Pseudo Labeling POC

DEEP NOID

Self-training: The simplest and effective way to leverage unlabeled data for medical image segmentation

AI 연구소 인턴 김주성

2023.10.4 ~ 2024.1.26

목차

DEEP NOID

Self-training: The simplest and effective way to leverage unlabeled data for medical image segmentation

Abstract

최근 딥러닝 기술의 빠른 발전으로 딥러닝 모델 학습에 필요한 데이터의 중요성이 계속해서 증가하고 있다. 그러나 의료 분야의 경우 데이터 레이블링(labeling)에 전문가 수준의 지식이 필요하다는 어려움이 있어 대량의 데이터를 수집하는데 한계가 있다. 본 연구에서는 의료 분야에서의 딥러닝 적용에 앞선 한계가 있음을 인지하고 이를 극복하기위한 방안으로 정답이 없는 대량의 데이터를 활용하는 방법에 대해 탐구한다. 그 첫 번째 단계로 Self-training을 흉부 X선 영상 속 병변 분할(disease segmentation) 문제에 적용함으로써 실험을 통해 그 가능성을 검증한다. Self-training은 정답이 없는 데이터에 대해 가짜 레이블(pseudo label)을 만들어 학습에 활용하고 이 과정을 반복함으로써 점진적으로 성능을 향상시키는 준지도학습(semi-supervised learning) 방법이다. 실험은 총 두 개의 태스크로 검증된다. 하나는 기흉(pneumothorax)만을 탐지(이하 1-finding)하는 문제이고 나머지 하나는 기흉, 폐경화(consolidation), 섬유화(fibrosis), 흉막 삼출(pleural effusion), 폐결절(nodule) 총 다섯 종류의 병변을 탐지(이하 5-finding)하는 문제이다. 이번 실험을 위해 총 4개의 병원에서 전문가에 의해 레이블링 된 소량의 데이터와 인터넷 상에 공개된 대량의 데이터를 사용했다. 실험의 평가를 위해 Dice기반 AUROC(이하 D-AUC)와 Dice 성능 지표가 사용됐고 Self-training 과정이 반복됨에 따라 성능의 증감 추이를 관찰했다. 실험 결과 1-finding의 경우 Self-training이 반복됨에 따라 D-AUC와 Dice가 점진적으로 향상됨을 확인했다. 그러나 5-finding의 경우 초기 가짜 레이블 생성 과정에서 확증 편향(confirmation bias) 문제가 발생하며 이로 인해 성능 향상에 한계가 있음을 보았다. 이를 통해 Self-training 적용 시 초기 단계에서 생성되는 가짜 레이블의 품질이 매우 중요한 요소임을 확인했다. 더 나아가 이러한 Self-training의 문제를 개선하기 위한 방법으로 초기 단계에서부터 정답이 없는 데이터의 표현(representation)을 학습할 수 있는 Consistency 기반의 방법을 함께 적용하는 것이 효과적일 수 있음을 주장한다. 이번 연구를 통해서 Self-training의 가능성과 한계를 확인했고 실험 결과의 분석을 통해 후속 연구의 방향성을 제시할 수 있었다.

II 01. Pseudo labeling POC

Introduction

연구 주제: unlabeled data를 활용한 병변 탐지 인공지능의 성능 고도화 방안 연구

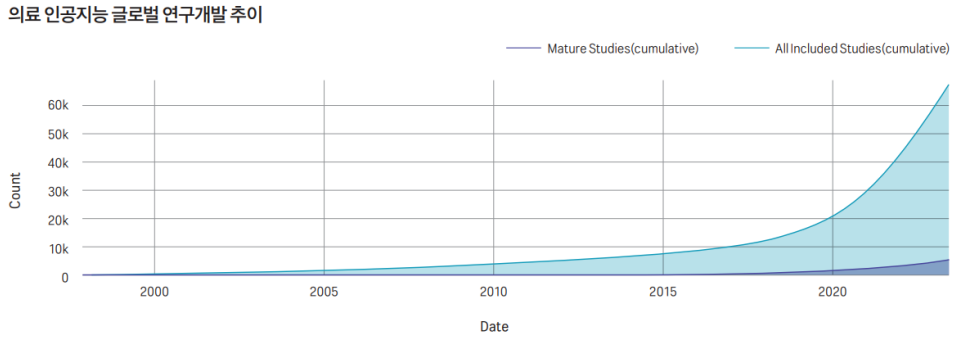
세부 연구 주제: Self-training 적용 및 구현. 성능 향상 가능성 검증

연구 배경

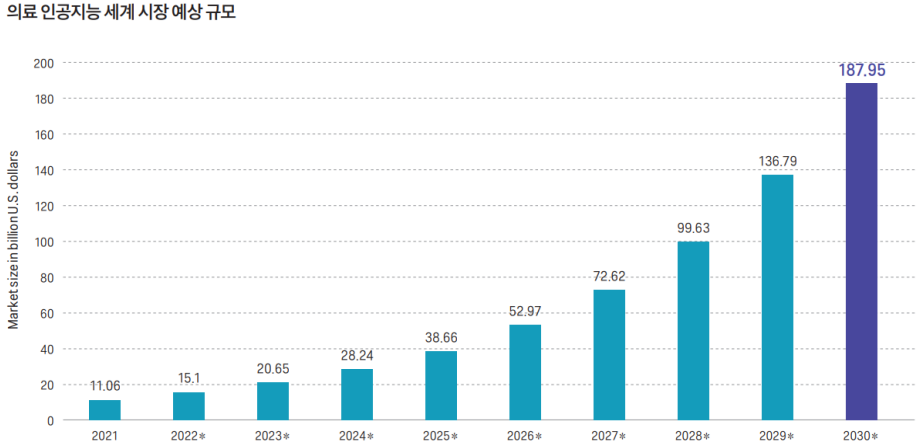
- 최근 빠른 속도로 발전하는 인공지능 기술을 의료 분야에 적용하고자 하는 연구가 활발히 진행되고 있다.
- 인공지능의 핵심 기술 중 하나인 딥러닝(deep learning)은 많은 데이터를 필요로 하지만 의료 분야의 경우 데이터를 수집하는 것이 매우 비싸고 어렵다.
- 이러한 문제를 해결하기 위해 다양한 분야에서 제도적 논의와 연구가 진행되고 있다.

연구 동기

- 대량의 데이터가 필요한 딥러닝 기술의 한계를 극복하기 위해 다양한 연구가 이루어지고 있고 그 중 하나의 방법으로 unlabeled data를 활용하고자 했다.
- 하지만 unlabeled data를 활용하는 대부분의 연구가 classification task에 대한 실험과 검증으로 이루어져 있다.
- 이러한 선행 연구의 한계를 확인하고 의료 분야의 segmentation task에서도 충분히 활용 가능한 방법을 확보하고자 이번 연구를 진행하게 되었다.

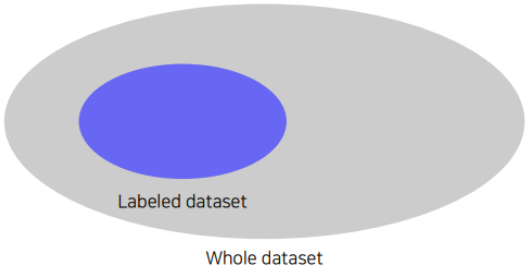


출처: Global Clinical Artificial Intelligence Dashboard(https://aiforhealth.app/)



출처: Statista, Artificial intelligence (AI) in healthcare market size worldwide from 2021 to 2030

Is there any way to learn from unlabeled data?

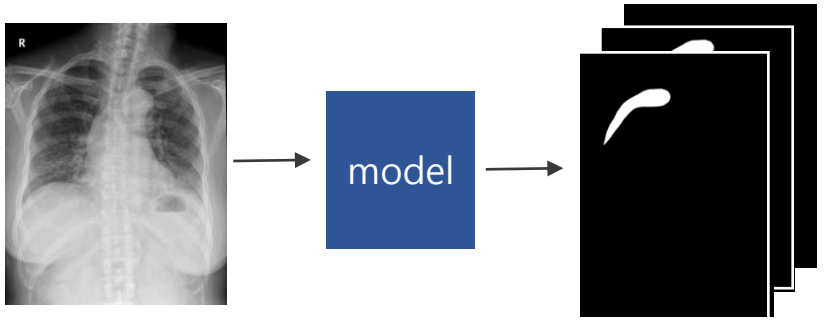


II 01. Pseudo labeling POC

Introduction

태스크 소개

흉부 X선 영상 속 병변의 위치를 질환별로 구분하여 segmentation 하는 문제



Input: Chest X-ray(CXR) image

Output: 병변 영역을 픽셀 단위로 구분한 mask (+ bounding box)

Target disease

- **1-finding**: Pneumothorax (PTX)
- **5-finding**: Consolidation (CNS), Pneumothorax (PTX), Fibrosis (FIB), Effusion (PEF), Nodule (NDL)

Dataset

- labeled dataset : BORA, CB, PAIK, KU (4개 기관에서 전문가에 의해 labeling된 private data), 약 5만장
- unlabeled dataset : SIIM, MIMIC, NIH, VBD, CXD, ETC (인터넷 상에 공개된 open data), 약 50만장

