Revisiting CycleGAN for semi-supervised segmentation

Semi-supervised learning

- Labeled dataset (별로 없음)

$$\mathcal{L} = \{(x_i, y_i)\}_{i=1}^{n}$$

Labeled dataset에 대해 지도학습 기반으로 네트워크 학습

- Unlabeled dataset (많이 있음)

$$\mathcal{U} = \{x'_i\}_{i=1}^{m}$$

unlabeled set도 regularization effect 줄 수 있음

GAN

semi-supervised 관련 unsupervised domain adaptation에서 많이 사용됨

- CycleGAN

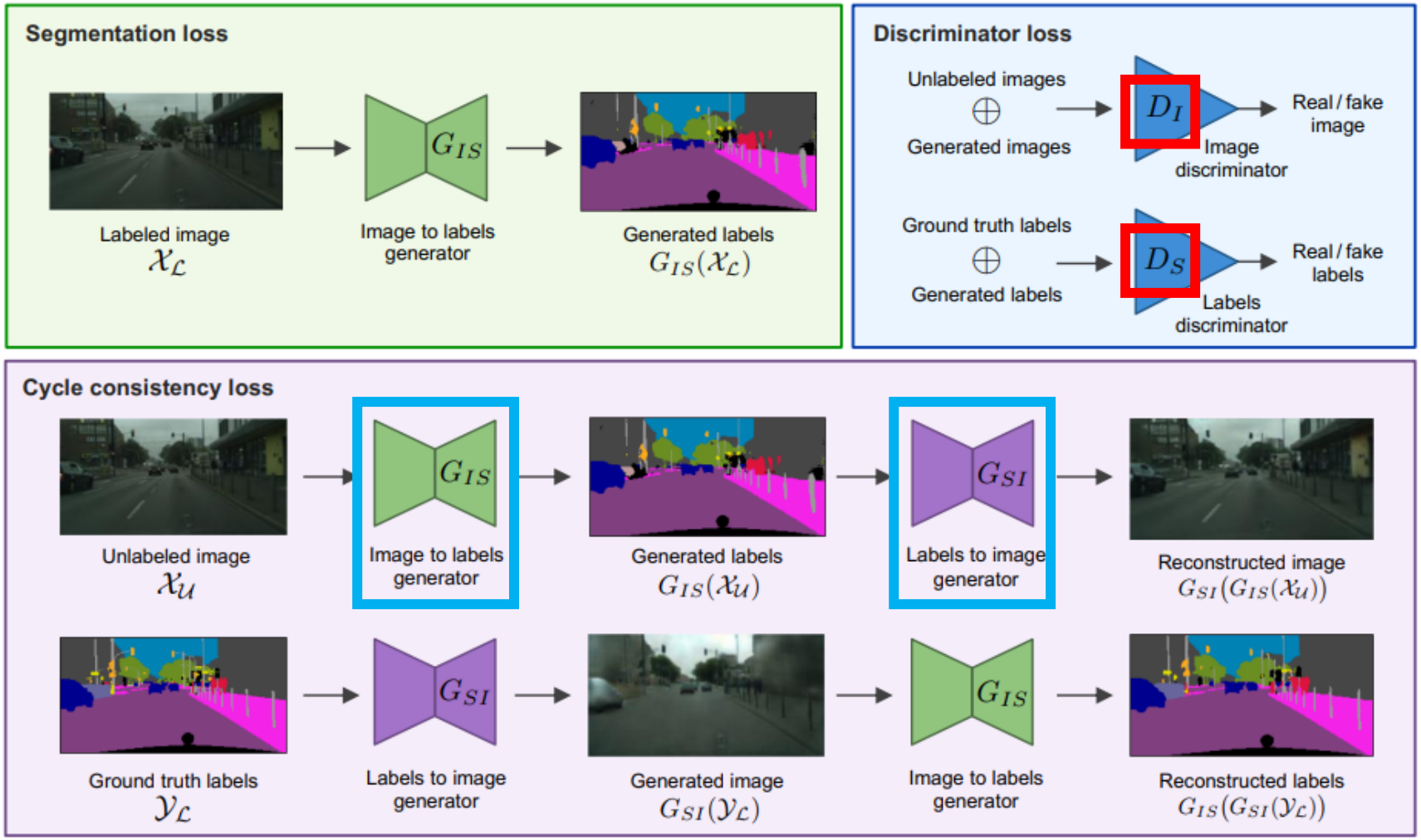
domain 간의 이미지 스타일 transfer를 위해 사용
→ 학습 시 Consistency loss 사용해서 image pair restriction을 없애 주기 때문
→ labeled set과 unlabeled set의 domain이 다르지 않지 때문에 semi-supervised segmentation에서는 많이 사용되지 않았다.

