2023 UNI-DTHON
Semantic Segmentation

[Indoor Image Semantic Segmentation]

CONTENTS

EDA

Class 살펴보기

Original Image

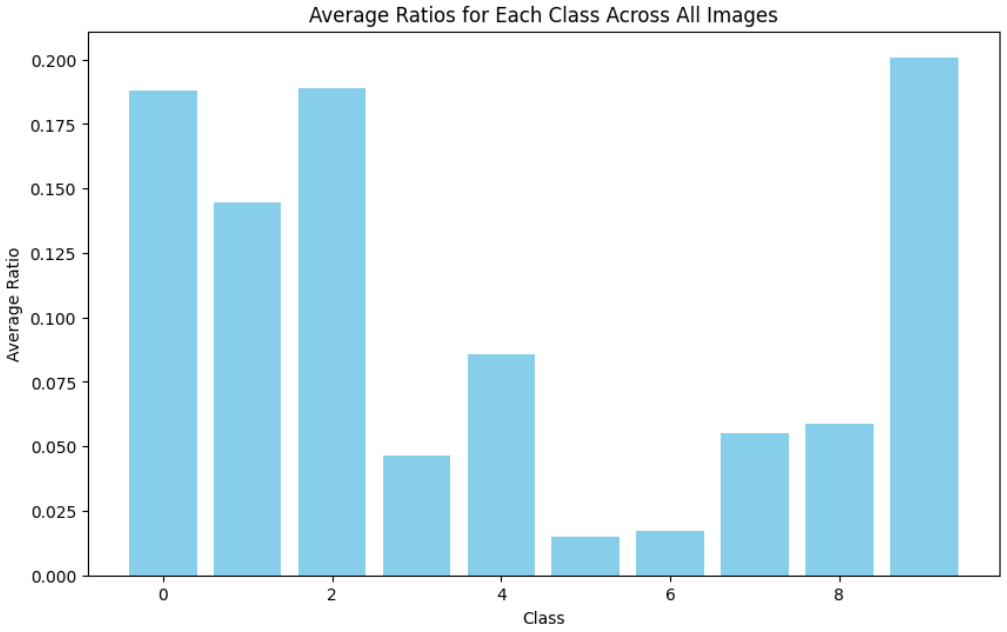
Mask

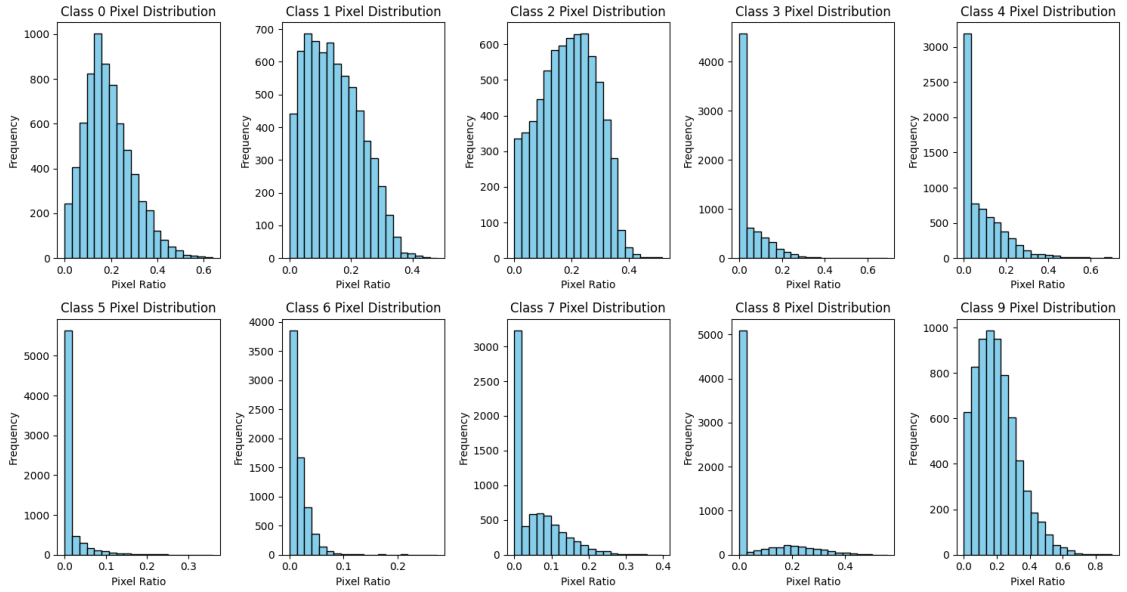
천장 등에 있는 작은 요소 존재

➔ Small Object Detection에 주목

EDA

Class 살펴보기





전체 이미지의 각 class 평균 비율 확인