평가 지표

- Dice based AUROC(D-AUC) , Dice (threshold agnostic metrics)
- Sensitivity(SEN), Specificity(SPE)

검증 방법

- external validation: BORA 데이터에 대한 D-AUC와 Dice, SEN, SPE를 계산해 평가
- Self-training의 iteration이 증가함에 따라 모델의 성능 증감 추이를 관찰

DEEP NOID

Related work

Self-Training & Noisy Student Training

Supervised learning

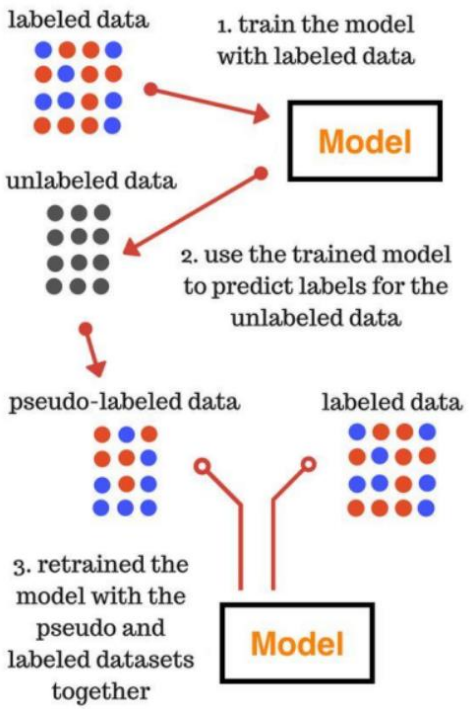
Semi-supervised learning

- **Cosistency-based method** – smoothness assumption
  - Consistency learning – consistency loss 활용
    ex) UDA, Mean Teacher
- **Entropy-minimization method** – cluster assumption
  - Self-Trainig – pseudo label 활용
    ex) Noisy Student, Meta-pseudo Label

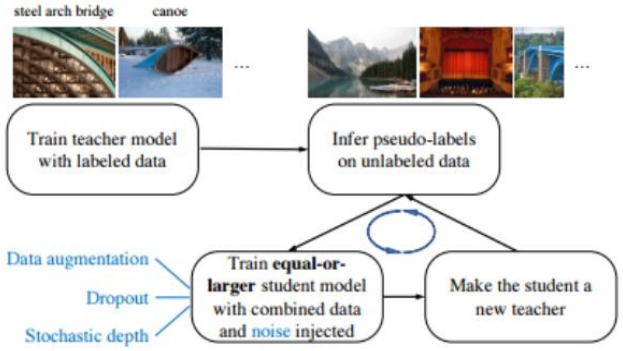
Unsupervised learning

- **Self-supervised learning**
  - pretext task를 해결하며 data 자체의 representation 학습
    - Contrasitve learning – contrastive loss 활용
      ex) simCLR, MoCo

Self-Training



Noisy Student Training



- 2020년 ImageNet classifiction에서 SOTA 달성
- 기본적인 방법은 Self-Training을 따르지만 student model에 추가되는 noise와 점진적으로 증가하는 model의 크기를 강조
- 특히 모델의 robustness 관점에서의 성능이 많이 향상 됨

DEEP NOID

Related work

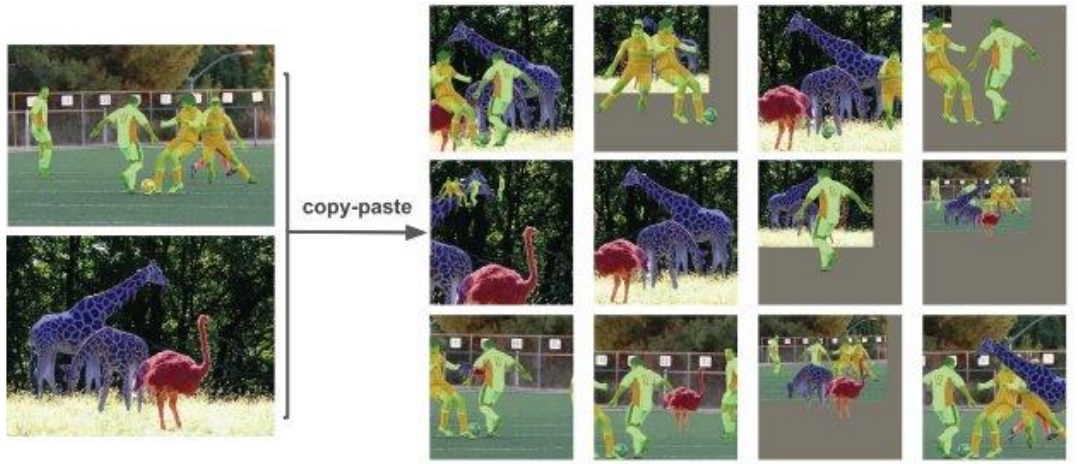
Noise: RandAugment & Simple Copy-Paste

RandAugment

```
transforms = [
'Identity', 'AutoContrast', 'Equalize',
'Rotate', 'Solarize', 'Color', 'Posterize',
'Contrast', 'Brightness', 'Sharpness',
'ShearX', 'ShearY', 'TranslateX', 'TranslateY']

def randaugment(N, M):
"""Generate a set of distortions.

  Args:
    N: Number of augmentation transformations to
        apply sequentially.
    M: Magnitude for all the transformations.
"""

  sampled_ops = np.random.choice(transforms, N)
  return [(op, M) for op in sampled_ops]
```

최적의 Augmentation 조합을 찾기 위한 다양한 변수들을
단, 2개의 파라미터(N, M)로 줄이면서도 좋은 성능을 유지한
Augmentation 방법론

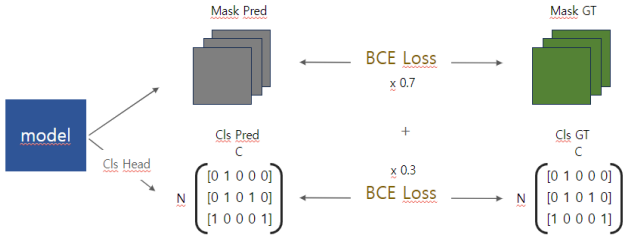
Simple Copy-Paste



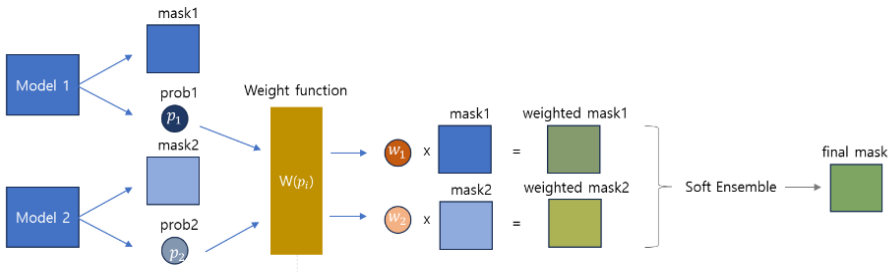
Instance에 대한 Visual Context를 고려하지 않고 random하게 copy-paste한는 것
만으로도 Instance segmentation task에서 성능을 향상 시킬 수 있음을 검증

Method

1. 기본적인 학습의 Framework는 **Self-Training** 방식을 따름
2. Noisy Student Training에서 강조한 2가지 contribution 적용
   - Noise: **RandAugment**, Dropout
   - **Student model size ≥ Teacher model size**
3. **Simple Copy-Paste** – normal case와 abnormal case의 class imbalance 문제 완화
4. **Auxiliary classification** - 모델의 robustness를 높임과 동시에 classification probability와 mask probability를 모두 활용
5. **Weighted-Soft Ensemble** - Pseudo label 생성 시 높은 품질의 pseudo label 생성을 위해 Ensemble 적용
   - 각 모델의 confidence를 고려해 weight를 부여할 수 있도록 구현



Auxiliary classifier



Weighted-soft ensemble

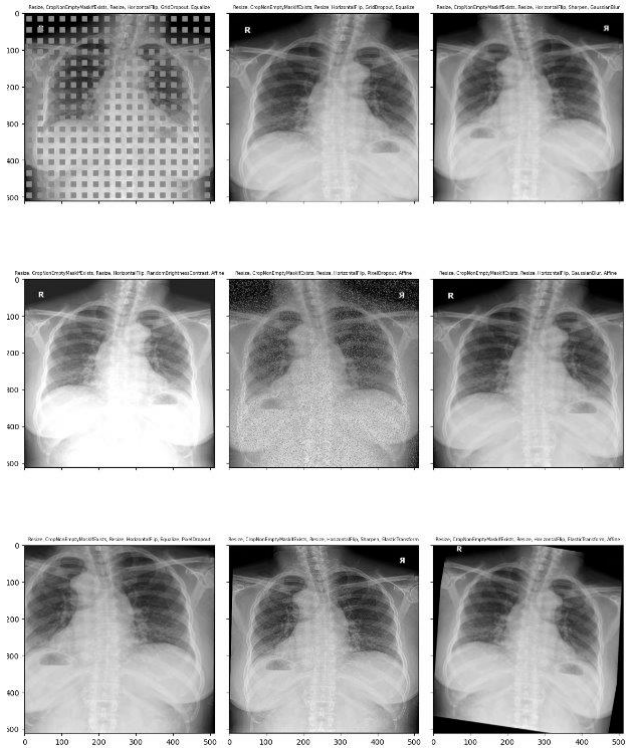
Copy-Paste 적용예시

DEEP NOID

Method

RandAugment

- Default augment: Crop, Resize, HorizontalFlip
- Random augment:
  ElasticTransform, Affine, GridDropout, PixelDropout, GaussianBlur, Sharpen, Equalizer, Contrast,