이 논문에서는 CycleGAN의 unpaired domain adaptation ability를 사용해서 "unlabeled real image"에서 GT mask로 그리고 그 반대로 매핑하는 방법을 학습함

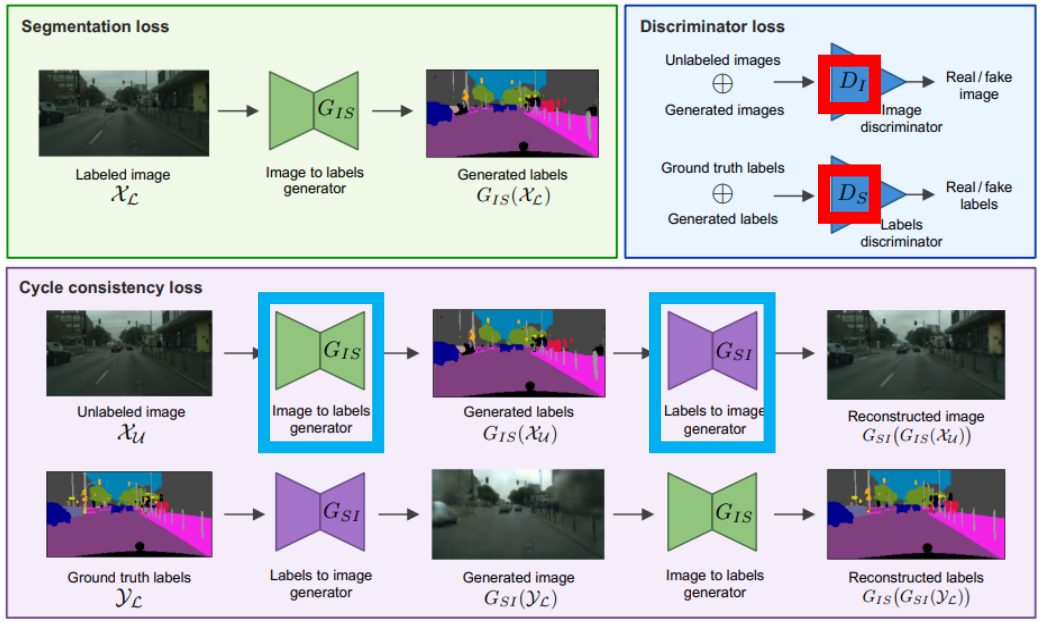
- unsupervised regularization loss로써 작용
- CycleGAN이 domain adaptation(두 domain 간의 shift)이 아니라 segmentation을 위해 사용됨
- GAN이 사용된 적은 있지만 cycle consistency를 처음 사용
  - Unlabeled image와 GT mask 간의 cycle consistent mapping

