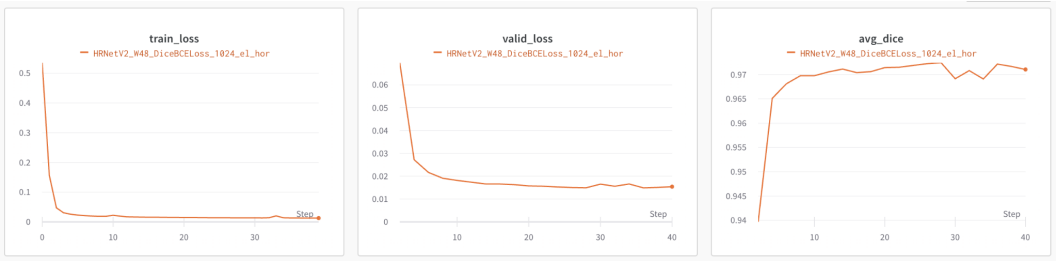
e) DeepLab v3+

Torchvision 모델과 달리 encoder 설정이 쉽기 때문에 smp패키지로 모델을 구현하였다. Atrous conv를 이용하여 input 이미지의 다양한 영역을 보는 모델이기 때문에 해상도 제어로 성능이 겹친 영역인 손목 부분 탐지에 도움이 될 것 같아서 학습을 진행하였다. 또한, Encoder로 HRNetV2_W48을 사용하여 화소가 보존되도록 구현하였다. 실험 결과 baseline으로 학습했을 때보다 월등한 성능 향상이 있었다.



f) HRNetV2_W48

HRNet은 multi resolution의 branch를 병렬적으로 진행시켜 high resolution 정보를 보존하고, 각 resolution이 가지는 feature map을 융합하여 보완하는 모델이다. timm 라이브러리 내에 hrnet_w48 모델을 기반으로, netron을 통해 시각화하여 그 구조를 확인하였다. 이후 classification head를 제거하고 segmentation head를 구현해 추가함으로써 강의에서 배운 HRNetV2_W48 구조로 수정하여 본 task에 적용 가능하게 하였다.



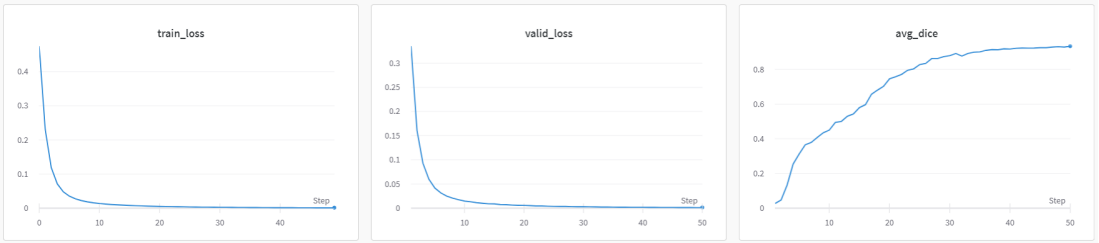
실험 결과, 본 모델에 1024 resize, elastic deformation, horizontal flip을 적용한 것이 submission dice score 0.9707로 단일 모델 중에는 가장 높은 성능을 보였다.

2) torchvision

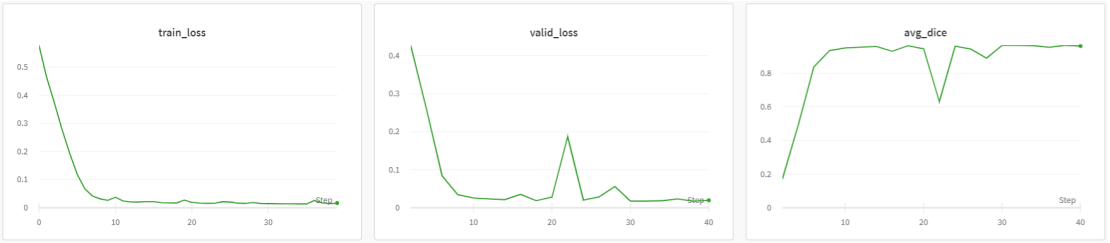
a) FCN

Semantic Segmentation task의 시초 network라고 할 수 있는 FCN 모델을 pre-trained weight를 import하여 사용하고자 하였다. 특히 Backbone network로 ResNet을 사용하였는데, ResNet50과 ResNet101 2가지 version을 구성하여 2개의 FCN을 구축해 실험을 수행하였다.

FCN_ResNet50 모델의 경우 Input Image size를 512 * 512로 Resize하여 별도의 Augmentation 없이 학습을 수행했으며, 50 epoch으로 학습한 결과는 아래의 그림과 같다. 전반적으로 Loss의 경우 우하향하며 Dice의 경우 우상향하는 등 준수한 성능을 보였다.

