Semantic Image Synthesis
With Spatially-Adaptive Normalization

한양대학교 AILAB 석사과정 엄희송

AUTHORS

CVPR 2019 oral paper

Taesung Park   Ming-Yu Liu   Ting-Chun Wang   Jun-Yan Zhu

*Taesung Park contributed
to the work during his NVIDIA internship.

Taesung Park

FOLLOW

UC Berkeley
Verified email at berkeley.edu - Homepage
Computer Vision

TITLE	CITED BY	YEAR
Unpaired image-to-image translation using cycle-consistent adversarial networks JY Zhu, T Park, P Isola, AA Efros Proceedings of the IEEE international conference on computer vision, 2223-2232	1966	2017
Cycada: Cycle-consistent adversarial domain adaptation J Hoffman, E Tzeng, T Park, JY Zhu, P Isola, K Saenko, AA Efros, T Darrell arXiv preprint arXiv:1711.03213	220	2017
Inverse optimal control for humanoid locomotion T Park, S Levine Robotics Science and Systems Workshop on Inverse Optimal Control and Robotic ...	20	2013
Semantic Image Synthesis with Spatially-Adaptive Normalization T Park, MY Liu, TC Wang, JY Zhu arXiv preprint arXiv:1903.07291	8	2019

1. INTRODUCTION

...

We are interested in a specific form of conditional image synthesis,
which is converting a semantic segmentation mask to a photorealistic image.

...

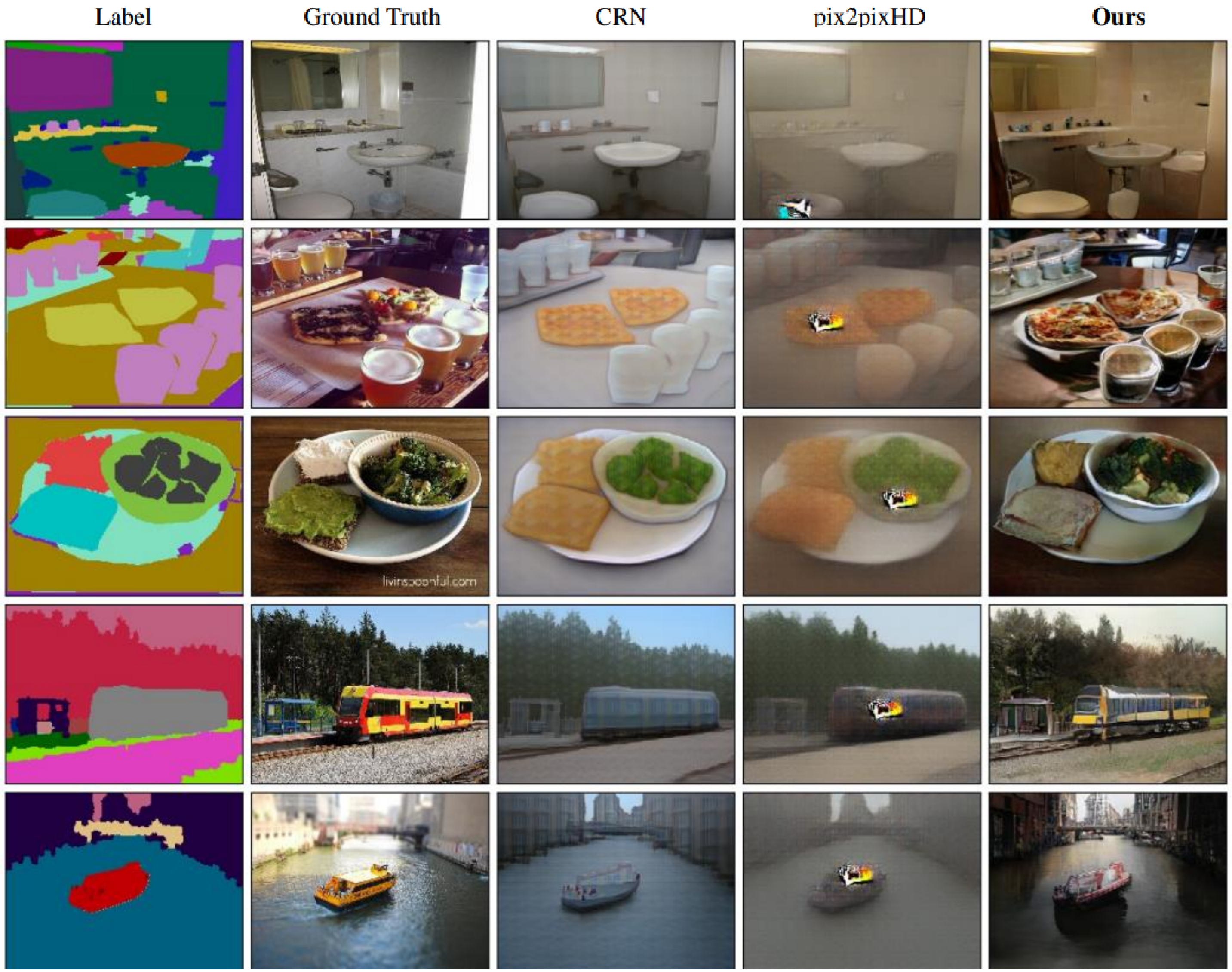
In this paper, we show that the conventional network architecture [20,40],
which is built by stacking convolutional, normalization, and nonlinearity layers, is at best sub-optimal,
because their normalization layers tend to "wash away" information in input semantic masks.