CycleGAN for semi-supervised learning

두 개의 conditional **generator**와 두 개의 **discriminator**로 구성



CycleGAN for semi-supervised learning



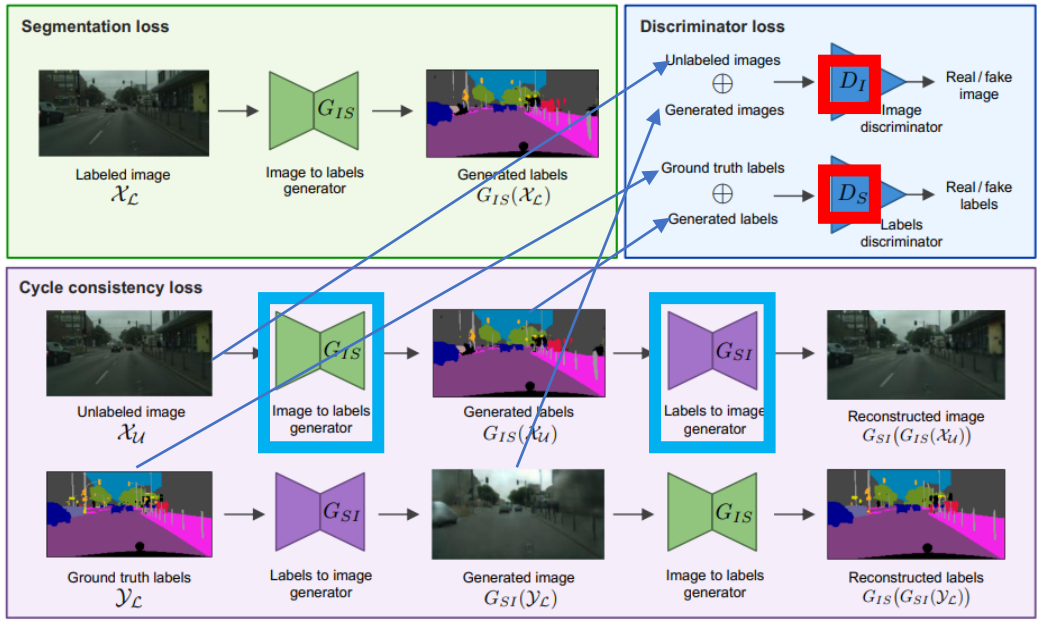
$$L_{\text{total}}(G_{IS}, G_{SI}, D_S, D_I) = L^S_{\text{gen}}(G_{IS}) + \lambda_1 L^I_{\text{gen}}(G_{SI}) + \lambda_2 L^S_{\text{cycle}}(G_{IS}, G_{SI}) + \lambda_3 L^I_{\text{cycle}}(G_{IS}, G_{SI}) - \lambda_4 L^S_{\text{disc}}(G_{IS}, D_S) - \lambda_5 L^I_{\text{disc}}(G_{SI}, D_I) \quad (8)$$

$$\arg\min_{G_{IS}, G_{SI}} \arg\max_{D_S, D_I} L_{\text{total}}(G_{IS}, G_{SI}, D_S, D_I). \quad (9)$$

Segmentation loss(Labeled set 활용 loss)

Unlabeled set 활용한 cycle consistency loss

CycleGAN for semi-supervised learning



$$L_{\text{total}}(G_{IS}, G_{SI}, D_S, D_I) = L^S_{\text{gen}}(G_{IS}) + \lambda_1 L^I_{\text{gen}}(G_{SI}) + \lambda_2 L^S_{\text{cycle}}(G_{IS}, G_{SI}) + \lambda_3 L^I_{\text{cycle}}(G_{IS}, G_{SI}) - \lambda_4 L^S_{\text{disc}}(G_{IS}, D_S) - \lambda_5 L^I_{\text{disc}}(G_{SI}, D_I) \quad (8)$$

$$\arg\min_{G_{IS}, G_{SI}} \arg\max_{D_S, D_I} L_{\text{total}}(G_{IS}, G_{SI}, D_S, D_I). \quad (9)$$

Segmentation loss(Label set 활용 loss)

Unlabeled set 활용한 cycle consistency loss

Unlabeled set 활용한 adversarial loss

Loss function – Labeled set (supervised)

Cross-entropy (Classification)

$$L^S_{\text{gen}}(G_{IS}) = \mathbb{E}_{x,y\sim\mathcal{X}_\mathcal{L},\mathcal{Y}_\mathcal{L}}\left[\mathcal{H}\left(y, G_{IS}(x)\right)\right] \quad (1)$$