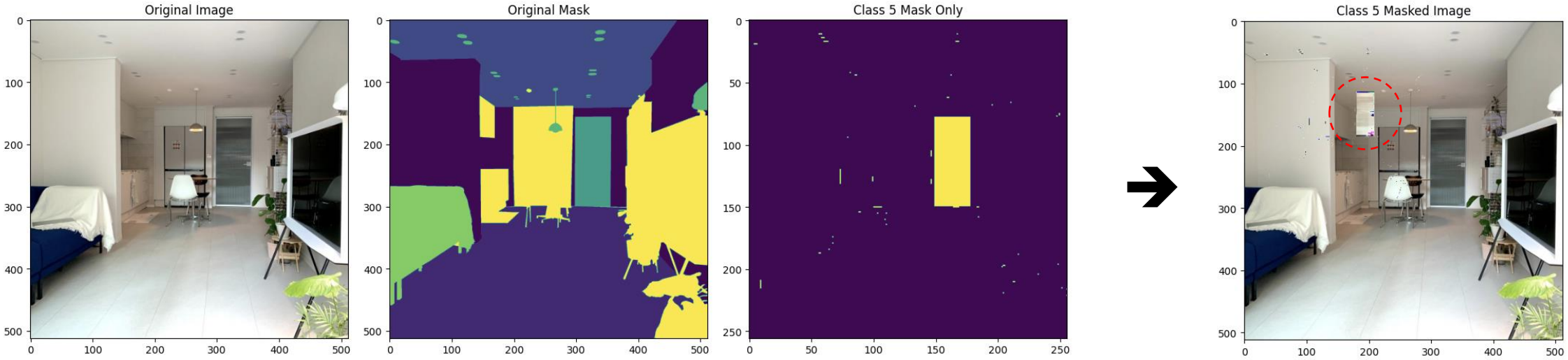
➔ **Class 5 면적이 작아 인식이 어려움**

Class별 픽셀 분포 시각화 확인, Class 5 분포 최저

➔ **Class 5 인식률이 낮음**

EDA

Class 5 문제 해결 시도



Class 5(Door) 인식률을 높이기 위해,
Class 5 마스크를 추출하여 원본 마스크의 랜덤으로 추가 시도

➔ 성능향상X, 모델 구축 과정에서 추가해결 시도

Model

모델 관리

WandB 사용하여 모델 관리

Model

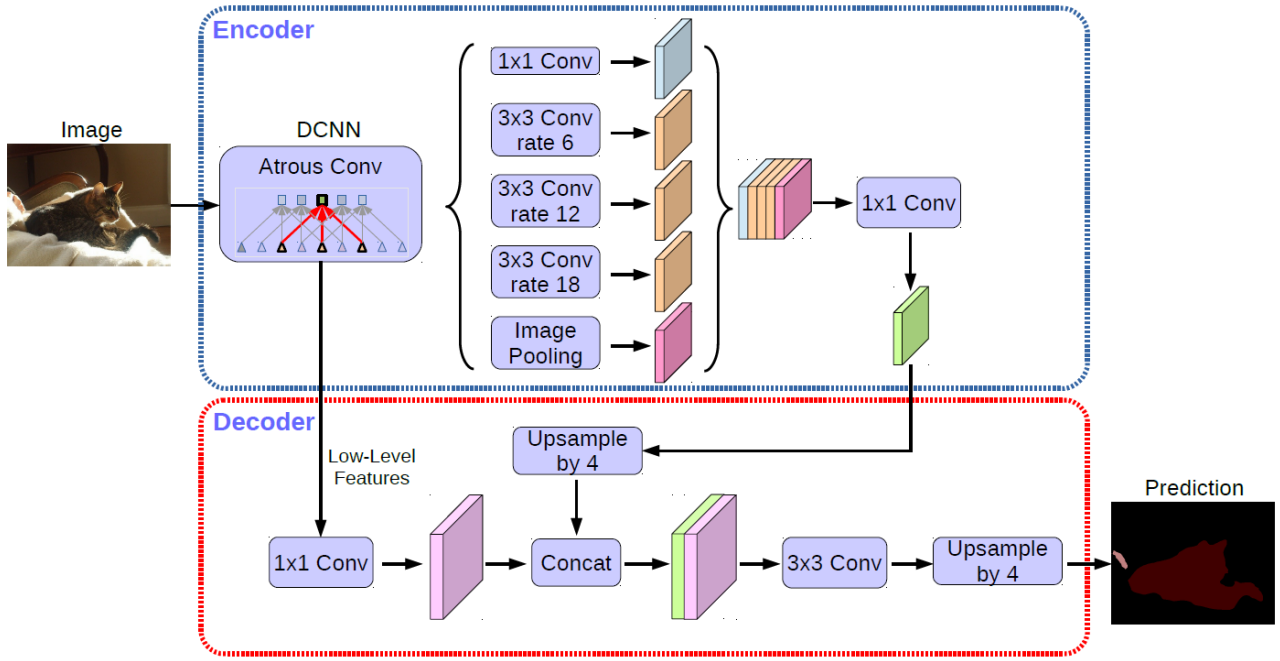