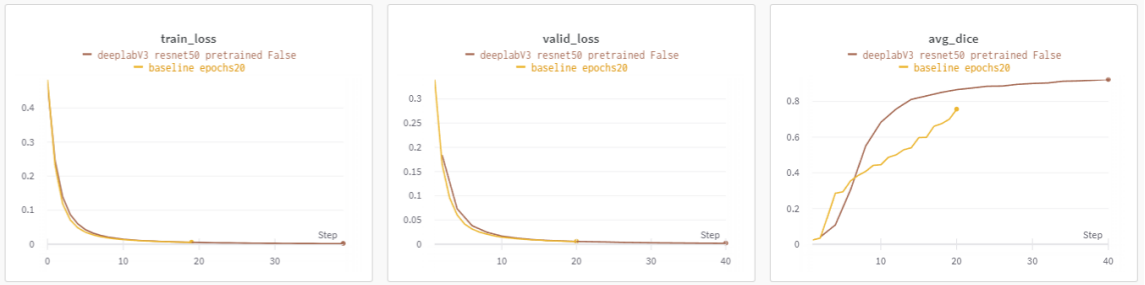


FCN_ResNet101 모델의 경우 Input Image size를 1024 * 1024로 Resize하여 ColorJitter/Sharpen/RandomBrightnessContrast/Normalize Augmentation과 함께 학습을 수행했으며, 40 epoch으로 학습한 결과는 아래의 그림과 같다. 전반적으로 Loss의 경우 우하향하며 Dice의 경우 우상향하는 등 단일 모델 기준 96.53%의 Avg Dice로 준수한 성능을 보였다.



b) DeepLab v3

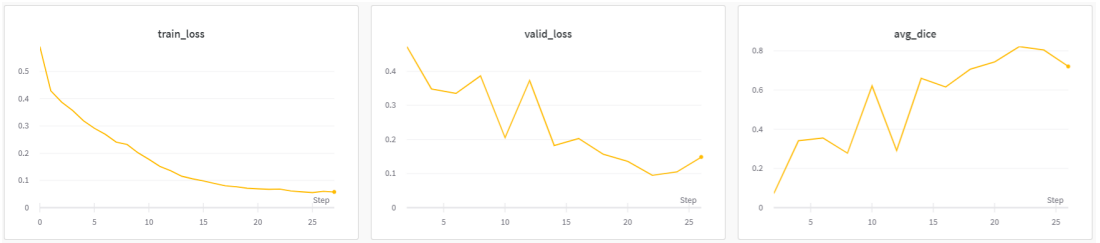
dilated conv를 이용하여 receptive field를 확장하여 해상도가 줄어드는 현상을 방지한 모델이다. 다른 모델을 사용했을 때, 학습 결과 손목 부분의 겹쳐있는 영역이 학습이 잘 이뤄지지 않고 있음을 확인하였고, 해상도가 줄어들지 않으면 이러한 부분이 해결될 것으로 예상하여 학습을 진행했다. FCN모델보다 성능이 월등이 좋음을 확인하였다.



3) scratch version

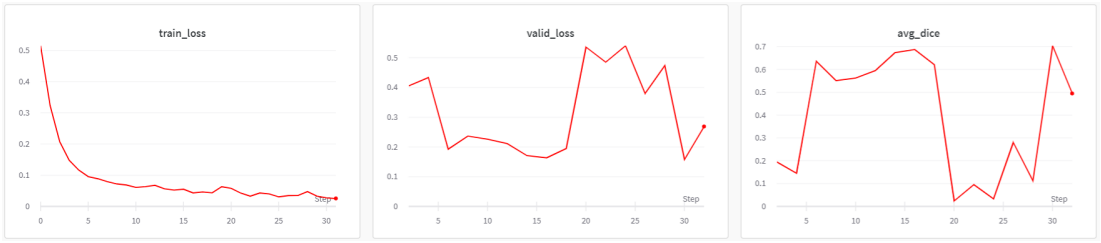
a) U-Net

Encoder-Decoder의 Architecture를 가지며 Encoder Stage에서의 Output feature map을 Concatenation 방식의 Skip-Connection을 통해 그에 대응되는 Decoder Stage로 추가하여 Input feature map으로 사용함으로써 Semantic한 정보와 Spatial한 정보를 잘 전달하는 U-Net을 scratch로 구현하여 학습을 진행했다. 특히 모델을 scratch로 구현하는 과정에서 Activation function으로 SELU 함수를 채택해 모델의 정확도 향상을 꾀하였다. Input Image size를 1024 * 1024로 Resize 하였으며 ColorJitter/Sharpen/RandomBrightnessContrast/GridDropout/HorizontalFlip Augmentation을 적용하여 40 epoch으로 학습한 결과는 아래의 그림과 같다.



b) HRNet v2

전반적인 Architecture에서 feature map의 해상도를 High-Resolution으로 유지하며 object boundary의 detail한 정보 및 pixel간 spatial 정보를 잘 보존함으로써 SOTA 성능을 달성한 HRNet v2를 scratch로 구현하여 학습을 수행하였다. 특히 모델을 scratch부터 구현하는 과정에서 Stage Fusion 단계에 존재하는 Low-Resolution으로의 feature map 생성의 경우 그 과정을 간소화했으며, Activation function으로 SELU 함수를 채택해 모델의 정확도 향상을 꾀하였다. Input Image size를 1024 * 1024로 Resize 하였으며 ColorJitter/Sharpen/RandomBrightnessContrast/GridDropout Augmentation을 적용하여 40 epoch으로 학습한 결과는 아래의 그림과 같다.