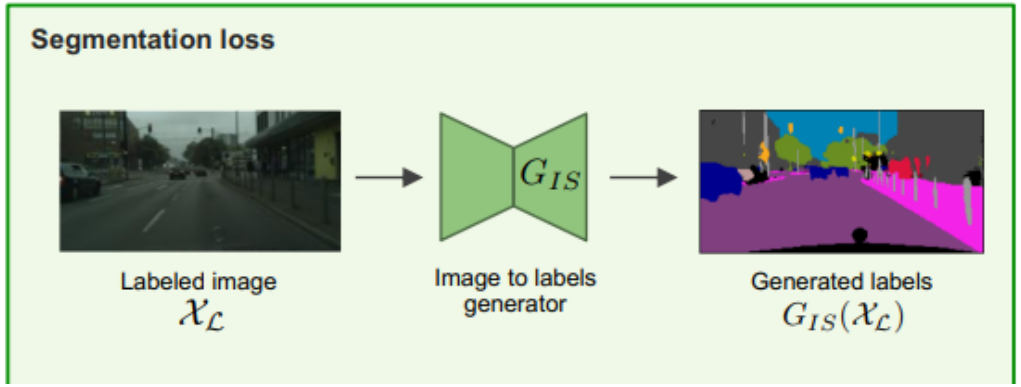
$\mathcal{H}$ is the pixelwise cross-entropy defined as

$$\mathcal{H}(y,\widehat{y}) = -\sum_{j=1}^{N}\sum_{k=1}^{K} y_{j,k}\log\,\widehat{y}_{j,k}. \quad (2)$$

픽셀 j 가 k 일 확률

L2 norm

$$L^I_{\text{gen}}(G_{SI}) = \mathbb{E}_{x,y\sim\mathcal{X}_\mathcal{L},\mathcal{Y}_\mathcal{L}}\left[\,\|G_{SI}(y) - x\|_2^2\,\right]. \quad (3)$$



Loss function – Unlabeled set

1. Adversarial loss

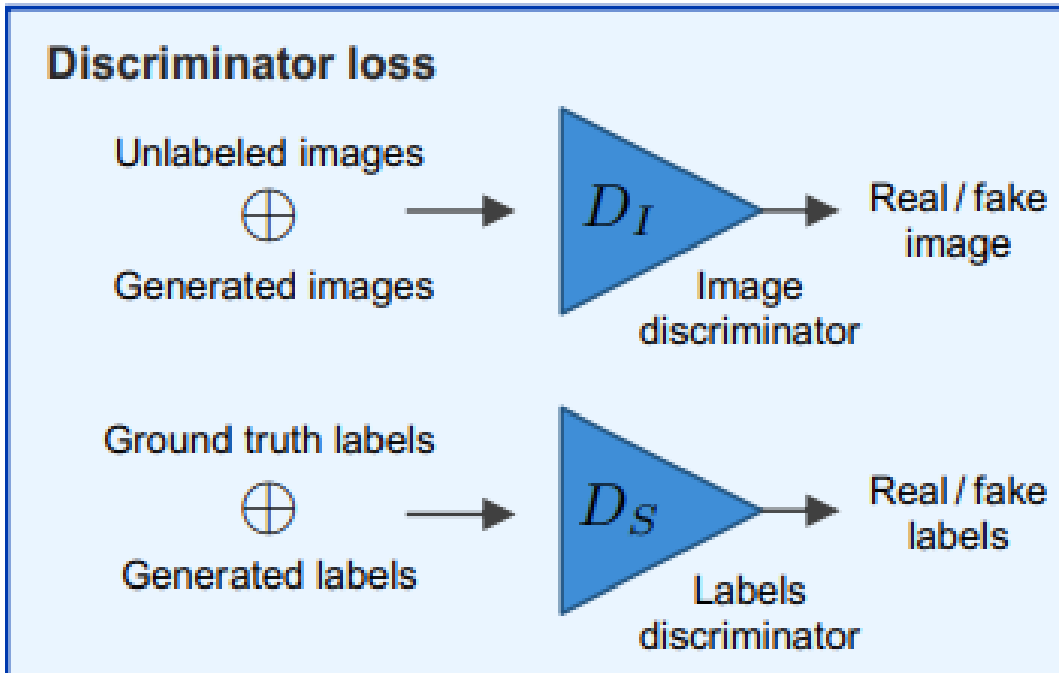
- G, D compete위해 사용
- D를 잘 학습하기 위해 square loss 사용