모델 구조

DeepLabv3+ 모델, Resnet101을 Feature Extractor로 활용

Segmentation Models

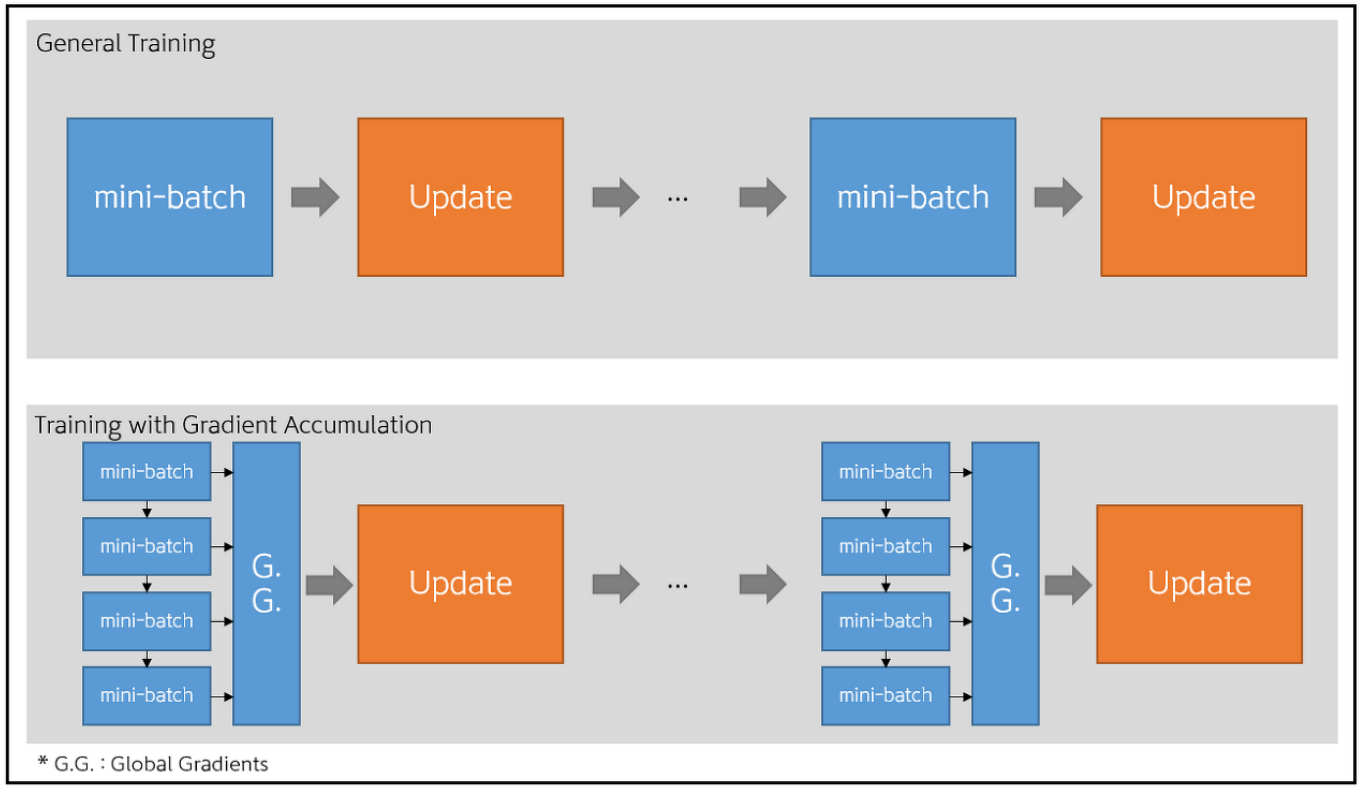
```
import segmentation_models_pytorch as smp

model = smp.Unet(
    encoder_name="resnet34",
    encoder_weights="imagenet",
    in_channels=1,
    classes=3,
)
```