c) Problems in Scratch version Models

scratch로 구현한 U-Net과 HRNet v2 모두 학습 과정에서 ① Bus Error가 발생해 학습을 끝까지 완료하지 못한 문제, ② 학습 과정에서 Valid Loss와 Avg Dice가 Oscillate하는 문제가 발생했다. ①의 근거로 Input Image의 Resolution이 1024 * 1024로 비교적 컸으며 batch_size 및 num_workers 하이퍼 파라미터를 충분히 작게 조정하지 못한 점으로 추측된다. ②의 근거로 Bus Error를 피하기 위해 batch_size를 작게 조정하는 과정에서 다양한 Augmentation을 수행할 경우 1 epoch당 AI 모델이 관찰하는 data의 개수가 적기 때문에 Data Distribution이 일관되지 않아 Generalization 성능에 문제가 발생한 것으로 추측된다.

d. Loss and Optimizer

1) Dice Loss

Dice Loss란 Segmentation task의 평가 metric으로 사용되는 Dice Score를 targeting하기 위해 고안된 Loss function으로, 민감도(Sensitivity)와 정밀도(Precision)의 조화 평균으로 구성되는 Dice Score를 1에서 뺀 값으로 정의한다. Dice Loss에 대한 정의는 아래의 식과 같다.

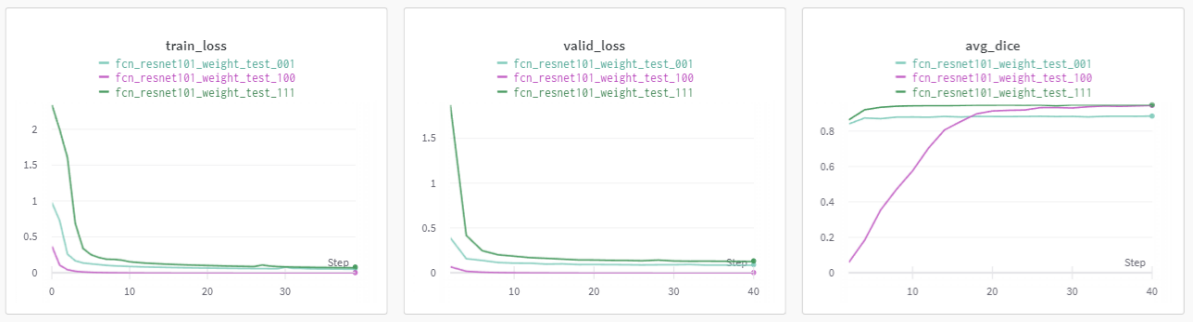
$$Dice\ Loss = 1 - Mean(\frac{2\ X\ TP}{(TP + FP) + (TP + FN)})$$

Dice Loss만을 사용해(단일 Loss로 Dice Loss 채택) 학습을 수행한 결과, 본 Competition의 평가 metric으로 사용되는 Avg Dice에 대해 좋은 성능을 보였으며 Train Loss 및 Valid Loss 모두 빠르게 수렴함을 확인하였다.

2) IoU Loss

Segmentation task의 대표적인 metric 중 하나인 IoU(Intersection over Union)는 Prediction Mask 영역과 Ground Truth Mask 영역 사이의 합집합에 대한 교집합의 비율값이다. 이를 targeting한 IoU Loss는 1에서 각 Class 별 IoU값의 전체 평균을 뺀 값으로 정의되며, Segmentator 학습 시 IoU Loss를 기반으로 weight를 update한다면 좋은 성능을 보일 것이라 기대했다.

IoU Loss만을 사용해(단일 Loss로 IoU Loss 채택) 학습을 수행한 결과, 성능 개선이 미미한 것을 확인하였고, IoU Loss만을 사용하는 것은 부적절할 것이라고 예상하였다.



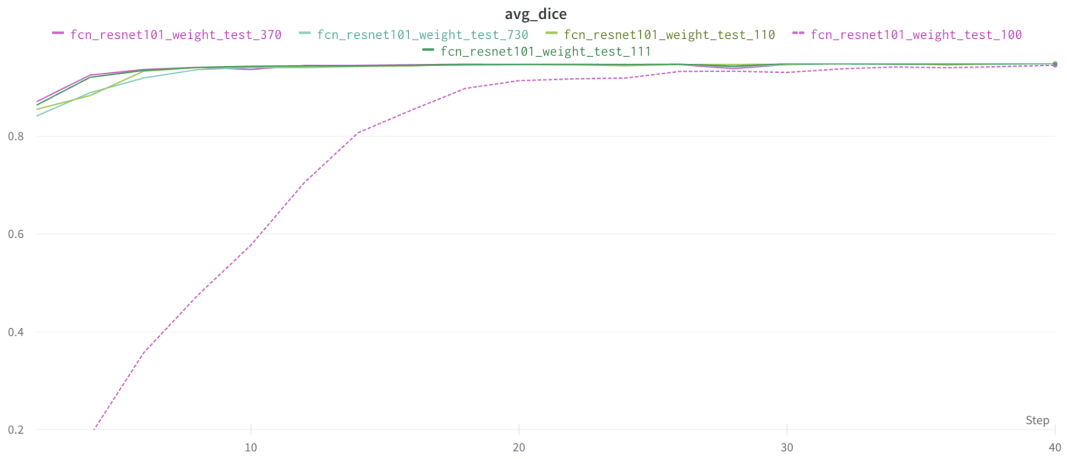
3) Hybrid Loss

Segmentation task에서 BCE Loss도 충분한 성능을 보여주었지만 평가 Metric이 Dice Score인 만큼 Dice Score 과 IoU Loss를 추가하면 성능 개선이 이루어질 것이라 판단하였고, 아래와 같은 Hybrid Loss를 만들었다.