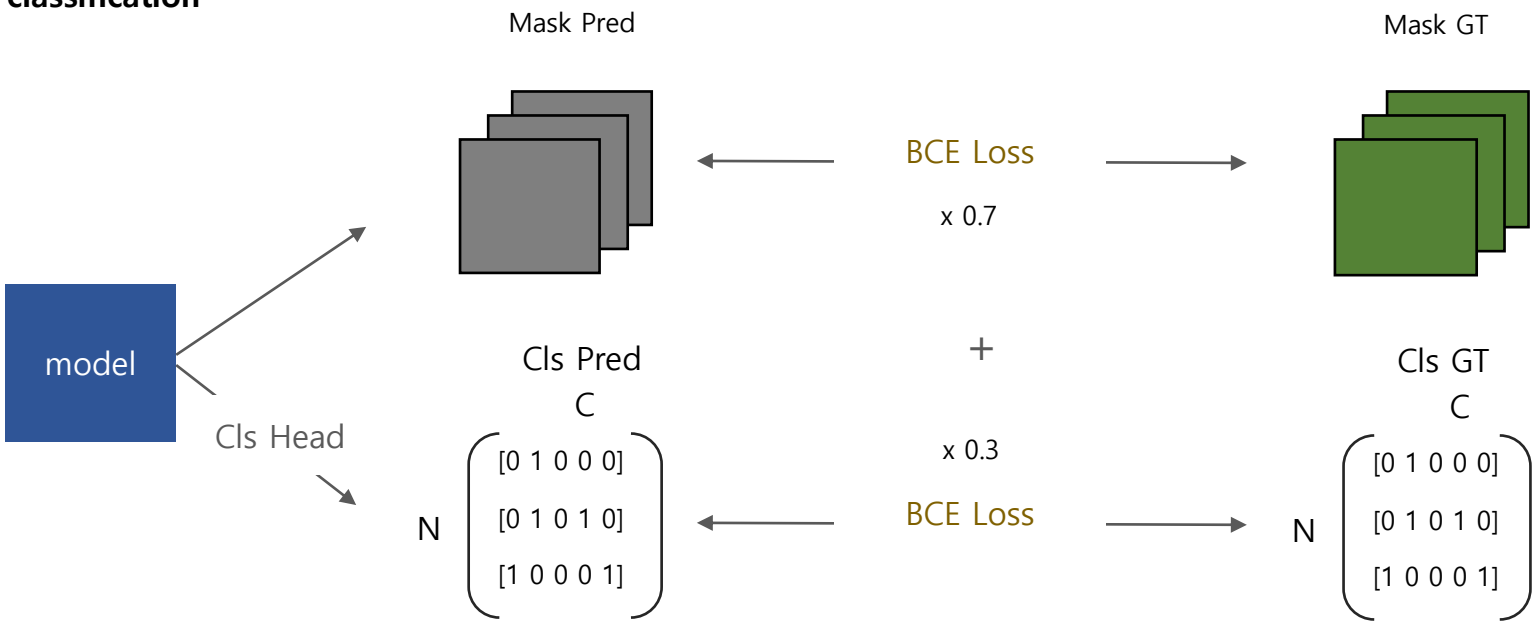
RandAugment 적용 예시

```python
class MyRandAugment:

    def __init__(self, n, m):
        self.n = n
        self.shift_x = np.linspace(0,0.1,10)[m]
        self.shift_y = np.linspace(0,0.1,10)[m]
        self.scale = np.linspace(0, 0.5, 10)[m]
        self.rot = np.linspace(0,40,10)[m]
        self.post = [4,4,5,5,6,6,7,7,8,8][m]
        self.cont = [np.linspace(-0.8,-0.1,10),np.linspace(0.1,2,10)][m]
        self.bright = np.linspace(0.1,0.7,10)[m]
        self.shar = np.linspace(0.1,0.9,10)[m]
        self.cut = np.linspace(0,60,10)[m]

    def __call__(self, img):
        min_size = min(img.shape[:2])
        crop_size = random.randint(round(min_size*0.8), min_size)
        fill_value = img.mean() if np.random.rand() > 0.5 else 0.0

        # default
        default_Aug = [
            A.CropNonEmptyMaskIfExists(crop_size, crop_size, p=0.5),
            A.Resize(512, 512, interpolation=cv2.INTER_AREA),
            A.HorizontalFlip(p=0.5)
        ]
        # geo
        geo_Aug =[
                A.ElasticTransform(border_mode=cv2.BORDER_CONSTANT, approximate=True, interpolation=cv2.INTER_AREA, alpha_affine=20.0, p=0.5),
                A.Affine(rotate=(-20, 20), p=0.5),
        ]
        # noise
        noise_Aug = [
                A.GridDropout(unit_size_min=25, unit_size_max=50, fill_value=fill_value, random_offset=True, p=0.3),
                A.PixelDropout(dropout_prob=0.2, drop_value=fill_value, p=0.3),
                A.GaussianBlur(p=0.5),
                A.Sharpen(p=0.5),
        ]
        # color
        color_Aug = [
                A.Equalize(p=0.5),
                A.RandomBrightnessContrast(p=0.5),
                A.Sharpen(p=0.5)
        ]

        ops = default_Aug + random.sample(geo_Aug + noise_Aug + color_Aug, self.n)
        transforms = A.Compose(ops)

        return transforms
```

DEEP NOID

Method

Auxiliary classification

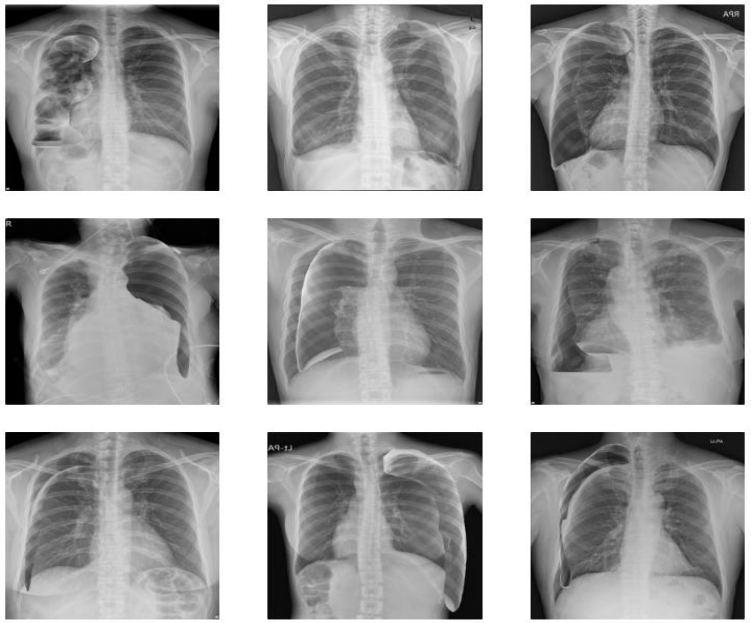



```python
class ClassificationHead(nn.Sequential):
    def __init__(self, in_channels, classes, pooling="avg", dropout=0.2, activation=None):
        if pooling not in ("max", "avg"):
            raise ValueError("Pooling should be one of ('max', 'avg'), got {}.".format(pooling))
        pool = nn.AdaptiveAvgPool2d(1) if pooling == "avg" else nn.AdaptiveMaxPool2d(1)
        flatten = nn.Flatten()
        dropout = nn.Dropout(p=dropout, inplace=True) if dropout else nn.Identity()
        linear = nn.Linear(in_channels, classes, bias=True)
        activation = Activation(activation)
        super().__init__(pool, flatten, dropout, linear, activation)
```