Model

1. Accumulation



GPU 메모리 부족 현상 발생

Batch Size가 원인

Gradient Accumulation 적용

실질적인 Batch Size는 작지만, 크게 설정한 것과 같은 효과를 냈다

성능 개선!

```
loss = loss / accumulation_step
loss.backward()
if (i+1) % accumulation_step == 0:
    optimizer.step()
    model.zero_grad()
```

Model

2. Data Argumentation

```python
self.base_transform = A.Compose([
            A.Resize(*self.resize),
            A.HorizontalFlip(p=0.5),
            A.Normalize(),
            ToTensorV2()
])
```

[base]
이미지를 바꾸지 않음

```python
self.hard_transform = A.Compose([

    A.OneOf([
        A.HorizontalFlip(p=0.5),  # 수평 뒤집기
        A.Rotate(limit=10, p=0.5),                  # -10도에서 10도 사이 랜덤 회전
    ], p=0.75),  # OneOf로 랜덤 선택, p=1.0은 항상 적용하도록 함

    A.RandomBrightnessContrast(p=0.15),  # 랜덤한 밝기와 대비 조절
    A.RandomBrightness(p=0.15),                      # 랜덤 밝기 조절
    A.HueSaturationValue(p=0.15),                   # 색조, 채도, 명도 조절
    A.RandomGamma(p=0.15),                          # 랜덤 감마 조절
    A.GaussNoise(var_limit=(0.0, 25.0), p=0.15),     # 가우시안 노이즈 추가
    A.Sharpen(p=0.15),                              # 이미지 선명도 조절

    A.OneOf([
        A.RandomCrop(height=334, width=334, p=0.7),  # 무작위 크롭 (정사각형으로 crop)
        A.RandomCrop(height=384, width=384, p=0.8),  # 무작위 크롭 (정사각형으로 crop)
        A.RandomCrop(height=448, width=448, p=0.9),  # 무작위 크롭 (정사각형으로 crop)
        A.RandomCrop(height=496, width=496, p=1),  # 무작위 크롭 (정사각형으로 crop)
    ], p=0.3),  # OneOf로 랜덤 선택, p=1.0은 항상 적용하도록 함


    A.Resize(*self.resize),
    A.Normalize(),
    ToTensorV2()
])
```

[hard] 강한 색상 변환 및 crop

```python
self.mask_transform = A.Compose([

    A.OneOf([
        A.HorizontalFlip(p=0.5),  # 수평 뒤집기
        A.Rotate(limit=10, p=0.5),                  # -10도에서 10도 사이 랜덤 회전
    ], p=0.75),  # OneOf로 랜덤 선택, p=1.0은 항상 적용하도록 함

    A.GridDropout(holes_number_x=6, holes_number_y=6, p=0.75),

    A.OneOf([
        A.RandomCrop(height=334, width=334, p=0.7),  # 무작위 크롭 (정사각형으로 crop)
        A.RandomCrop(height=384, width=384, p=0.8),  # 무작위 크롭 (정사각형으로 crop)
        A.RandomCrop(height=448, width=448, p=0.9),  # 무작위 크롭 (정사각형으로 crop)
        A.RandomCrop(height=496, width=496, p=1),  # 무작위 크롭 (정사각형으로 crop)
    ], p=0.3),  # OneOf로 랜덤 선택, p=1.0은 항상 적용하도록 함


    A.Resize(*self.resize),
    A.Normalize(),
    ToTensorV2()
])
```