To address the issue, we propose spatially-adaptive normalization, a conditional normalization layer
that modulates the activations using input semantic layouts through a spatially adaptive, learned
transformation and can effectively propagate the semantic information throughout the network.

...


Label
Ground Truth
CRN
pix2pixHD
Ours

1. INTRODUCTION



2. RELATED WORK

Deep generative models – GAN, VAE 등이 있다. 이 논문에서는 GAN을 이용함.

Conditional image synthesis – 다양한 input data (category label, text, images) 에 대한 이미지 합성이 연구됨.
이 논문에서는 image-to-image translation 에 집중함.

Unconditional normalization layers – Batch Normalization, Instance Normalization, Layer Normalization 등…

Conditional normalization layers – Conditional Batch Normalization, Adaptive Instance Normalization 등…
먼저 layer activation 이 normalized 되고, 이 값이 external data를 반영하도록 affine transform 됨.
기존의 conditional normalization layer 들은 spatial conditions 에 대해서는 동일하게 normalized 되기 때문에 문제가 생김. 이 논문의 normalization layer 는 spatially varying affine transformation 을 적용함.

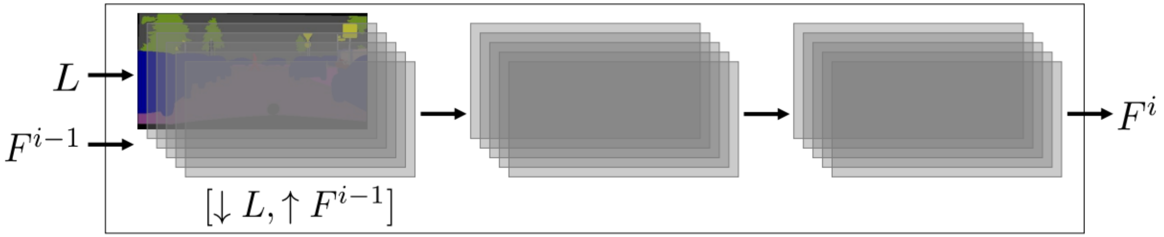


Figure 3. A single refinement module.

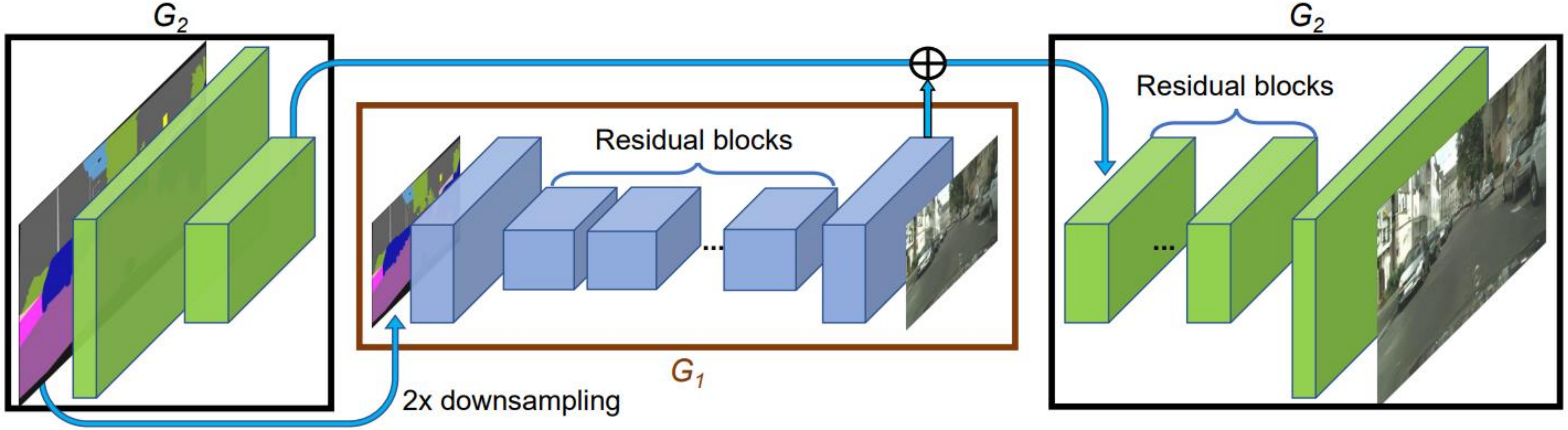


Figure 3: Network architecture of our generator. We first train a residual network G_1 on lower resolution images. Then, another residual network G_2 is appended to G_1 and the two networks are trained jointly on high resolution images. Specifically, the input to the residual blocks in G_2 is the element-wise sum of the feature map from G_2 and the last feature map from G_1.