Auxiliary classification 실험 결과: Appendix - auxiliary classification 참고

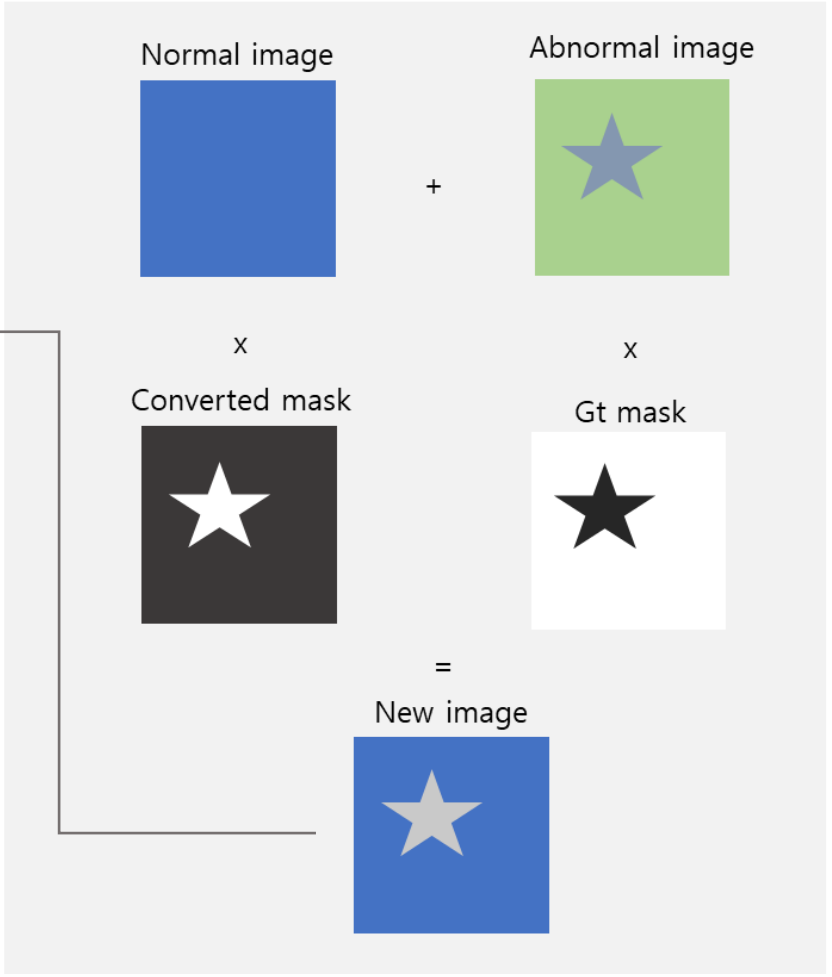
Method

Simple Copy-Paste



Copy-paste 적용 예시

Normal case인 경우 0.5 의 확률로 Abnormal case의 label을 sampling



Copy-Paste 3가지 옵션

- Copy-paste 적용 확률
- 한 입력 이미지에 대해 copy-paste를 적용하는 횟수

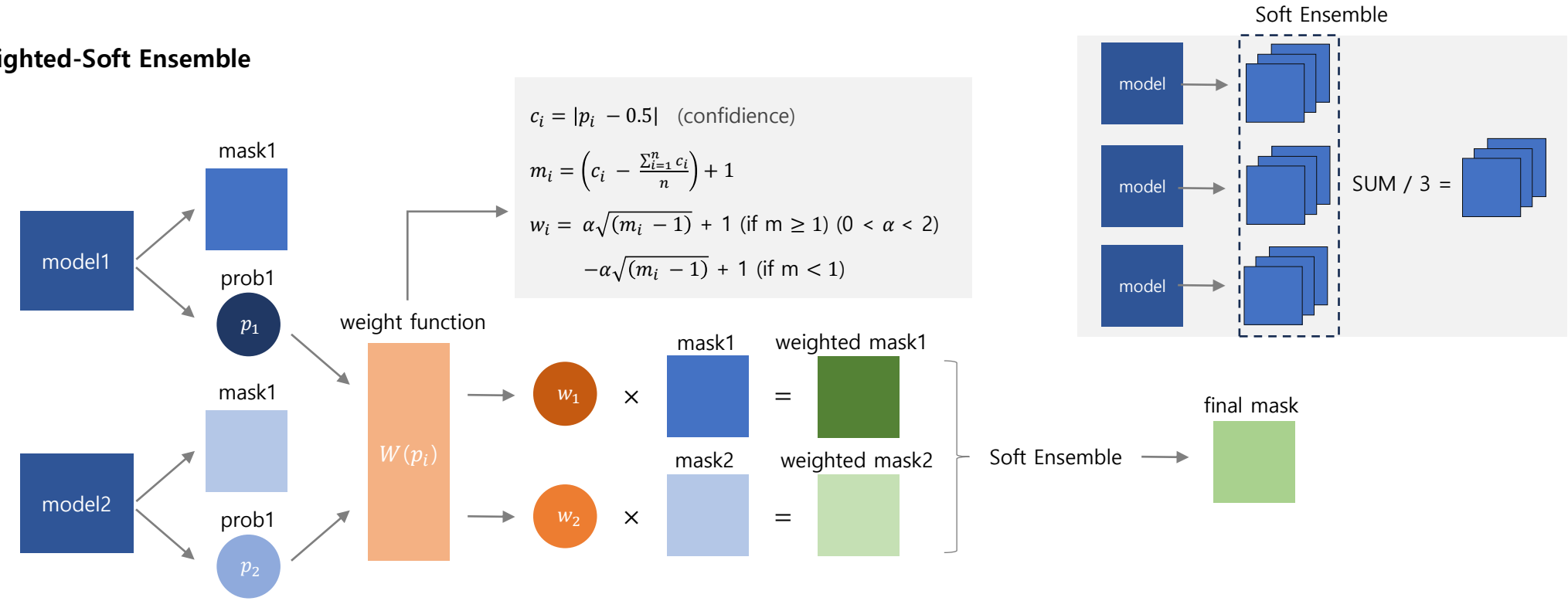
  ex) 0.5의 확률로 2번 적용
- Nomal case의 이미지에 대해서만 copy-paste 적용 or 모든 입력 이미지에 대해서 적용
  - 1-finding에서는 normal case에 대해서만 적용
  - 5-finding에서는 모든 case에 대해서 적용

Copy-paste 실험 결과: Appendix - Copy-Paste 참고

DEEP NOID

Method

Weighted-Soft Ensemble



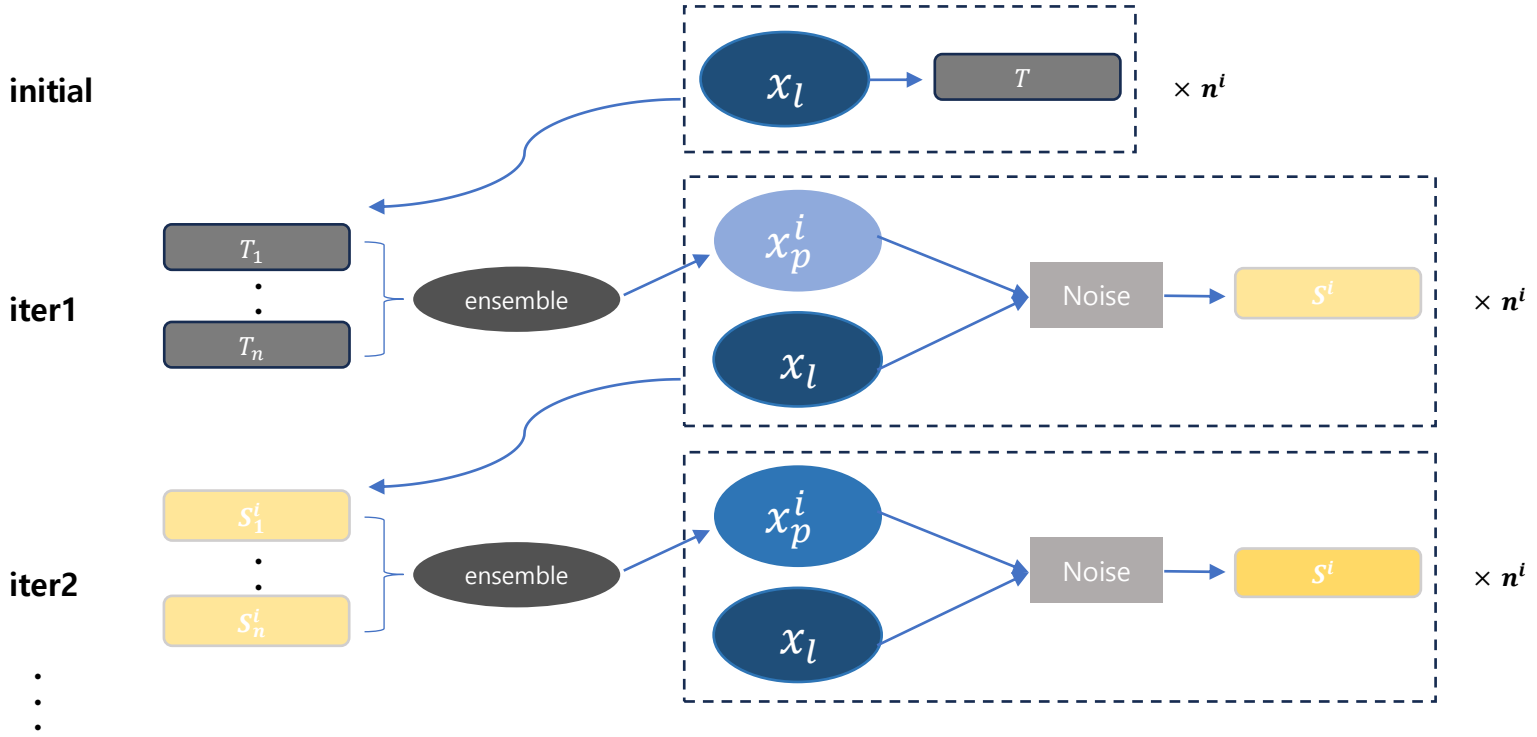
- Classifier의 probability 값을 반영한 weight를 구해 ensemble에 활용
- 기본 아이디어: 입력 이미지에 대해 애매하게 예측한 모델의 probability map은 더 애매하게 만들고 확실하게 예측한 모델의 probability map은 더 확실하게 만든다.
  - 애매하게: confidience가 낮은 경우
  - 확실하게: confidience가 높은 경우
- W(p)에 들어가는 probability 값은 classifier의 probability 값만 사용하는 것 보다 mask의 max probability 값을 5:5로 섞었을 때 더 좋은 결과를 얻을 수 있었다.