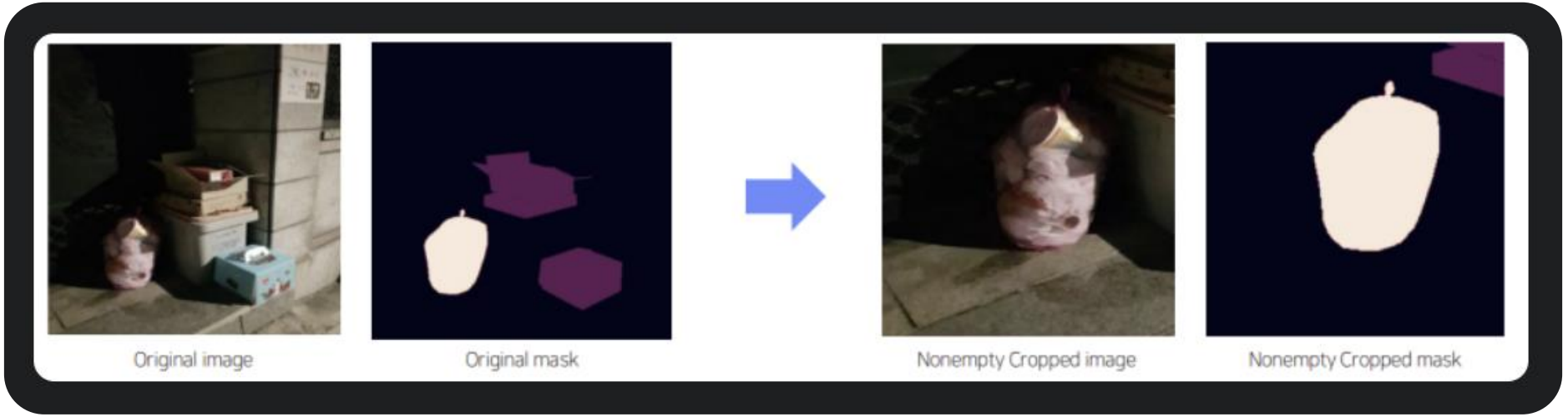
```
A.augmentations.crops.transforms.CropNonEmptyMaskIfExists(height=384,width=384, ignore_values=[0,1,2,3,4,7,8,9],p=1),
```

[mask] cutout argumentation + crop

Model

2-1. CropNonEmptyMaskIfExists

Class 5, Class 6 Crop



Model

3. Loss Function

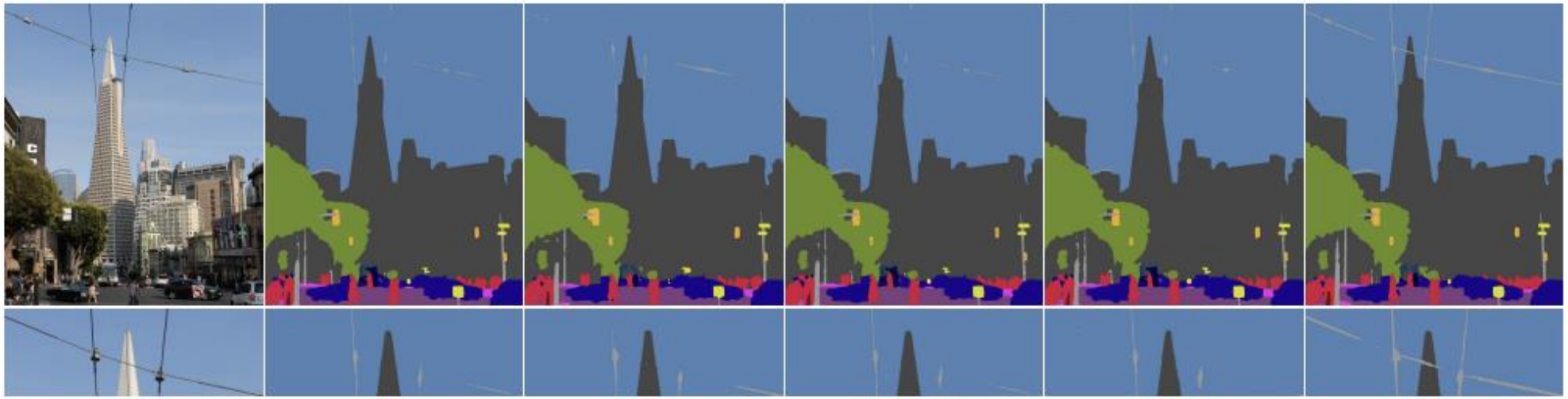


Figure 1: Segmentation results of an internet image. CE: Produced by DeeplabV3 (Chen et al. 2017a) trained with cross-entropy loss on Cityscapes (Cordts et al. 2016) dataset. CIBL: Produced by DeeplabV3 trained with cross-entry loss + lovász-softmax (Berman et al. 2018) + Boundary Loss (Kervadec et al. 2019) on Cityscapes dataset. DenseCRF: Refined results of column 'CE' by DenseCRF (Krähenbühl et al. 2011). Segfix: Refined results of column 'CE' by Segfix (Yuan et al. 2020b). Ours: Re-trained by adding our loss.