$$HybridLoss = (\alpha * BCELoss) + (\beta * DiceLoss) + (\gamma * IoULoss)$$

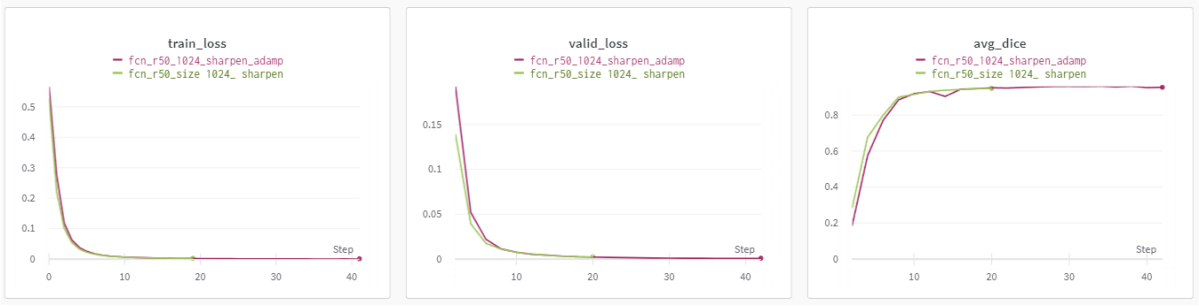
수식의 하이퍼 파라미터 값을 다르게 조율하면서 Hybrid Loss에 대한 실험을 진행해보았다. BCE Loss, Dice Loss, IoU Loss를 다양한 비율로(1:0:0, 1:1:1, 1:1:0, 7:3:0, 3:7:0) 적용시켜본 결과 비율에

상관 없이 BCE Loss와 Dice Loss를 같이 사용하기만 하면 성능과 수렴 속도에서 개선되는 것을 확인할 수 있었다.



4) AdamP

학습과정중 normalization을 적용하면 weight norm이 과도하게 빠르게 증가하여 성능에 좋지않다. 따라서, momentum에 의해 발생하는 weight 방향과 평행인 gradient 성분을 projection을 통해 과도한 weight norm 증가를 막는다. 이를 이용하여 Adam을 이용했을 때보다 0.01정도의 성능 향상을 확인했다.



e. Ensemble

1) Group K-Fold

PAN_resnet101 모델에 대하여 k-fold 학습을 수행하여, 그 효과를 확인해 보았다. 결과적으로 적은 epoch수에도 불구하고 kfold를 수행하여 ensemble 한 결과 값이 단일 fold로 학습한 결과보다 더 높은 성능을 보였다.

Model	epochs	kfold	Submission Dice
PAN_resnet101	40	X	0.9296
PAN_resnet101	20	O	**0.9351**

2) Inference Ensemble

앙상블을 수행한 결과 단일모델보다 더 높은 성능을 보였다.

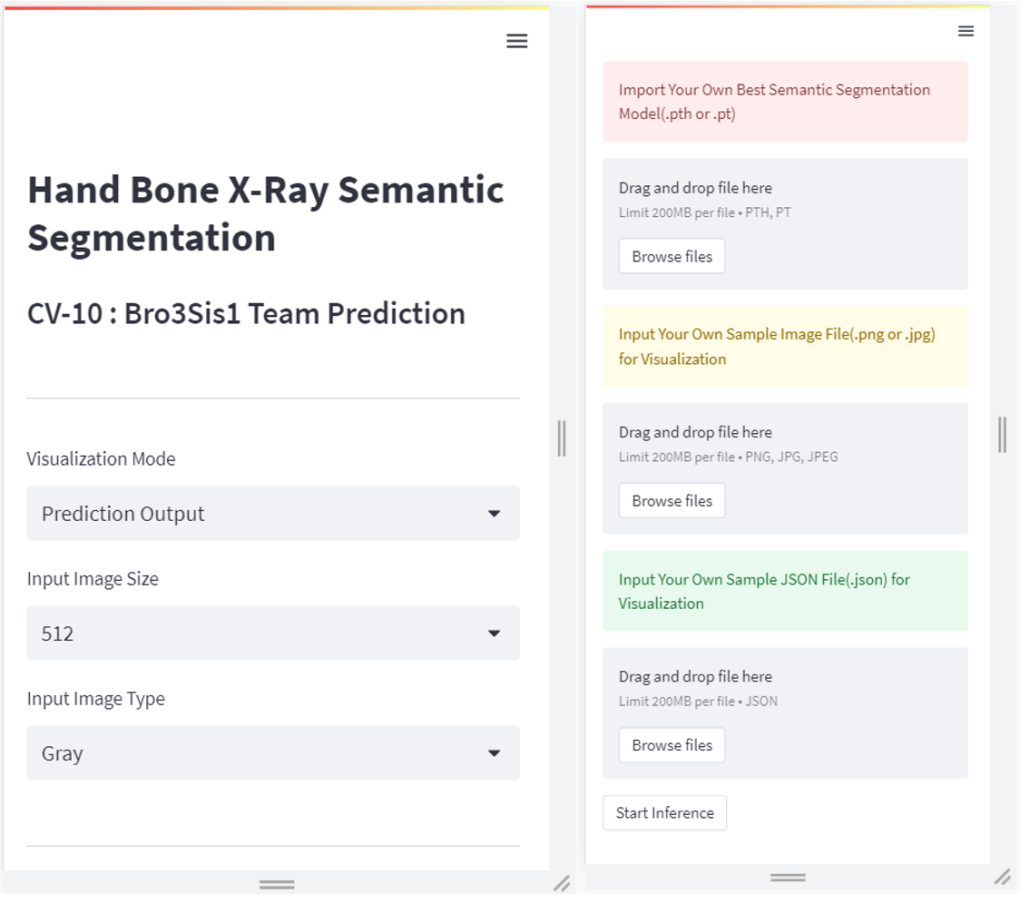
Model	Dice Score
FCN_ResNet101	0.9605
HRNet_DiceBCE	0.9707
DeepLab+_HRNet	0.9699
MANet	0.9510
Ensemble	**0.9721**