- Ds(y) : label y가 진짜 mask일 확률
- Adversarial loss for Ds

$$L^S_{\text{disc}}(G_{IS}, D_S) = \mathbb{E}_{y\sim\mathcal{Y}_\mathcal{L}}\left[\left(D_S(y)-1\right)^2\right] + \mathbb{E}_{x'\sim\mathcal{X}_\mathcal{U}}\left[\left(D_S(G_{IS}(x'))\right)^2\right]. \quad (4)$$

- Di(x) : image x가 진짜 이미지일 확률
- Adversarial loss for Di

$$L^I_{\text{disc}}(G_{SI}, D_I) = \mathbb{E}_{x'\sim\mathcal{X}_\mathcal{U}}\left[\left(D_I(x')-1\right)^2\right] + \mathbb{E}_{y\sim\mathcal{Y}_\mathcal{L}}\left[\left(D_I(G_{SI}(y))\right)^2\right]. \quad (5)$$

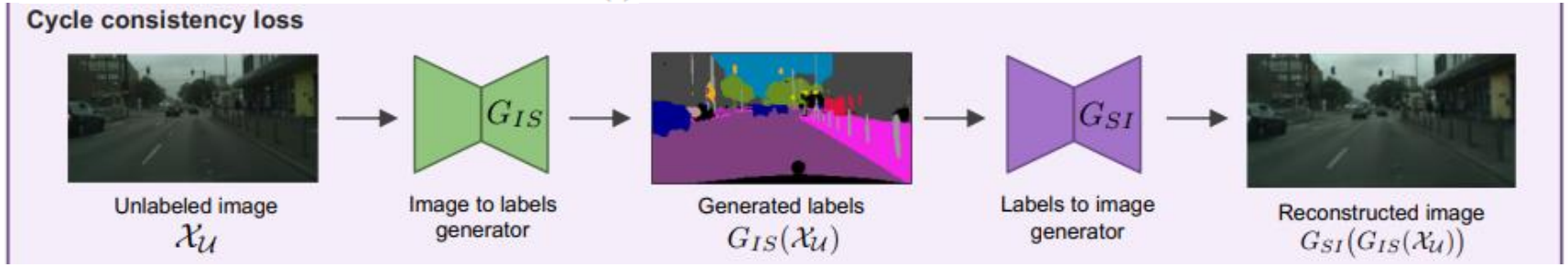


Loss function – Unlabeled set

2. Cycle consistency loss

$$L^{I}_{\text{cycle}}(G_{IS}, G_{SI}) = \mathbb{E}_{x' \sim \mathcal{X}_{\mathcal{U}}} \left[\| G_{SI}(G_{IS}(x')) - x' \|_1 \right]. \quad (6)$$

Sharper image를 위해 L1 norm을 사용



$$L^{S}_{\text{cycle}}(G_{IS}, G_{SI}) = \mathbb{E}_{y \sim \mathcal{Y}_{\mathcal{L}}} \left[\mathcal{H}(y, G_{IS}(G_{SI}(y))) \right]. \quad (7)$$