Upper image: Photographic Image Synthesis with Cascaded Refinement Networks. Figure 3.
Lower image: High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs. Figure 3.

2. RELATED WORK – UNCONDITIONAL NORMALIZATION LAYERS

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

Batch Normalization: Accelerating Deep Network Training b y Reducing Internal Covariate Shift. Algorithm 1.

On the other hand, the generator network of Ulyanov et al. (2016) does contain a normalization layers, and precisely *batch normalization* ones. The key difference between eq. (1) and batch normalization is that the latter applies the normalization to a whole batch of images instead for single ones:

$$y_{tijk} = \frac{x_{tijk} - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}}, \quad \mu_i = \frac{1}{HWT}\sum_{t=1}^{T}\sum_{l=1}^{W}\sum_{m=1}^{H} x_{tilm}, \quad \sigma_i^2 = \frac{1}{HWT}\sum_{t=1}^{T}\sum_{l=1}^{W}\sum_{m=1}^{H}(x_{tilm} - mu_i)^2. \tag{2}$$

In order to combine the effects of instance-specific normalization and batch normalization, we propose to replace the latter by the *instance normalization* (also known as "contrast normalization") layer:

$$y_{tijk} = \frac{x_{tijk} - \mu_{ti}}{\sqrt{\sigma_{ti}^2 + \epsilon}}, \quad \mu_{ti} = \frac{1}{HW}\sum_{l=1}^{W}\sum_{m=1}^{H} x_{tilm}, \quad \sigma_{ti}^2 = \frac{1}{HW}\sum_{l=1}^{W}\sum_{m=1}^{H}(x_{tilm} - mu_{ti})^2. \tag{3}$$

We replace batch normalization with instance normalization everywhere in the generator network g. This prevents instance-specific mean and covariance shift simplifying the learning process. Differently from batch normalization, furthermore, the instance normalization layer is applied at test time as well.

Instance Normalization: The Missing Ingredient for Fast Stylization. Equation 2, 3.

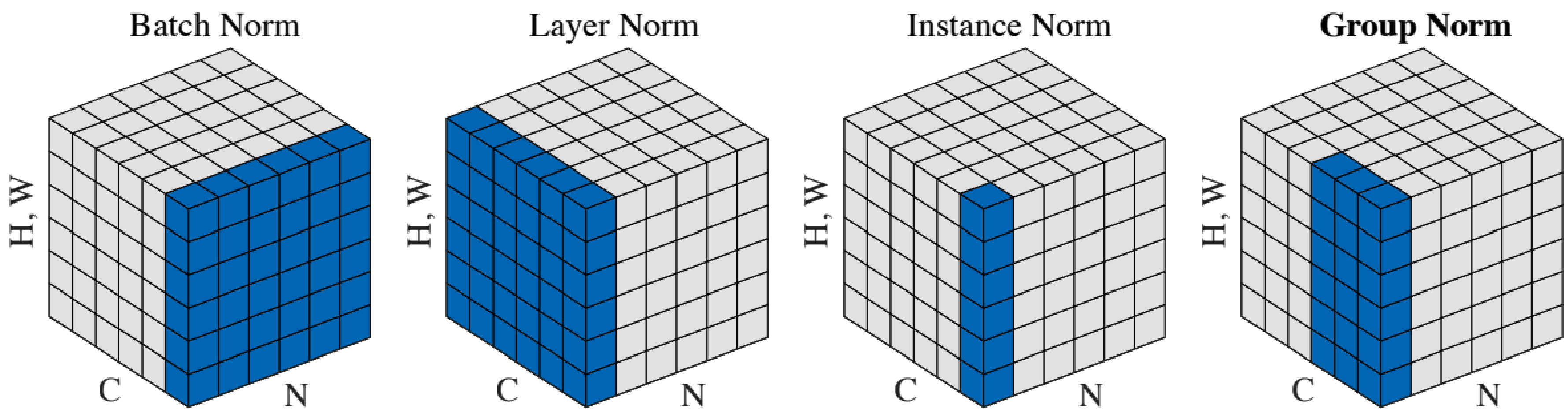


Figure 2. **Normalization methods**. Each subplot shows a feature map tensor, with N as the batch axis, C as the channel axis, and (H, W) as the spatial axes. The pixels in blue are normalized by the same mean and variance, computed by aggregating the values of these pixels.