추가적으로 Metric이 되는 Dice Score의 분모에 오탐(FP)과 미탐(FN)이 포함되어 있어 적절한 threshold를 찾고자 하였다. Threshold 값에 대해서도 실험을 해본 결과 (모델들의 수 * 0.7) 이 가장 높은 점수를 보였다.

Model	Treshold	Dice Score
MANet + Fcn_ResNet50 + FCN_ResNet101	int(모델 수 * 0.7) = 2	**0.9610**
	1	0.9510
HRNetv2_w48 + DeepLabv3plus_hrnet + PAN_hrnet + FCN_ResNet101 + UNet++_HRNet, MANet	int(모델 수 * 0.7) = 4	**0.9700**
	5	0.9647

f. Visualization

Streamlit 라이브러리를 활용해 간단한 웹앱 플랫폼을 제작하는 과정에서 inference.py의 코드 중 Output Prediction Mask를 출력하는 부분을 적절히 활용해 app.py를 구축하였다. 더불어 app.py에 사용하게 될 Output Prediction Mask와 Ground Truth Mask의 Pixel-wise Difference Value를 시각화하는 함수와 Loss Value based feature map을 시각화하는 함수를 app_utils.py에 별도로 구현하였다. 아래의 그림은 Streamlit 라이브러리를 활용해 최종적으로 구축한 AI 모델 결과 시각화용 웹앱 플랫폼 화면의 모습이다.



Visualization Mode(Prediction Output, Compare Loss)와 더불어 Input Image Size(512, 1024)와 Input Image Type(Gray, RGB)을 선택할 수 있으며, 학습 후 가장 성능이 좋은 Best Model과 시각화를 수행할 Sample Data(Image-Label pair)를 업로드할 수 있게 구현했다. 이를 통해 본 Competition을 위해 설계한 모델의 Output을 시각화함으로써 예측 결과를 파악할 수 있었고, 낮은 정확도로 예측한 경우를 발견해 성능 향상에 도움이 될 수 있었다.

5. 자체 평가 의견

a. 잘한 점

- github을 활용한 협업을 하도록 노력하였다.
- 다양한 tool을 적용하려고 노력했다.
- streamlit을 사용해서 inference 결과를 시각화하여 monitering 하였다.
- optuna 사용법을 익히고 우리 코드에 적용시켰다.
- error 상황을 공유하여 문제를 해결위해 노력했다.
- Augmentation Ablation Study를 체계적으로 수행했다.
- smp, torchvision, scratch 등 다양한 방식을 통한 모델 실험을 진행하였다.
- 구성원 모두가 오프라인 미팅에 성실히 참여해 활발히 협업했다.

b. 시도했으나 잘 되지 않았던 것

- optuna를 구현하여 실험하는 과정에서 wandb와의 연동성이 문제가 있었다.
- scratch로 model을 구현했지만 학습이 제대로 이루어지지 않았다.
- 학습시간을 줄이려고 다양한 노력(FP16 등)을 했지만 효과가 없었다.
- smp 내의 model(U-net, Linknet)의 학습이 잘 이루어지지 않았고 해결하지 못했다.

c. 아쉬웠던 것

- 시간이 부족하여 K-Fold 실험을 1024 크기에 대하여 제대로 진행하지 못했다.
- 다양한 Backbone network에 대해서 실험해보지 못했다.
- Augmentation Ablation Study를 진행했으나 이번 프로젝트의 task에 적합한 조합을 찾지 못했다.
- 고질적인 문제였던 Bus Error에 대해 근본적인 해결책이 찾지 못했던 점이 아쉬웠다.
- Input Image Size를 512와 1024가 아닌 더 다양한 Size로 실험했다면, batch size 조정을 통해 AI 모델이 더욱 일반화된 Data Distribution을 학습해 더 좋은 성능을 보였을 것이라 기대한다.
- data cleaning을 진행해보지 못했다.
- Jaccard loss를 사용하여 loss의 다양화를 시도해보지 못했다.
- TTAch패키지를 이용해보지 못했다.

d. 프로젝트를 통해 배운 점 또는 느낀 점

- Semantic Segmentation task에 사용되는 다양한 SOTA 모델의 동작 원리와 그 구조를 알 수 있었고, 단순히 라이브러리를 사용해 pre-trained 모델을 Import하는 방법과 scratch부터 구현하는 방법 모두를 사용해 모델을 구축하는 방법에 대해 깊이 배울 수 있었다.
- Semantic Segmentation Competition의 제출용 Prediction Output format으로 RLE가 사용된다는 것을 알게 되었고, Inference 단계에서의 Model Ensemble 수행 시 RLE format을 기준으로 Encoding/Decoding하여 threshold를 사용한 Hard-Voting을 자세히 이해하고 직접 구현할 수 있었다.
- Streamlit 라이브러리에 존재하는 다양한 API를 최대한으로 활용해 AI 모델의 예측 결과를 시각적으로 파악하기 위한 Web Platform을 제작한 점에 대해 뿌듯함을 느꼈다.