분류 문제니까 cross-entropy 사용

Implementation detail

1. Generator

transfer. This network is composed of two stride-2 convolutions, followed by 9 residual blocks and two fractionally-strided convolutions with stride $1/2$. Similarly, instance normalization [30] was employed and no drop-out was adopted. Furthermore, we used softmax as output function when generating segmentation labels from images, whereas *tanh* was the selected function when translating from segmentation labels to images, in order to have continuous values. In pre-processing, each channel of an image is nor-

```
class ResnetGenerator(nn.Module):
    def __init__(self, input_nc=3, output_nc=3, ngf=64, norm_layer=nn.BatchNorm2d, use_dropout=True, num_blocks=6, softmax=False):
        super(ResnetGenerator, self).__init__()
        if type(norm_layer) == functools.partial:
            use_bias = norm_layer.func == nn.InstanceNorm2d
        else:
            use_bias = norm_layer == nn.InstanceNorm2d

        res_model = [nn.ReflectionPad2d(3),
                     conv_norm_relu(input_nc, ngf * 1, 7, norm_layer=norm_layer, bias=use_bias),
                     conv_norm_relu(ngf * 1, ngf * 2, 3, 2, 1, norm_layer=norm_layer, bias=use_bias),
                     conv_norm_relu(ngf * 2, ngf * 4, 3, 2, 1, norm_layer=norm_layer, bias=use_bias)]

        for i in range(num_blocks):
            res_model += [ResidualBlock(ngf * 4, norm_layer, use_dropout, use_bias)]

        if softmax:
            res_model += [dconv_norm_relu(ngf * 4, ngf * 2, 3, 2, 1, 1, norm_layer=norm_layer, bias=use_bias),
                          dconv_norm_relu(ngf * 2, ngf * 1, 3, 2, 1, 1, norm_layer=norm_layer, bias=use_bias),
                          nn.ReflectionPad2d(3),
                          nn.Conv2d(ngf, output_nc, 7)]
        else:
            res_model += [dconv_norm_relu(ngf * 4, ngf * 2, 3, 2, 1, 1, norm_layer=norm_layer, bias=use_bias),
                          dconv_norm_relu(ngf * 2, ngf * 1, 3, 2, 1, 1, norm_layer=norm_layer, bias=use_bias),
                          nn.ReflectionPad2d(3),
                          nn.Conv2d(ngf, output_nc, 7),
                          nn.Tanh()]
        self.res_model = nn.Sequential(*res_model)

    def forward(self, x):
        return self.res_model(x)
```

- False로 줌

norm_layer = functools.partial(nn.BatchNorm2d, affine=True)
norm_layer = functools.partial(nn.InstanceNorm2d, affine=False, track_running_stats=False)

Implementation detail

2. Discriminator

Unlike the original CycleGAN model, we make use of pixel-wise discriminators [11] where the size of the output is the same as the input and the adversarial label (i.e., real / generated) is recopied at each output pixel. We found this model to perform better than having a single discriminator output. Each discriminator contains three convolutional blocks, followed by Leaky ReLU activations with negative slope of $\alpha = 0.2$. In addition, batch normalization is used in the discriminators after the second convolutional block.

```
class PixelDiscriminator(nn.Module):
    def __init__(self, input_nc, ndf=64, norm_layer=nn.BatchNorm2d, use_bias=False):
        super(PixelDiscriminator, self).__init__()
        dis_model = [
            nn.Conv2d(input_nc, ndf, kernel_size=1, stride=1, padding=0),
            nn.LeakyReLU(0.2, True),
            nn.Conv2d(ndf, ndf * 2, kernel_size=1, stride=1, padding=0, bias=use_bias),
            norm_layer(ndf * 2),
            nn.LeakyReLU(0.2, True),
            nn.Conv2d(ndf * 2, 1, kernel_size=1, stride=1, padding=0, bias=use_bias)]

        self.dis_model = nn.Sequential(*dis_model)

    def forward(self, input):
        return self.dis_model(input)
```

Experiments

PASCAL VOC 2012 – object / 200x200 pix로 resize하고 feed 함
Cityscapes – scene / 128x256 pix로 resize
ACDC – medical

- Fully supervised 방식으로 학습해서 Upper bound performance를 구함
- 레이블이 있는 이미지의 10%, 20%, 30%, 50%를 사용해서 학습한 baseline을 Partial이라고 부름
- 같은 subset으로 semi-supervised 방식으로 학습됨
- 이전 SOTA와 비교 (Adversarial learning for semi-supervised semantic segmentation)

3개 데이터셋

Method	Labeled %	VOC	Cityscapes	ACDC
Full	100	0.5543	0.5551	0.8982
Hung et al. [11]	20	0.2032	0.3490	0.8063
Partial	50	0.4108	0.4856	0.8863
	30	0.3237	0.4502	0.8785
	20	0.2688	0.4112	0.8642
	10	0.2158	0.3636	0.8418
Ours	50	0.4197	0.4997	0.8890
	30	0.3514	0.4654	0.8804
	20	0.2981	0.4321	0.8688
	10	0.2543	0.3923	0.8463