➔ 전선을 나타내는 노란색 줄이 마지막 사진에서는 잡아낸다.

Model

3-1. Boundary Loss

```python
class ABL_CE_IOU(nn.Module):
    def __init__(self,weight=None, label_smooth=0.2):
        super(ABL_CE_IOU,self).__init__()
        if weight!=None:
            label_smooth = 0
        self.abl_loss = ABL(weight=weight,label_smoothing=label_smooth)
        self.focal_loss = nn.CrossEntropyLoss()
    def forward(self,logits, targets):
        x = self.abl_loss.forward(logits, targets) +  \
            self.focal_loss.forward(logits, targets) + \
            lovasz_softmax(F.softmax(logits,dim=1),targets)
        return x
```

$$\mathcal{L}_t = \mathbf{CE} + \mathbf{IoU} + w_a\mathbf{ABL},$$

논문 출처 : Wang, C., Zhang, Y., Cui, M., Ren, P., Yang, Y., Xie, X., ⋯ & Xu, W. (2022, June). Active boundary loss for semantic segmentation. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 36, No. 2, pp. 2397-2405).

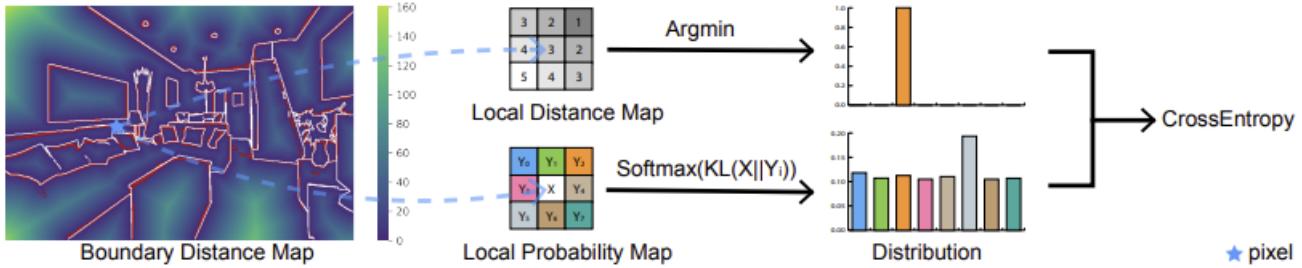


Figure 2: Pipeline of ABL. Boundary distance map is obtained via the distance transform of GTBs, taken an image in the ADE20K (Zhou et al. 2017) dataset as an example. The overlayed white and red lines in the boundary distance map indicate the GTBs and PDBs, respectively. Local distance map: the number indicates the closest distance to the GTBs. Local probability map: $\mathbf{X}$ and $\mathbf{Y}_i, i \in \{0, 1, ..., 7\}$ denote the class probability distribution for these pixels.

Active Boundary Loss를 통한 경계선 인식을 통한 성능 향상 시도

Model

3-2. Focal + Dice Loss

```python
class MixedLoss(nn.Module):
    def __init__(self, alpha=10):
        super().__init__()
        self.alpha = alpha
        self.focal = FocalLoss()

    def forward(self, input, target):
        """

        Args:
            input (_type_): logit [# of batch, class, 512, 512]
            target (_type_): [# of batch, 512, 512]

        Returns:
            _type_: loss
        """
        loss = self.alpha*self.focal(input, target) - torch.log(dice_loss(input, target))
        return loss.mean()
```