Ensemble 실험 결과: Appendix - weighted-soft ensemble 참고

DEEP NOID

Method

Training Framework



ensemble : Weighted-Soft Ensemble
noise: RandAugment, Simple Copy-Paste, Dropout

i = iteration
n = number of teacher model
T = teacher
S = student
x_l = labled data
x_p = pseudo labeled data

Model				Pseudo label threshold			
	Architecture	Backbone	Size(params)	1finding		5finding	
				pos	neg	pos	neg
initial	UNet	EfficientNet-B5	30M	0.85	0.001	0.97	0.005
iter1	UNet	EfficientNet-B6	42M	0.7	0.002	-	-
iter2	UNet	EfficientNet-B7	65M	-	-	-	-

Self-Training setting

DEEP NOID

Experiments & Results

Dataset

OPEN : SIIM, MIMIC, NIH, CXD, VBD, ETC

		Training and internal validation					External validation
		T=initial			*T=1*	*T=2*	
		Labeled			Unlabeled (Pseudo label)		
		CB	KU	PAIK	OPEN	OPEN	BORA
1-finding	normal	14,162	14,897	5,641	2,508	9,423	15,068
	abnormal	1,058	1,287	960	66,158	265,219	1,155
	total	15,220	16,184	6,601	68,666	274,642	16,223
5-finding	normal	7,200	6,920	237	16,796	-	6,285
	abnormal	8,020	9,264	6,364	19,412	-	9,938
	total	15,220	16,184	6,601	36,208	0	16,223

Hyperparameter Setting

Epoch	50 ~ 100	RandAugment	n= 2, m = default
Batch Size	16	Dropout	0.25
Optimizer	Adam	Image Size	512 x 512
Learning Rate	0.0001	Loss rate (cls : mask)	0.3 : 0.7

※ 1finding iter2에서는 Batch Size = 32, Epoch = 32

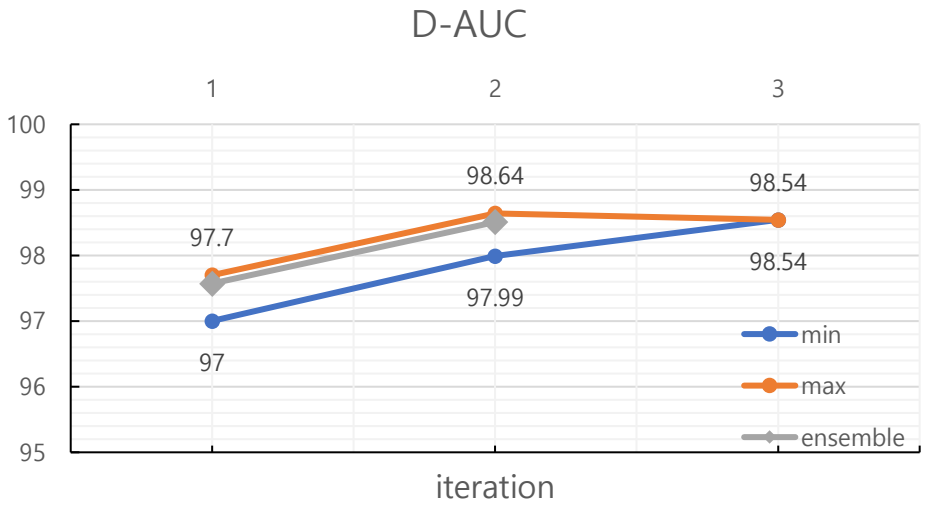
II 01. Pseudo labeling POC

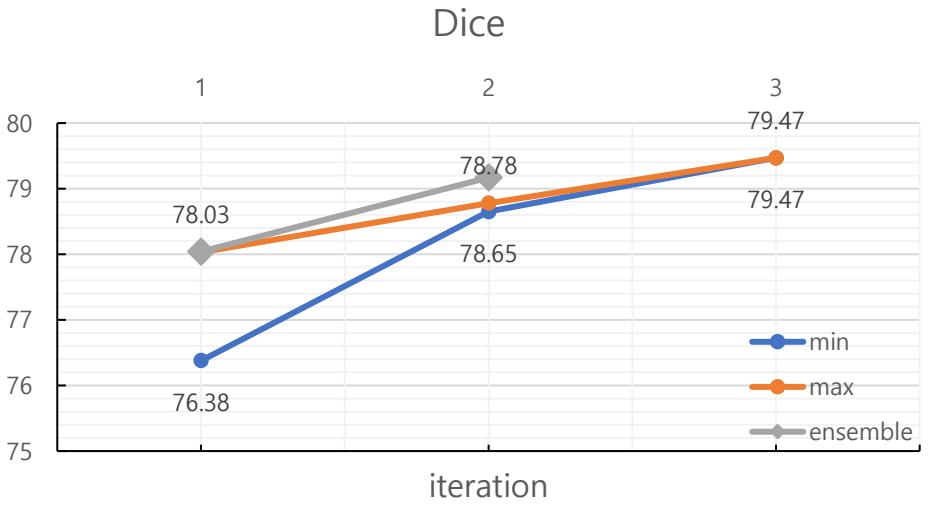
DEEP NOID

Experiments & Results

1-finding Result

	Name	모델	Ann_ver	AUROC	DICE	민감도	특이도
initial	exp20	effiU-b5_copypaste25_noise1	-	97.00	76.38	94.03	99.87
	exp21	effiU-b5_copypaste50_noise1	-	97.70	78.03	95.41	99.83
	exp22	effiU-b5_copypaste75_noise1	-	97.12	77.96	94.29	99.6
	exp20 + exp21 + exp22	ensemble	-	97.57	78.04	95.15	99.90
-	exp23	effiU-b6_copypaste25_noise1	-	97.01	77.28	94.03	99.87
iter1	exp24	effiU-b6_copypaste25_noise1-v5	v5	97.99	78.65	96.02	99.62
	exp25	effiU-b6_copypaste50_noise1-v5	v5	98.64	78.78	96.80	98.89
	exp24 + exp25	ensemble	-	98.51	79.17	96.80	99.60
iter2	exp27	effiU-b7_aux-copypaste50-noise1 -v6	v6	98.54	79.47	97.14	99.32





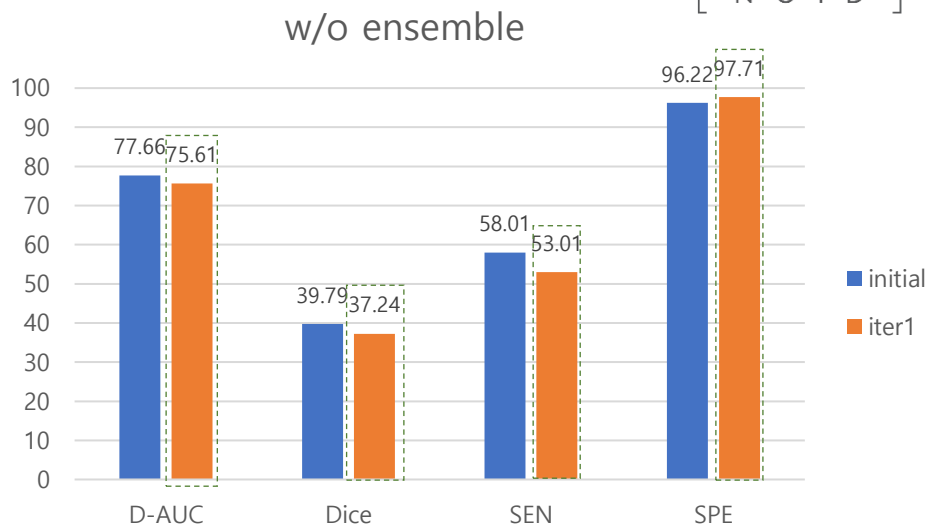
II 01. Pseudo labeling POC

Experiments & Results

5-finding Result - (average)

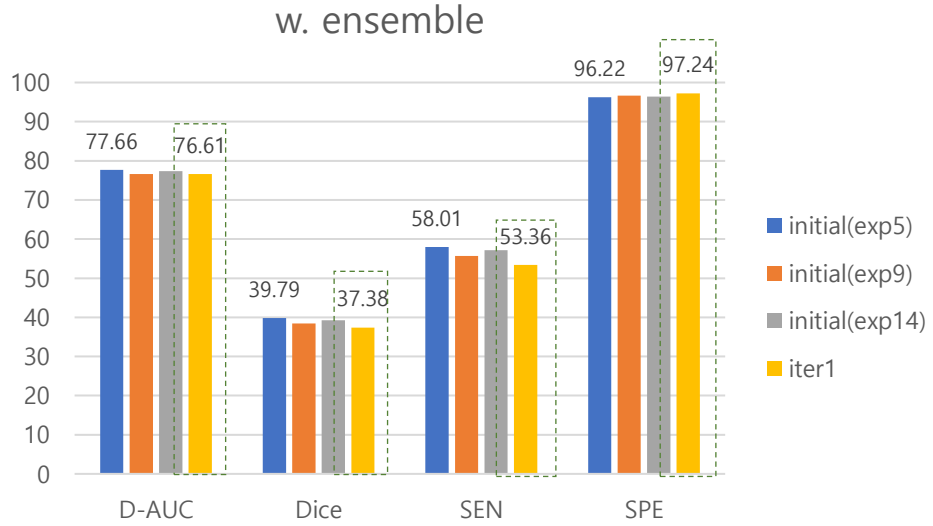
w/o ensemble

	Name	모델	AUROC	DICE	민감도	특이도
initial	exp5	effiU-b5_aux_noise1-copypaste75	77.66	39.79	58.01	96.22
iter1	exp24	effiU-b6_aux-noise1-copypaste75-v2	75.61	37.24	53.01	97.71



w. ensemble

	Name	모델		AUROC	DICE	민감도	특이도
initial	exp5	effiU-b5_aux_noise1-copypaste75		77.66	39.79	58.01	96.22
	exp9	effiU-b3_aux-noise1-loss55		76.59	38.44	55.70	96.62
	exp14	effiU-b3_aux-noise2-loss19-copypaste25-2		77.31	39.27	57.16	96.39
	exp5 + exp9 + exp14		ensemble	77.27	38.96	56.71	97.12
iter1	exp25	effiU-b6_aux-noise1-copypaste75-v3		75.61	37.38	53.36	97.24