Upper Bound -> full보단 많이 떨어짐

이전 SOTA 방식 <- Partial보다 별로
(논문에서랑 달리 pretrain 없이)

일부만 갖고 supervised 방식으로 학습 (Partial)
-> 이것보단 조금씩 향상
-> 사용되는 데이터가 적을수록 더 많이 향상

일부만 갖고 우리 semi-supervised 방식으로 학습

Experiments

정확도는 낮아도 global semantic 정보, 디테일은 잘 캡쳐하는 것 같다고 주장

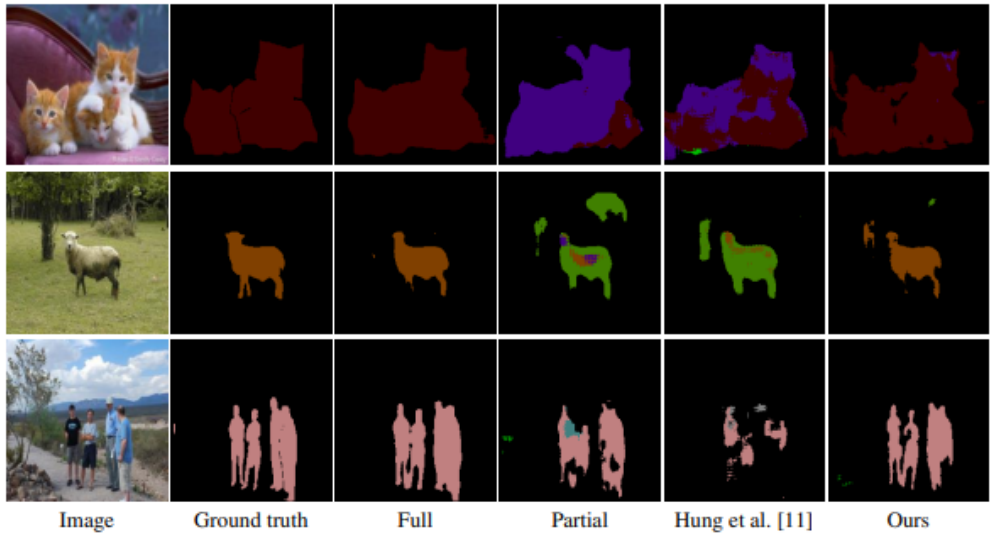


Figure 3: Visual comparisons on the PASCAL VOC 2012 dataset employing 20% of labeled images for training.

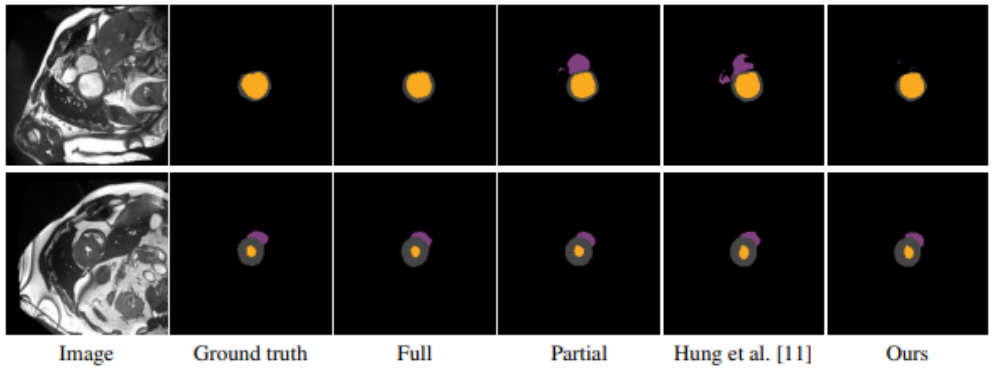


Figure 5: Visual comparisons on the ACDC dataset employing 20% of labeled images for training.