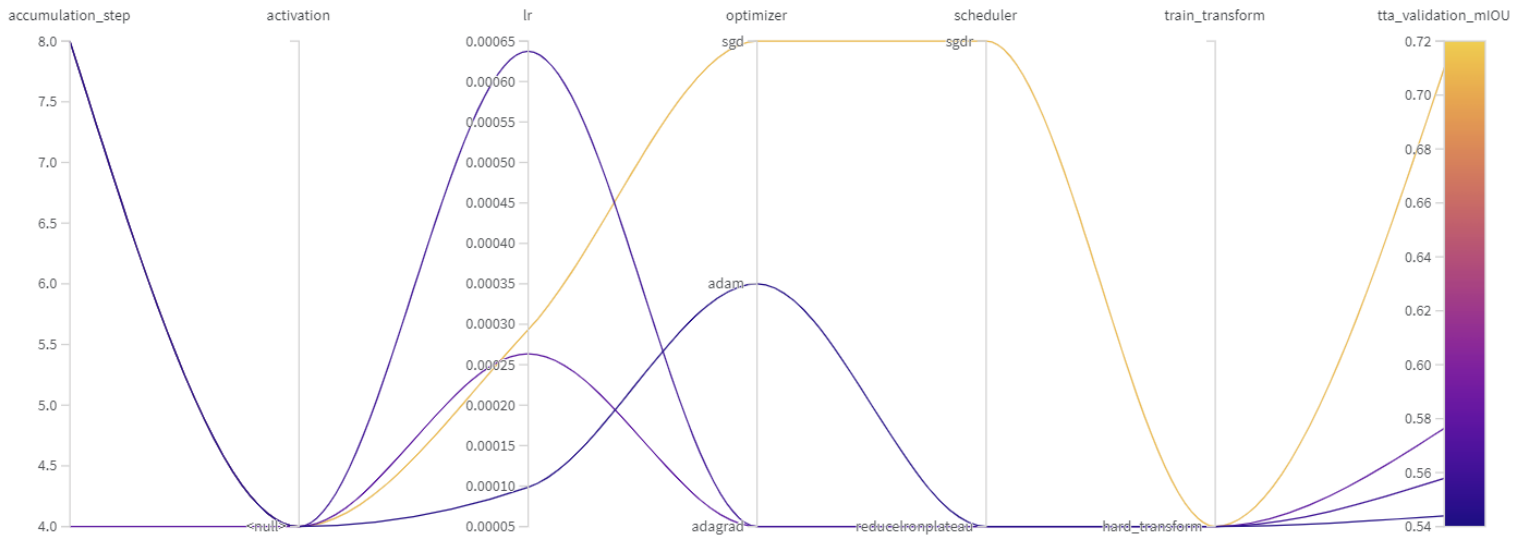
클래스 불균형 문제를 개선하기 위해서

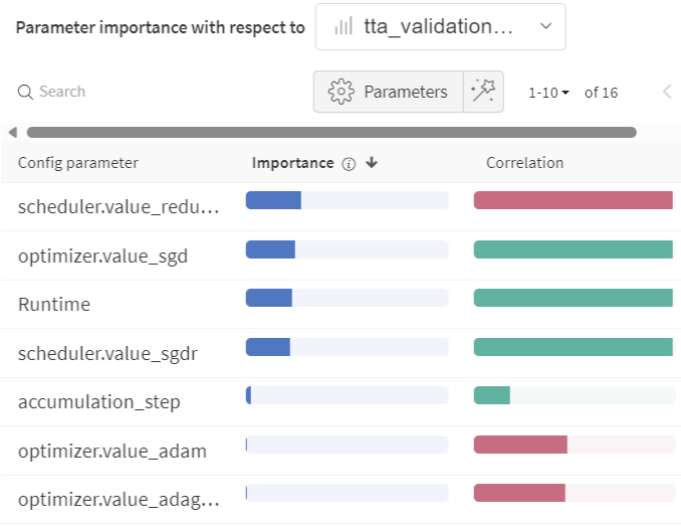
Focal loss와 Dice loss를 사용했다.