Wu, Y. and He, K., 2018. Group normalization. Figure 2.

$$\mathrm{AdaIN}(x, y) = \sigma(y)\left(\frac{x - \mu(x)}{\sigma(x)}\right) + \mu(y) \qquad (8)$$

Normalization 은 input x의 평균과 분산을 이용, affine transformation 은 output y의 평균과 분산을 이용하는 방법
AdaIN의 경우 learnable parameter 없이 external data (y)를 참조한 계산 값을 사용하기 때문에 속도에서 장점

Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization, Equation 8.

SPatially-Adaptive (DE)normalization (SPADE)

$$\gamma^i_{c,y,x}(\mathbf{m})\frac{h^i_{n,c,y,x}-\mu^i_c}{\sigma^i_c}+\beta^i_{c,y,x}(\mathbf{m}) \tag{1}$$

where $h^i_{n,c,y,x}$ is the activation at the site before normalization, μ^i_c and σ^i_c are the mean and standard deviation of the activation in channel c:

$$\mu^i_c=\frac{1}{NH^iW^i}\sum_{n,y,x}h^i_{n,c,y,x} \tag{2}$$

$$\sigma^i_c=\sqrt{\frac{1}{NH^iW^i}\sum_{n,y,x}(h^i_{n,c,y,x})^2-(\mu^i_c)^2}. \tag{3}$$

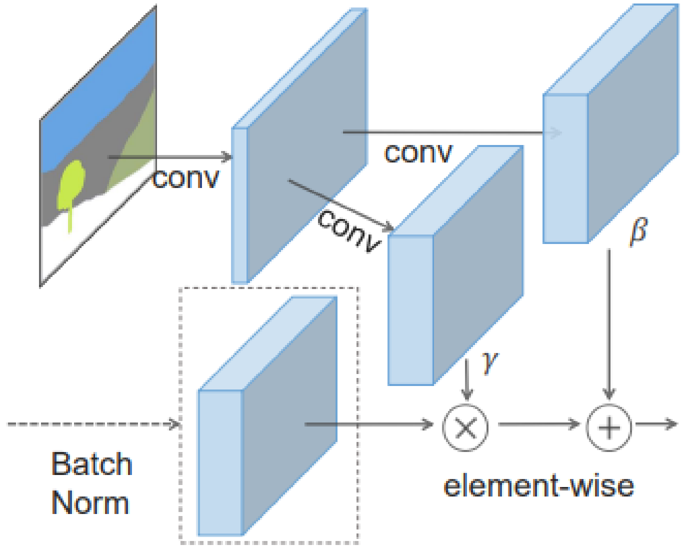


Figure 2: In SPADE, the mask is first projected onto an embedding space, and then convolved to produce the modulation parameters γ and β. Unlike prior conditional normalization methods, γ and β are not vectors, but tensors with spatial dimensions. The produced γ and β are multiplied and added to the normalized activation element-wise.

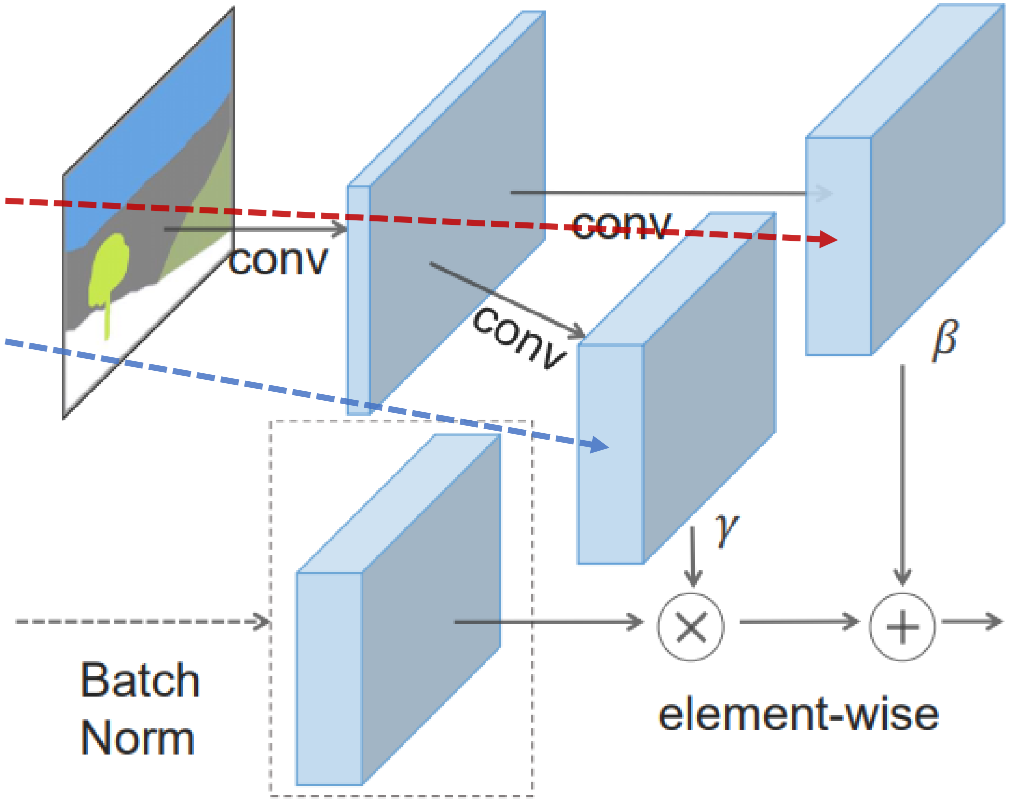
SPatially-Adaptive (DE)normalization (SPADE)