Lv.2 Competition Personal Wrap-up Report

김보경**_T5033(CV-10** 형셋누나하나**)**

Project Direction Setting

- Baseline 리팩토링 및 wandb 연동 코드 작성
- DataEDA를 통한 Data 특성 파악
- torchvision 기반의 FCN, LR-ASPP 모델 및 SMP기반의 FPN 모델 실험
- hybrid loss의 weight 실험 구상 및 진행
- 모델 별 output.csv들을 앙상블하는 ensemble.py 코드 기획 및 작성

- 하이퍼 파라미터 튜닝을 위한 train_optuna.py 기획 및 작성

Trials for Myself

- 단일 loss만을 사용한 실험들을 대조군으로 여러 loss를 결합한 hybrid loss를 실험군으로 두었으며, hybrid loss에 각 가중치를 다르게 주어 실험하였다.
- 모델 별 output.csv를 mask map으로 decoding 한 후, hard voting하여 ensemble을 진행하였다.
- 프로젝트를 진행하면서 논의한 내용, 실험 수행한 결과 등을 공유하기 위한 노션 페이지 구축 후 지속적으로 관리 및 사용 독려하였다.
- 주어진 baseline 코드를 스크립트 형식의 py파일로 리팩토링하여 팀원들에게 공유하였으며, 팀원들의 피드백을 통한 지속적인 코드 업데이트를 진행하였다.
- train.py에 하이퍼파라미터 튜닝을 위한 optuna를 연동시킨 train_optuna.py를 작성하고 실험하였다.
- 시간 절약을 위한 학습 종료를 slack으로 알려주는 alarm.py를 작성 및 공유하였다.

Feedback

- wandb 와 github 사용을 독려한 덕분에 팀원들이 실험한 내용을 적극적으로 공유할 수 있었다.
- 팀원들과 논의를 통해 기획했던 다양한 실험들을 체계적으로 실험할 수 있어서 좋았다.
- train_optuna.py를 구현하였으나 시간 부족으로 실제 hyper parameter tuning에 활용하지 못하였고, wandb와의 연동을 바르게 하지 못했다.
- 최신 라이브러리(mmsegmentation)을 활용해보지 못해 아쉬웠다.
- 라이브러리 내 코드와 docs를 보면서 실험들을 기획하였지만 추후 우리가 알지 못한 backbone의 존재를 알게되면서 라이브러리를 충분히 분석하지 못했다고 느꼈다.
- Data EDA를 통해 insight를 얻어 실험에 적용하는 부분이 충분하지 못했다고 느꼈다.
- 대회 성능 개선 외에도 서비스 코드 등 추가적인 목표를 기획하고 실험해보았으면 좋았을 것같다는 아쉬움을 느꼈다.

Lv.2 Competition Personal Wrap-up Report

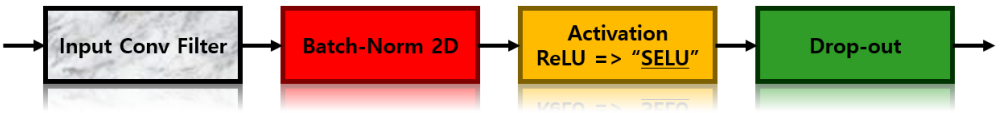
김정주**_T5052(CV-10** 형셋누나하나**)**

Project Direction Setting

- **Data EDA**를 통해 **Dataset**의 특성 파악 및 Pixel-wise Normalization을 위한 **Mean/Std** 계산
- AI 모델의 Generalization 성능 향상을 위한 다양한 **Augmentation** 조합 검색 및 실험 수행
- torchvision 라이브러리 기반/scratch 구현 기반 **Segmantator** 설계**/**구축 및 **Architecture** 고도화
- BCE Loss 이외에 Avg Dice Score를 향상시킬 수 있는 다양한 **Loss function** 구현 및 실험
- Streamlit 라이브러리 기반 **AI** 모델 예측 결과 **Visualization**용 **Web Platform** 제작

Trials

- Data EDA를 통해 일반적인 왼손/오른손 Image 이외에 **Noisy**한 **Image**들의 존재를 파악함. 특히 Augmentation으로 Normalize 기법을 활용하기 위해 **argument**로 인가되는 **channel-dim** 기준 **Mean Value** 및 **Std Value**를 계산함
- **512 or 1024 size**로의 **Resize** 기법을 포함해 **ColorJitter, Sharpen, RandomBrightnessContrast, GridDropout, ElasticTransform, HorizontalFlip, Normalize**를 수행하여 그 성능을 파악함
- **torchvision** 라이브러리 기반의 **pre-trained weight**를 Import한 **FCN_ResNet50**과 **FCN_ResNet101**을 설계함. 또한 **U-Net**과 **HRNet v2**를 **scratch**로 구현하여 학습을 수행함. 특히 그 과정에서 **Basic Conv Block**을 **Advanced**하게 고도화하여 학습 성능 향상을 위해 노력함
  ① **Batch-Normalization**을 추가하여 **Over-fitting** 방지
  ② **Activation function**을 **ReLU**에서 **SELU**로 대체하여 정확도 향상
  ③ **AI** 모델 하단부에 **Drop-out**을 추가하여 **Generalization** 성능 향상