II 01. Pseudo labeling POC

DEEP NOID

Discussion

- Self-training을 병변 영역 분할(disease segmentation) 문제에 적용하고 1-finding과 5-finding 실험을 통해 Self-training의 가능성 및 한계점을 확인할 수 있었다.
- 소량의 labeled 데이터로도 성능이 보장되는 task라면 간단한 pseudo labeling을 통해 unlabeled 데이터를 효과적으로 활용 할 수 있다.
- 하지만 task 자체의 난이도가 있거나 labeled 데이터가 매우 부족한 상황이라면 Self-training이 confirmation bias 문제를 발생시키며 오히려 성능을 하락시킬 수 있다.
  - pseudo label은 실제 label이 존재하지 않기 때문에 정량적인 방법으로 품질을 평가하기 힘들다.

1) open 데이터셋에서 샘플을 추출해 직접 레이블링한 데이터로 평가 -> 가장 신뢰도가 높지만 제약이 많음
2) 병원 label에 대한 성능을 통해 간접적인 방법으로 품질을 비교한다 (병원 데이터와 open data 사이 domain gap이 존재)
3) label을 제공하는 open 데이터셋을 활용한 간접적인 평가 (SIIM 데이터의 경우 PTX에 대한 mask label 제공)
4) pseudo label의 시각화를 통해 정성적인 방법으로 평가

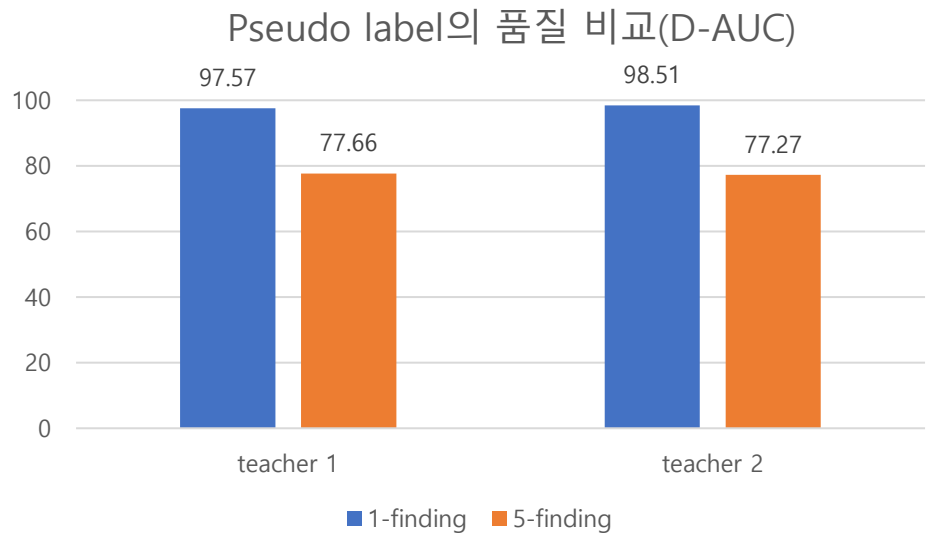
↓

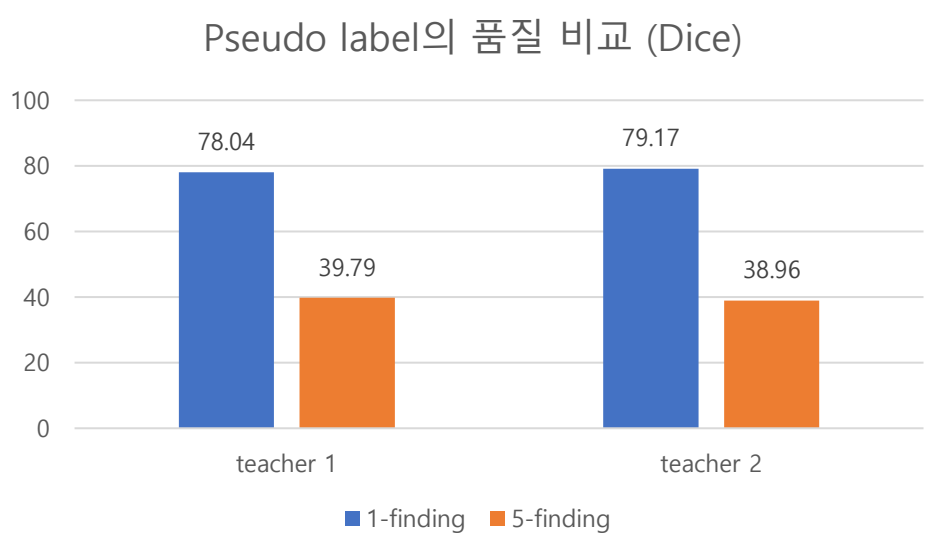
간접적인 방법을 통해 pseudo label을 평가하여 confirmation bias 문제가 발생함을 확인

Discussion

Pseudo label의 정량적 평가

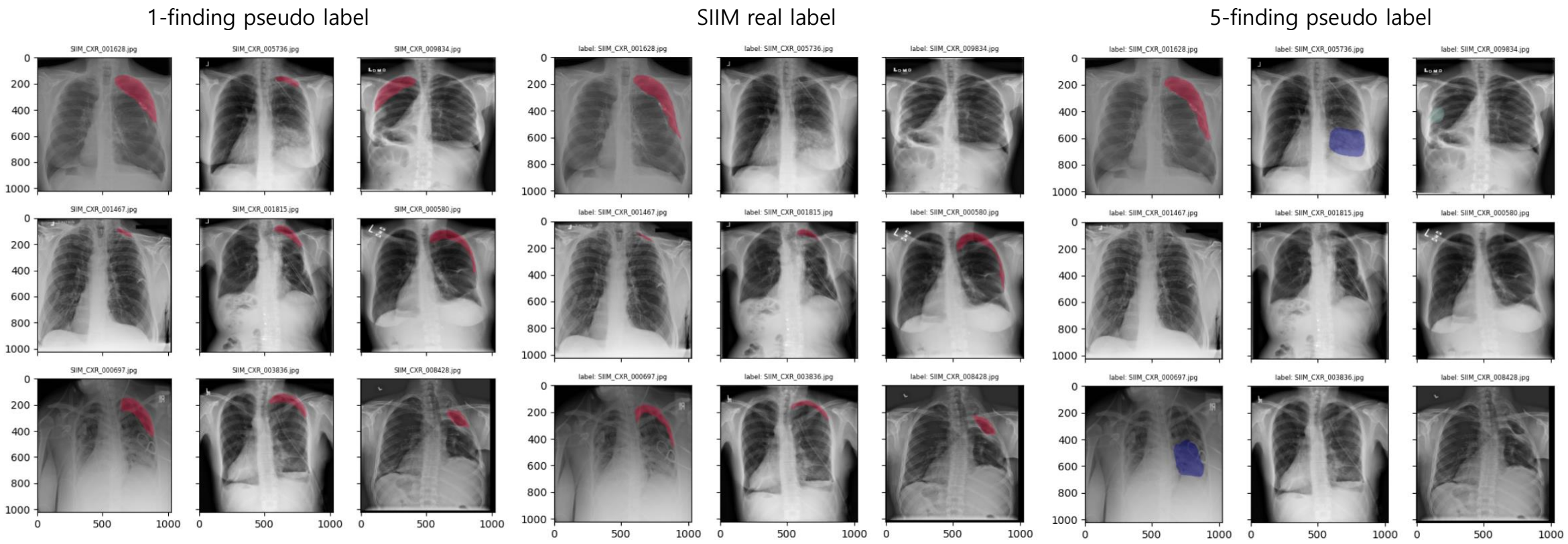
- 1-finding 과 5-finding의 Teacher 모델의 성능을 D-AUC와 Dice로 비교
- BORA 데이터로 평가