$$\gamma^i_{c,y,x}(\mathbf{m}) \frac{h^i_{n,c,y,x} - \mu^i_c}{\sigma^i_c} + \beta^i_{c,y,x}(\mathbf{m}) \quad (1)$$

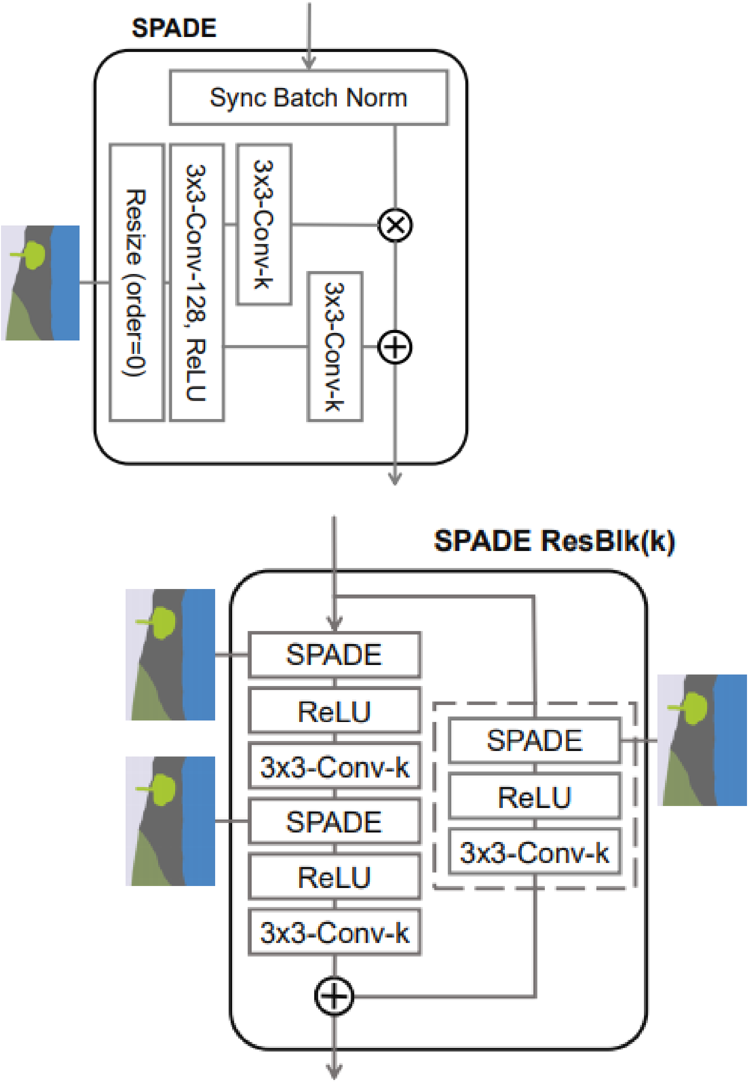
where $h^i_{n,c,y,x}$ is the activation at the site before normalization, μ^i_c and σ^i_c are the mean and standard deviation of the activation in channel c:

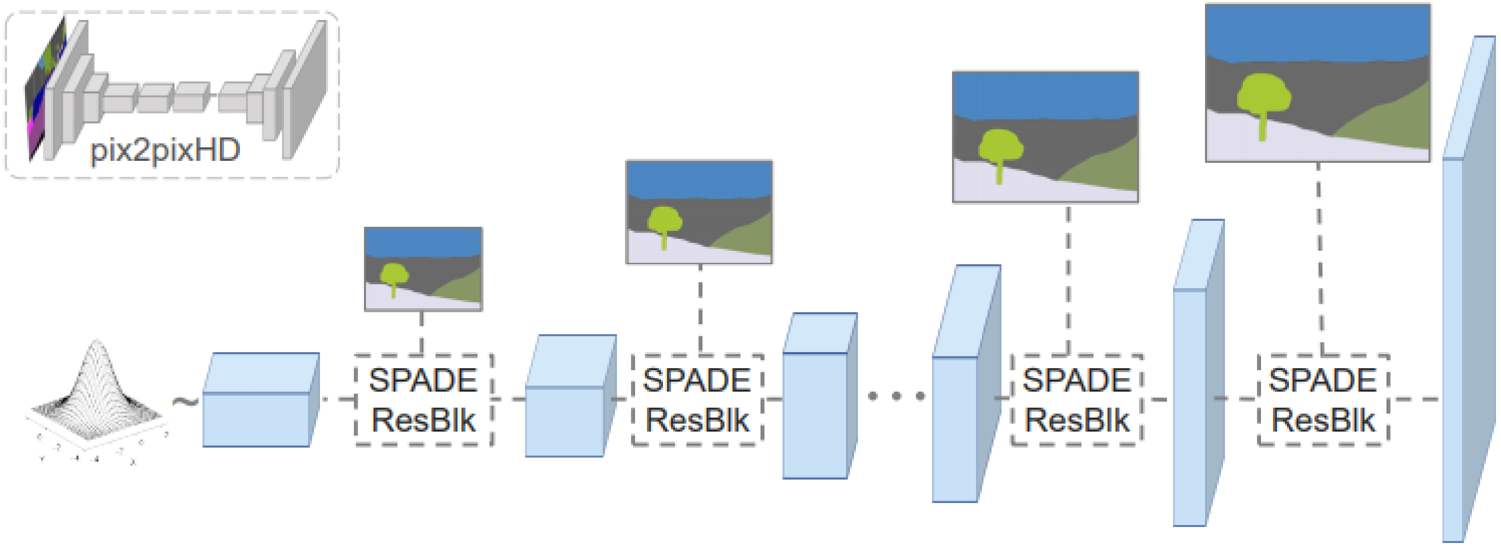
$$\mu^i_c = \frac{1}{NH^iW^i} \sum_{n,y,x} h^i_{n,c,y,x} \quad (2)$$

$$\sigma^i_c = \sqrt{\frac{1}{NH^iW^i} \sum_{n,y,x} (h^i_{n,c,y,x})^2 - (\mu^i_c)^2}. \quad (3)$$

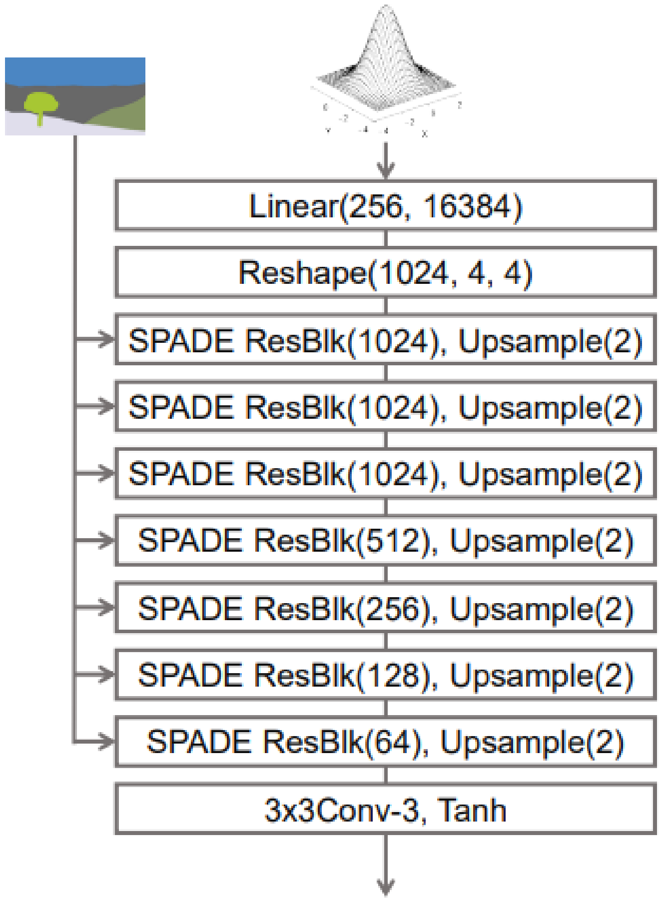


3. SEMANTIC IMAGE SYNTHESIS

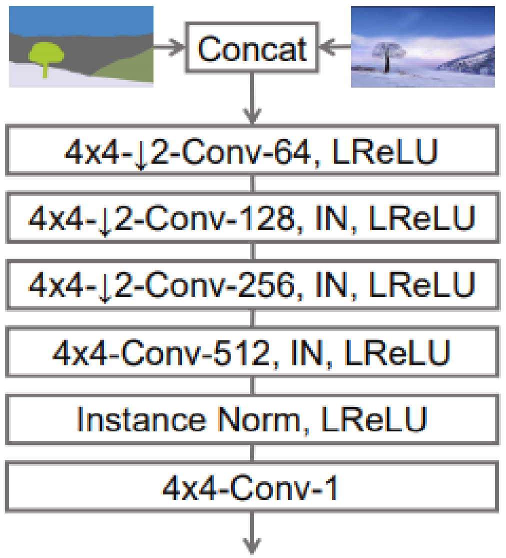




기존의 합성 모델들과 다르게 이 논문의 모델은 input을 standard multivariate Gaussian distribution 으로 받는 decoder 형식의 generator 를 사용함.
Segmentation mask 의 정보가 SPADE layer의 external data 로 들어감.