- Baseline으로 제공된 기본 BCE Loss 이외에 **Dice Loss** 및 **IoU Loss**를 직접 설계하여 **Weighted-Sum & Combined** 형식의 **Hybrid Loss**를 구축해 학습에 활용함

$$HybridLoss = (\alpha * BCELoss) + (\beta * DiceLoss) + (\gamma * IoULoss)$$

- **Streamlit** 라이브러리를 활용해 AI 모델의 **Output Prediction Mask**와 **Ground Truth Mask**를 비교하는 함수를 구현하고 **app.py**를 구현함. 특히 **Pixel-wise**한 **Different Value**와 **Compared Loss Feature Map**을 plot하여 AI 모델 예측 결과를 시각적으로 파악하고 추후 성능 개선에 활용함
- 프로젝트를 진행하며 팀원들 각자가 수행한 내용과 그 이유 및 장단점에 대해 **log**를 남길 수 있는 별도의 **Notion page**를 구축해 효율적인 협업을 위해 노력함
  https://flower-bobolink-12a.notion.site/Hand-Bone-X-Ray-Semantic-Segmentation-3d7ded851204451eb97dc712a30d5151

FeedBack

- *scratch*로 구현한 *U-Net*과 *HRNet v2*가 학습 과정에서 *Bus Error*가 발생한 문제와 *Loss & Dice*가 *Oscillate*한 문제를 해결하지 못했던 점이 아쉬움. *Input Image*를 *512*와 *1024* 사이의 크기로 *Resize*를 수행해 학습을 수행했다면 문제를 효과적으로 해결할 수 있었을 것이라 기대함
- 팀원들과 함께 협업하여 최적의 *Augmentation* 조합을 찾고, *Compound Loss*에 대해 최적의 *weight*값을 찾아내 학습을 수행한 것에 대해 뿌듯함을 느낌
- *Github*와 *wandb*를 최대한 활용하여 팀원들과 함께 실험 결과에 대해 효율적으로 인사이트를 공유하고, 성능 개선을 위해 노력했던 점이 좋았음

Lv.2 Competition Personal Wrap-up Report

양재민_**T5123(CV-10** 형셋누나하나**)**

Project Direction Setting

- Segmentation에 대한 이해가 부족했기 때문에 이번 프로젝트를 통해서 프로젝트의 흐름을 잘 따라가며 협업에 익숙해지는 것과 강의수강과 추가 학습을 통해서 segmentation의 개념을 이해하고 실험을 통해 이해한 내용을 적용 및 활용하는 것을 목표로 했습니다.

Trials for Myself

- 위의 목표를 달성하기 위해서 가장 중요한 것은 강의 수강이라고 생각해서 강의를 매우 꼼꼼하게 듣고 추가 학습을 한 후 개인 노션에 해당 내용을 정리하며 복습했습니다. 실험을 할 때는 기계적으로 하는 것이 아니라 알고 있는 이론이 어떤식으로 반영되는지 이해하려고 했습니다. 원활한 협업을 위해서는 팀원들이 어떤 작업을 하고 있는지, 내가 현재 해야할 작업은 무엇인지를 확실히 인지하고 맡은 작업을 확실히 수행하려고 했습니다. 또한 스스로 해결할 수 없다고 판단되면 팀원들에게 조언 및 도움을 구해서 해결했습니다.

Problem-solving Process and Results

- 팀에서 Augmentation Ablation Study를 맡았습니다. 해당 task에 맞는 Augmentation을 적용해보고자 Albumentation의 여러 Augmentation을 data에 적용시켜 눈으로 확인하고 가설을 세워 실험에 적용하였습니다.
- Scratch로 Model을 구현하면서 해당 layer에서 이뤄지는 과정을 생각하며 구현했고 Advanced한 activation function을 적용하여 성능을 높이고자 하였습니다.
- smp 라이브러리를 사용하여 pretrained된 모델을 사용해보고 모델에 대해서 공부했습니다. 또한 라이브러리에서 제공되는 다양한 encoder들을 적용하여 task에 최적인 모델을 찾고자 하였습니다.

Limitations

- Augmentation Ablation Study를 하면서 많은 실험을 했지만 최적의 조합을 찾지 못한 것이 아쉽습니다. 대부분 성능이 떨어지거나 유지되었고 크게 향상된 것은 없어서 다른 여러 자료들을 보며 공부할 예정입니다.
- Scratch로 구현하면서 팀원의 도움을 많이 받았습니다. 혼자서 모델의 구조와 핵심을 파악하고 구현할 수 있도록 공부할 예정입니다.

Future Direction

- 이번 프로젝트에서 많은 실험을 했지만 팀에 도움이 되는 결과를 얻지 못했습니다. 많은 논문을 찾아보고 적용하면서 팀에 도움이 되겠습니다. 개인적인 일로 집중이 흐트러지기도 해서 아쉽습니다. 최종 프로젝트에서는 할 수 있는 일을 찾아서 스스로 할 수 있는 사람이 되고 더 노력하는 모습을 보이겠습니다.