Discussion

Pseudo label의 정성적 평가



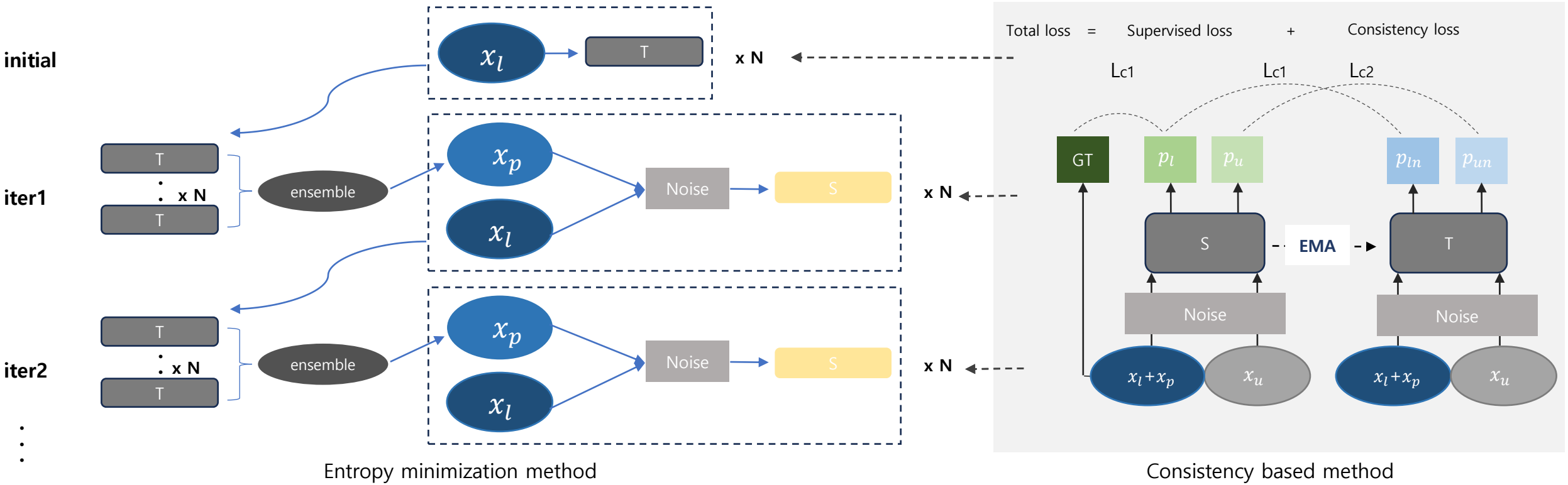
- 정성적 평가와 정량적 평가에서 모두 5-fidning 태스크에 대한 pseudo label의 품질이 매우 하락한 모습을 볼 수 있다
- Self-training 적용 시 특히 초기 단계에서 생성되는 pseudo label의 품질이 매우 중요하다.
  -> 나머지 단계에 비해 초기 단계에서의 모델 최적화 과정이 중요

Discussion

Future work

- Initial 단계에서의 pseudo label 품질을 높이기 위해 초반부터 unlabeled 데이터를 학습에 활용할 수 있는 있어야 한다
  - 더 많은 데이터를 초기 단계에서부터 활용 가능
  - labeled 데이터와 unlabeled 데이터 사이의 domain gap으로부터 발생하는 pseudo label 품질 하락 문제를 방지

- 개성 방안
  - 기본적인 Self-training framework에 Consistency 기반의 방법을 함께 적용



Entropy minimization method

Consistency based method

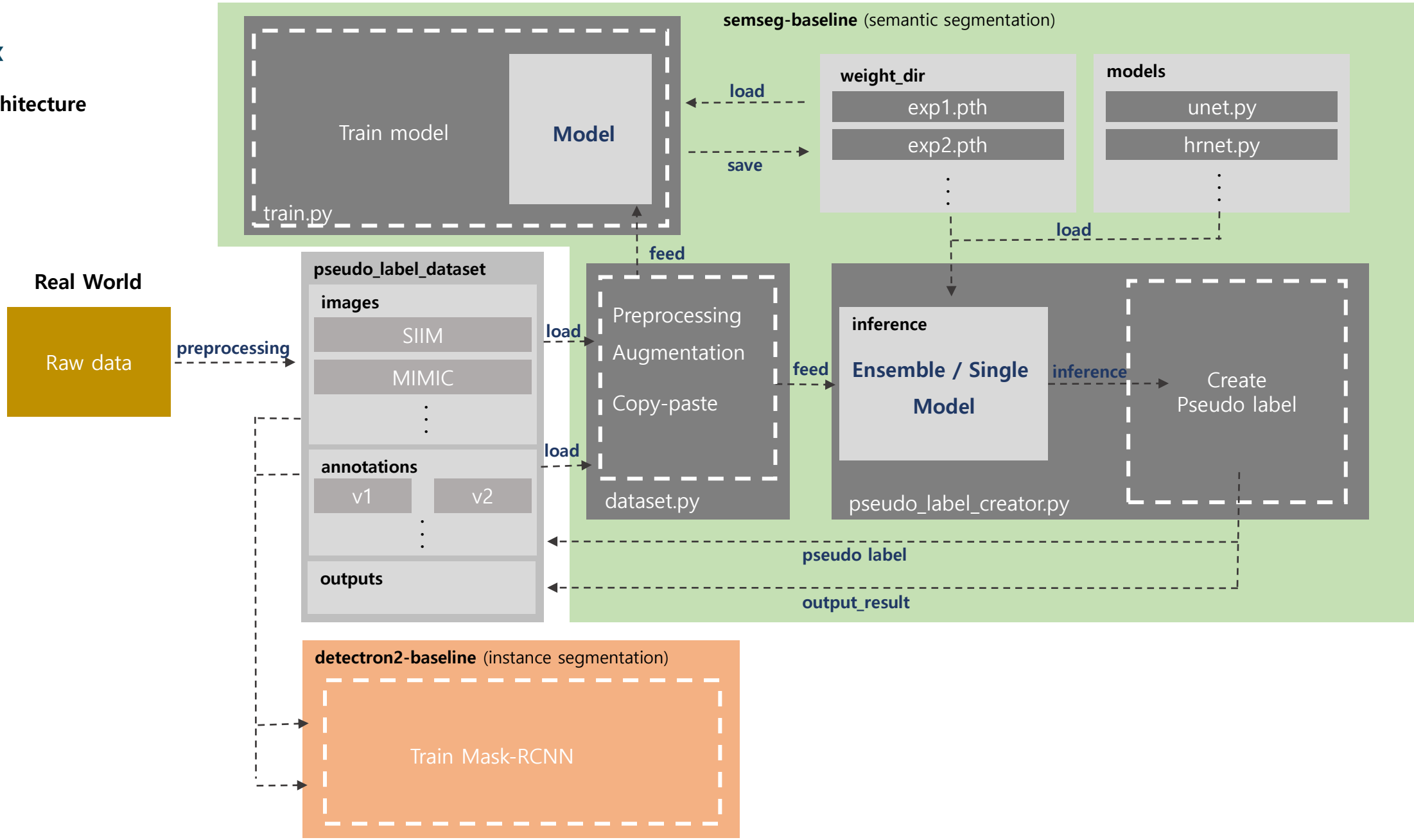
Conclusion

- 이번 연구는 의료 분야의 병변 분할 문제에 대량의 unlabeled data를 효과적으로 활용할 수 있는 방법을 탐구했다.
- 그 첫번째 단계로 가장 단순(simplest)하고 대부분의 상황에 활용 가능한(generally applicable)한 Self-training을 적용했다.
- 그 결과 Self-traing의 가능성과 한계점을 파악할 수 있었고 Self-training의 한계 분석을 통해 후속 연구의 방향성을 제시할 수 있었다.
- 이번 연구의 실험과 결과가 의료 분야의 unlabeled data 활용 및 연구에 유의미하게 사용될 수 있기를 희망한다.

DEEP NOID

Appendix

Project Architecture



DEEP NOID

Appendix

Process of creating pseudo labels

Preprocess

- Dicom -> jpg
- 1024 x 1024
- alpha channel 제거
- 기관별로 정리

→

Dataset

- 512 x 512
- RandAugment
- copypaste
- ToTensor

→

Teacher Model

Ensemble
or
Single Model

→

Postprocess

Step1. erosion (노이즈 제거)
Step2. dilation (빈 픽셀 메우기)
Step3. erosion (복원)
Step4. resize 1024 x 1024

class1 class2 class3

→

Finding Contour

polygon

$[[x_1, y_1], [x_2, y_2], \ldots [x_n, y_n]]$

box

$[x_{min}, y_{min}, x_{max}, y_{max},]$

→

Making label

image_id,
image_name,
institution,
target_classes,
width, is_normal,
annotations

Pseudo_label_dataset

- chestALL
  - images
    - BORA
    - CB
    - PAIK
    - KU
    - NIH
    - SIIM
    - ....
  - annotations
    - 1findings
      - v0
      - v1
      - ...
    - 5findings
      - v0
      - v1
      - ...