SPADE generator



Discriminator

Why does SPADE work better?

논문에서는 서술식으로 SPADE의 효과를 설명

기존 synthesis 방법에서는 segmentation mask 가 input,
이 정보가 normalization layer 를 거치면서 희석됨.

Ex) 이미지 전체가 one label 로 되어 있는 경우,

[sky…sky] 이미지와 [grass…grass] 이미지는
normalization layer를 거치면서 그 값이 모두 0이 됨.
Label은 다르지만 output이 같음.

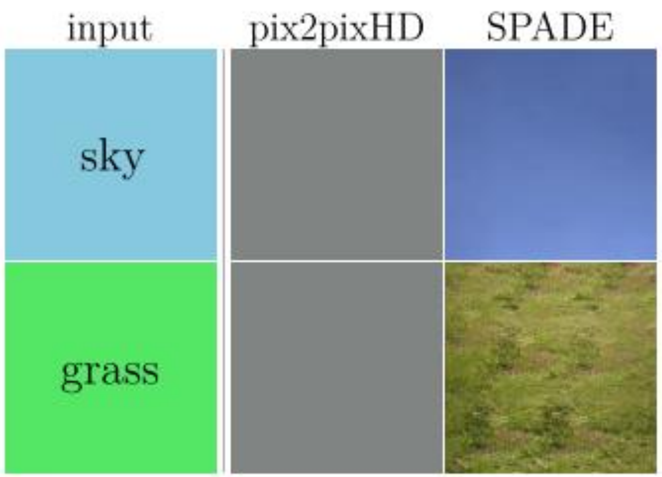


Figure 3: Comparing results given uniform segmentation maps: while SPADE generator produces plausible textures, pix2pixHD [40] produces identical outputs due to the loss of the semantic information after the normalization layer.

우리 모델은 random vector를 normalization layer에 거치고,
segmentation 정보는 그대로 사용할 수 있도록 learnable parameters 로 학습.

Training details

pix2pixHD 와 기본적으로 동일(Adversarial loss + feature matching loss),
Adversarial loss 에 사용된 LSGAN loss 만 Hinge loss로 변경

G와 D의 전체 layer weights에 대해 Spectral Normalization 적용

Lr_G = 0.0001, Lr_D = 0.0004 with ADAM optimizer(b1=0, b2=0.999)

Feature matching loss of pix2pixHD

Discriminator 의 각 convolutional layer 에서 생성되는 feature map 으로 비교
(feature map of real image – feature map of fake image)

$$\mathcal{L}_{\text{FM}}(G, D_k) = \mathbb{E}_{(\mathbf{s},\mathbf{x})} \sum_{i=1}^{T} \frac{1}{N_i} [||D_k^{(i)}(\mathbf{s}, \mathbf{x}) - D_k^{(i)}(\mathbf{s}, G(\mathbf{s}))||_1], \tag{4}$$

High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs, Equation 4.