Lv.2 Competition Personal Wrap-up Report

임준표_**T5175(CV-10** 형셋누나하나**)**

학습 목표

- 가설을 세우고 가설을 검증하는 방식으로 체계적으로 실험을 진행해본다.
- 강의 내용에 나온 최신 기술을 적용해 본다.
- 멘토링 때 접한 여러 tool 에 대해서도 사용한다.

나의 역할

- model 선정
- augmentation 조합 찾기
- fp16을 이용한 학습 시간 가속화

모델 개선 방법

- baseline으로 설정한 FCN을 기준으로 한가지씩 변화를 주어 더 좋은 조건을 찾는다.
- debugging을 이용하여 오류를 분석한다.
- onnx를 이용하여 모델 전체 구조를 확인 후 모델을 구현한다.

프로젝트 회고

- 노력한점
    1. 데이터 증강 기법 중, sharpen, cropNonEmptyMaskIfExists, resize 등을 구현하였다.
    2. torchvision과 smp를 이용하여 구현하였다.
    3. 모델 architecture외에도 adamp, inference에서의 조건 변화, fp16사용을 통해 더 좋은 결과를 도출해내고자 하였다다.

- 아쉬운점
    1. *Data cleaning*을 진행해보지 않았다.
    2. *Loss*를 *diceloss*와 *BCE loss*만 써봤는데 *Jaccard loss*등 다양한 *loss*를 사용해보지 않았다.
    3. 학습 후 결과를 이용하여 *EDA*를 진행해보지 못했다.
    4. *TTAch*를 이용하여 *TTA*를 진행하지 않았다.

[총평**]**

대조군인 BaseLine을 FCN으로 설정해놓고 조건을 한 개씩 바꾸며 성능 향상을 wandb로 확인하였다. 최신 기술인 AdamP, UNet++,HRNet 등을 적용해보았고, 그 과정 중에 배운 점도 많이 있는것 같다. 아쉬운 점이 있다면, git을 통한 공유를 제때제때 하지 않아서 팀원들에게 best model을 공유하는 것이 늦어진 점과 TTAch, optuna 등을 사용해보지 못한점과 dataload부분 시간 지연 문제를 해결해보지 못함이 있다. 다음에 기회가 된다면, 이들을 사용하여 성능 향상을 경험하고 싶다.

Lv.2 Competition Personal Wrap-up Report

정남교_**T5188(CV-10** 형셋누나하나**)**

Project Direction Setting

- 우리 팀은 base model 실험에 있어서 mmseg 라이브러리를 사용하면 그에 대한 이해와 커스텀하는데에 시간이 많이 소요될 것이라 생각하여 smp, torchvision 라이브러리 내의 모델을 활용하기로 하였다. 이번 프로젝트는 이전 프로젝트들의 아쉬웠던 점들을 최대한 반영하여 진행되도록 방향성을 잡아 진행하였다.

Trials for Myself

- kfold validation 기법을 활용하기 위해 train_kfold.py 와 inference_kfold.py를 구현하였다.
- ensemble.py를 구현하여 rle로 encoding 된 결과 파일들을 decoding 한 뒤에, hard voting 하여 ensemble을 진행하였다.
- smp 모델 중 PAN, MAnet의 실험을 진행하였다.
- bce, dice, hybrid loss를 구현하여 서로 다른 loss에 따른 실험을 진행하였다.
- timm hrnet.py를 segmentation task에 맞게 구조를 수정하여 사용하였다.

노력한 점

- 구현되어있는 라이브러리의 모델을 필요에 맞게 수정하여 사용할 수 있도록 노력하였다.
- netron을 활용한 모델 시각화를 통해 모델의 구조를 확인하고 구현에 및 수정이 수월하도록 활용하였다.
- 개별화된 역할 분담보다는 회의, 공유 등을 통해 전반적인 진행 과정을 함께하여 모든 팀원이 수행 내용 전체에 대해 이해할 수 있었다.
- 각자의 수정사항으로 인해 코드가 개별화가 되지않도록 깃을 활용하여 통합적인 하나의 코드 관리를 하도록 노력하였다.
- 주어진 baseline 코드를 그대로 활용하기 보다는 스크립트 형식의 py 파일로 우리의 필요에 맞게 커스텀한 코드를 구축하여 사용하였다.

아쉬운 점

- 512에 비해 1024로 resize 하는 것이 성능이 더 좋았지만, 학습 시간이 길어 kfold 까지 수행하지 못하였다.
- 실험을 모델을 분업하여 담당한 것은 좋았으나, 이후 loss, augmentation, optimizer 등 변경할 내용에 대해서 좀 더 계획적인 실험을 진행했으면 더 효율적인 실험이 되었을 것 같은 아쉬움이 있다.

- 다른 팀의 발표를 봤을때 DenseNet을 backbone으로 한 모델도 hrnet만큼 성능이 잘 나오는 것 같던데, 실험 해보지 못한 것이 아쉽다.
- optuna를 우리 코드에 맞게 구현하는것은 성공했으나, 시간 부족으로 실제 hyperparameter tuning에 활용하지 못한 것이 아쉽다.