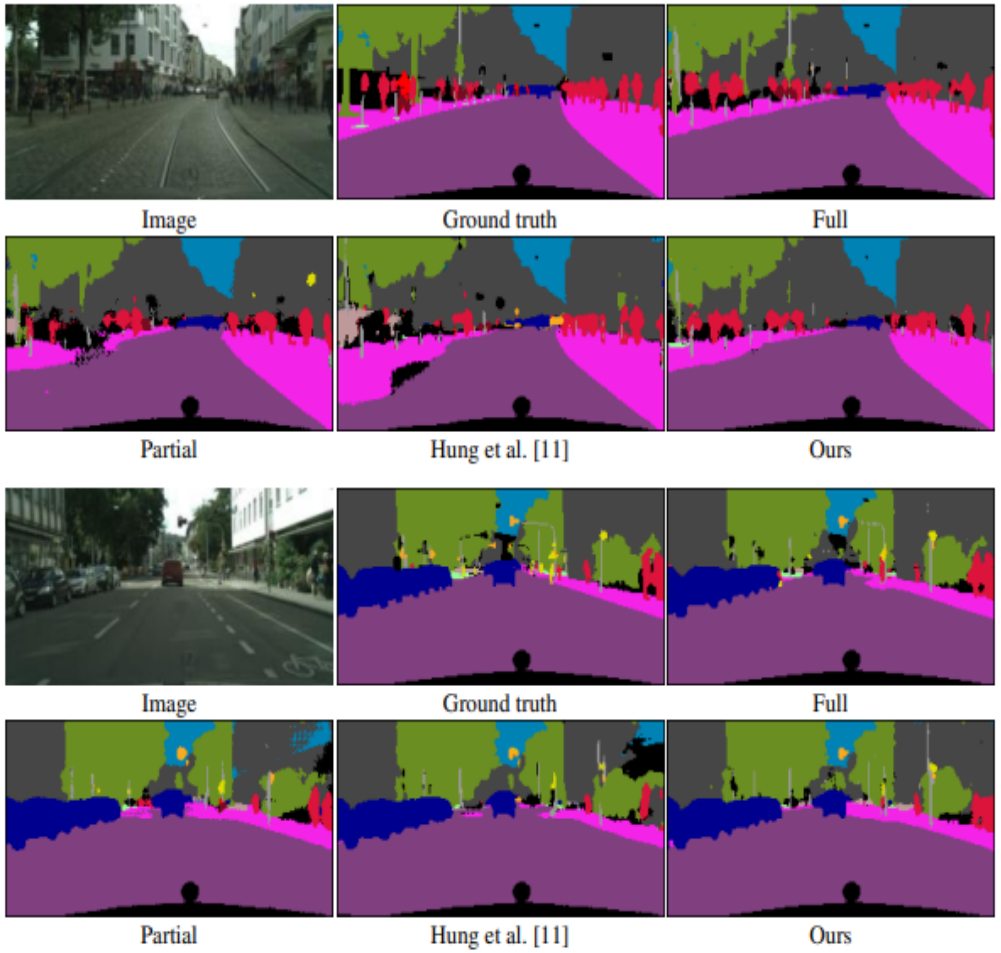


Figure 4: Visual comparisons on the Cityscape dataset employing 20% of labeled images for training.

Experiments

Method	VOC
Proposed	0.2981
w/o labels cycle loss (L^S_{cycle})	0.2627
w/o image cycle loss (L^I_{cycle})	0.2733
w/o labels discr. loss (L^S_{disc})	0.2614
w/o image discr. loss (L^I_{disc})	0.2543

Table 2: Ablation study on the PASCAL VOC 2012 dataset with 20% labeled data.

원래 loss function에서의 mIOU

Cycle consistency loss on seg mask가 더 중요

반대로 Image Discriminator 가 더 중요

Mask를 이미지로 바꾸고 다시 mask로 바꿨을 때 같아지느냐가 더 중요

Y를 갖고 생성시킨 이미지냐 / real unlabeled 이미지이냐 판단이 더 중요