Model

4. Parameter Optimization(Bayesian Search)

WandB Sweep을 통한 파라미터 최적화 시도





Model

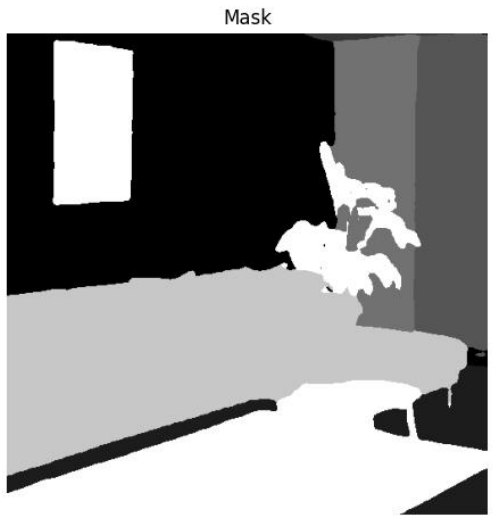
5. Post-Processing TTA

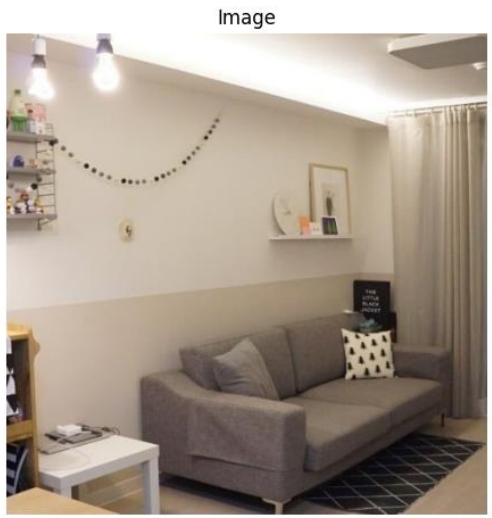
```
transform = tta.Compose(
    [
        tta.HorizontalFlip(),
    ]
)
tta_model = tta.SegmentationTTAWrapper(model, transform)
```

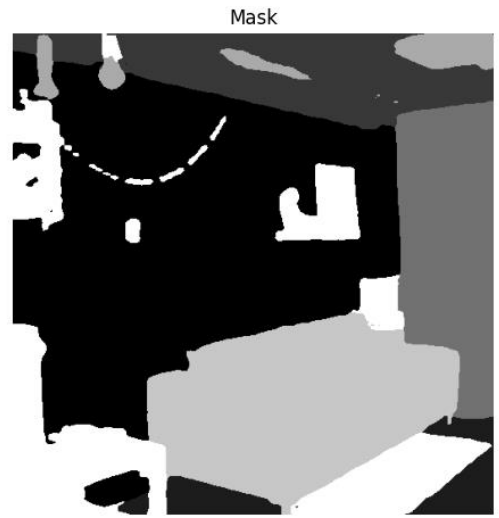
➔ Validation 테스트 결과 성능 mIoU 약 0.003 상승

Conclusion

결론 및 고찰



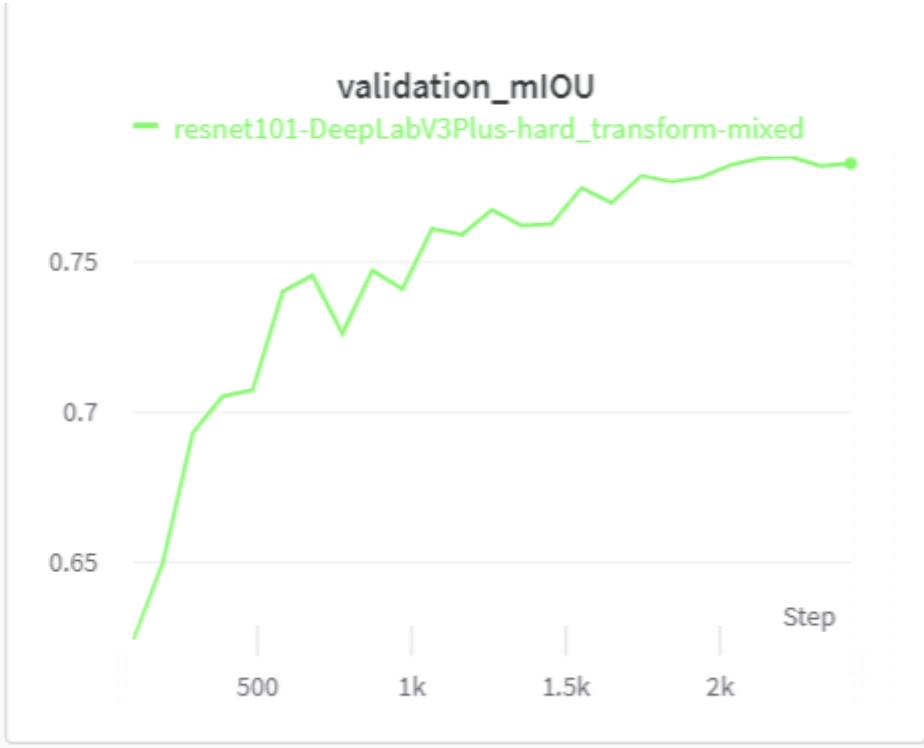




mIoU 성능 : public 0.5986으로 마무리

Conclusion

결론 및 고찰



mIoU 성능 : public 0.5986으로 마무리

Thank You