Dataset의 매개변수 값

IMG_ROOT
institution

ANN_ROOT
findings
ann_version

copypaste
num_copypaste

- 학습에는 anntation이 있는 데이터만 사용
- 1024 x 1024 기준으로 pseudo labeling
- Image는 jpg format, alpha channel은 제거
- Label은 json format
- Label과 image의 파일명은 동일 (image_id)
  - image_id: {institution}_CXR_{id}

Outputs

Output.csv
- 모델의 예측 결과 log
- Image_id, class_id, prob, box_coordinate

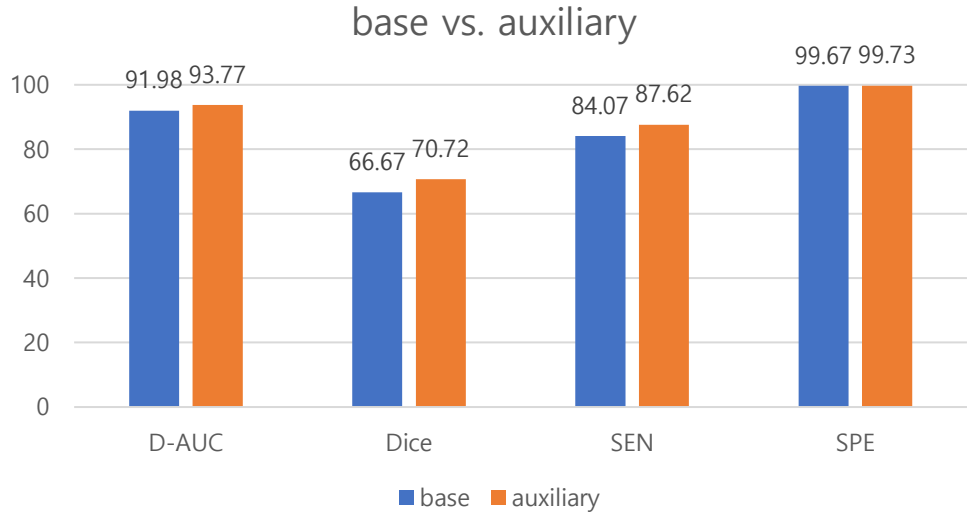
Output.json
- 생성된 pseudo label의 수량 분석
- annotation된 병변의 개수 분석

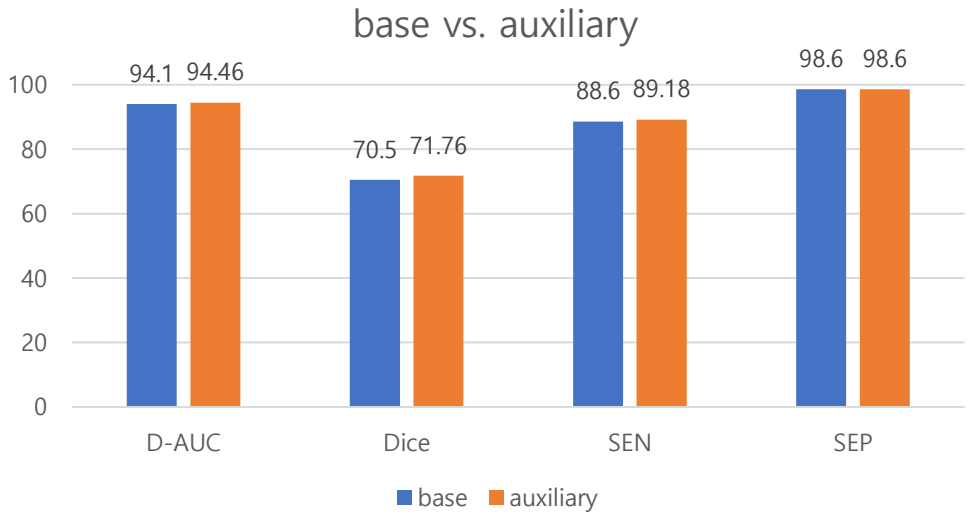
DEEP NOID

Appendix

Auxiliary classification

모델	effiU-b5			
copypaste	0%		50%	
auxiliary	base	auxiliary	base	auxiliary
D-AUC	91.98	93.77	94.11	94.46
Dice	66.67	70.72	70.45	71.76
민감도	84.07	87.62	88.57	89.18
특이도	99.67	99.73	98.58	98.6



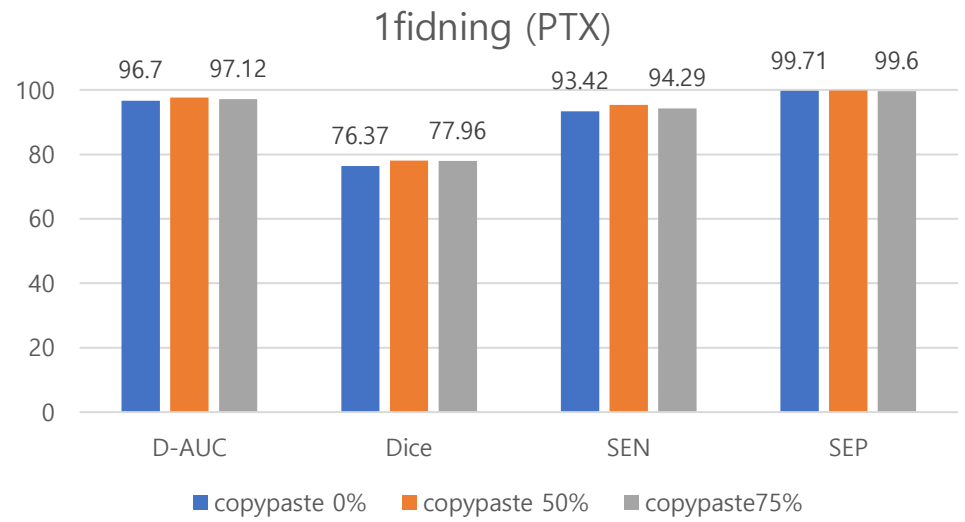


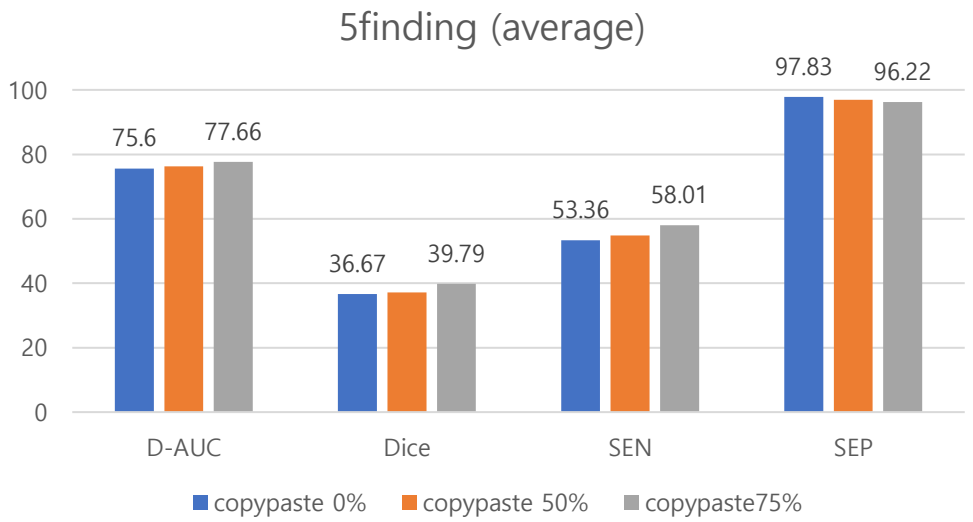
DEEP NOID

Appendix

Copy-Paste

모델	effiU-b5_aux					
fidning	1-finding			5-finding (average)		
copypaste	0%	50%	75%	0%	50%	75%
D-AUC	96.70	97.70	97.12	75.60	76.28	77.66
Dice	76.37	78.03	77.96	36.67	37.19	39.79
민감도	93.42	95.41	94.29	53.36	54.84	58.01
특이도	99.71	99.60	99.60	97.83	96.99	96.22





Appendix

Weighted-soft Ensemble

1-finding - PTX

model	EffiU-b5					EffiU-b6				
iter	initial					iter1				
ensemble	single model			ensemble		single model			ensemble	
exp	exp20	exp21	exp22	soft	weighted	exp24	exp25	-	soft	weighted
D-AUC	97.00	97.70	97.12	97.48	97.57	97.99	98.64	-	98.09	98.51
Dice	76.38	78.03	77.96	78.04	78.04	78.65	78.78	-	78.64	79.17
민감도	94.03	95.41	94.29	94.89	95.15	96.02	96.8	-	96.10	96.80
특이도	99.87	99.83	99.60	99.90	99.90	99.62	98.89	-	99.80	99.60

5-finding - (average)

model	EffiU-b3	EffiU-b5								
ensemble	single model			ensemble		single model			ensemble	
exp	exp14	exp5	-	soft	weighted	exp21	exp22	exp23	soft	weighted
D-AUC	77.31	77.66	-	77.24	77.79	74.95	74.65	74.35	74.53	75.05
Dice	39.27	39.79	-	38.91	39.60	36.16	35.91	34.71	35.46	36.11
민감도	57.16	58.01	-	56.65	57.89	51.94	50.90	50.33	50.45	51.67
특이도	96.39	96.22	-	96.93	96.83	97.60	97.78	97.84	98.21	98.10

Reference

1) Self-training with Noisy Student improves ImageNet classification
2) RandAugment: Practical automated data augmentation with a reduced search space
3) Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results
4) Simple Copy-Paste is a Strong Data Augmentation Method for Instance Segmentation
5) Unsupervised Data Augmentation for Consistency Training
6) A Simple Framework for Contrastive Learning of Visual Representations
7) Enhancing Self-Training Methods

감사합니다