Hinge loss of SPADE

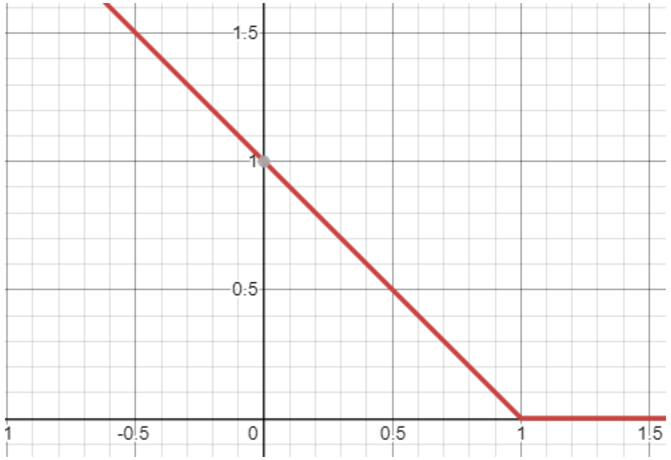
D가 판별한 값을 input으로 받아서 loss로 계산

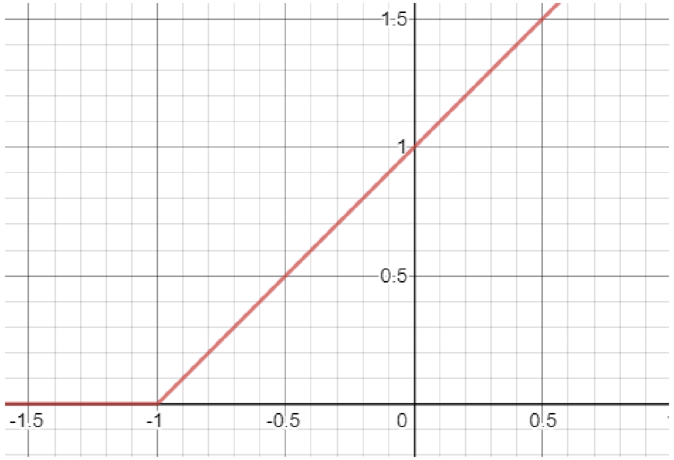
```
elif self.gan_mode == 'hinge':
    if for_discriminator:
        if target_is_real:
            minval = torch.min(input - 1, self.get_zero_tensor(input))
            loss = -torch.mean(minval)
        else:
            minval = torch.min(-input - 1, self.get_zero_tensor(input))
            loss = -torch.mean(minval)
    else:
        assert target_is_real, "The generator's hinge loss must be aiming for real"
        loss = -torch.mean(input)
    return loss
```

https://github.com/NVlabs/SPADE/blob/master/models/networks/loss.py

Hinge loss of SPADE - D의 경우

Real image: -(min(D(real) – 1, 0)) 의 평균값을 minimize = (min(D(real) – 1, 0)) 의 평균값을 maximize.
D(real) 값이 1에 가까워 지려고 함.
Fake(Generated): -(min (-D(fake) – 1, 0)) 의 평균값을 minimize = (min (-D(fake) – 1, 0)) 의 평균값을 maximize.
D (fake) 값이 -1에 가까워 지려고 함.



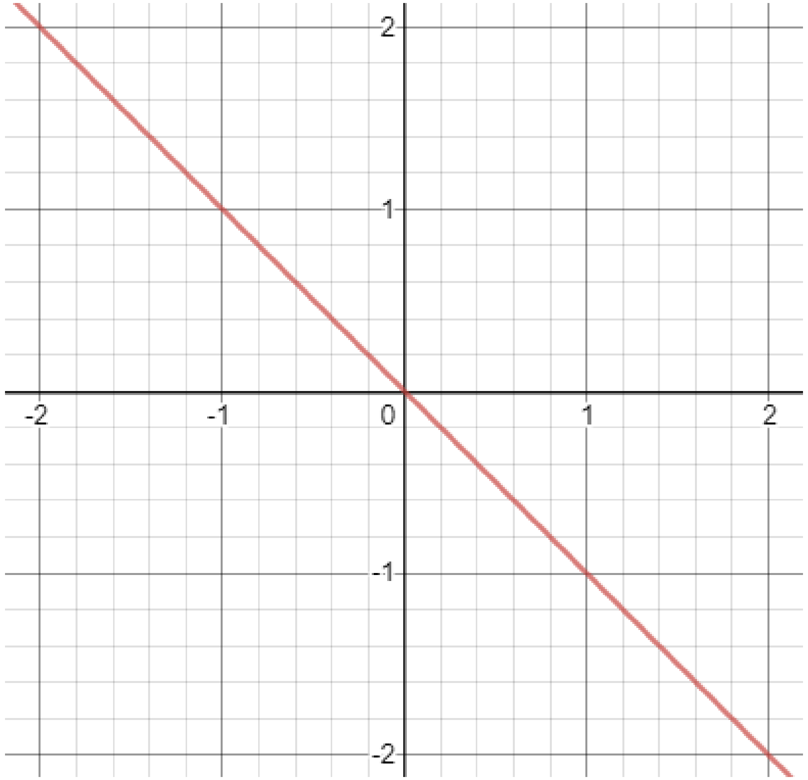



```
elif self.gan_mode == 'hinge':
    if for_discriminator:
        if target_is_real:
            minval = torch.min(input - 1, self.get_zero_tensor(input))
            loss = -torch.mean(minval)
        else:
            minval = torch.min(-input - 1, self.get_zero_tensor(input))
            loss = -torch.mean(minval)
    else:
        assert target_is_real, "The generator's hinge loss must be aiming for real"
        loss = -torch.mean(input)
    return loss
```

https://github.com/NVlabs/SPADE/blob/master/models/networks/loss.py

Hinge loss of SPADE – G의 경우

Fake(Generated) image 에 대해 D가 판별하는 input의 값이 무조건 최대가 되도록
Loss = - mean(input) ➔ maximizing mean(input)




```python
elif self.gan_mode == 'hinge':
    if for_discriminator:
        if target_is_real:
            minval = torch.min(input - 1, self.get_zero_tensor(input))
            loss = -torch.mean(minval)
        else:
            minval = torch.min(-input - 1, self.get_zero_tensor(input))
            loss = -torch.mean(minval)
    else:
        assert target_is_real, "The generator's hinge loss must be aiming for real"
        loss = -torch.mean(input)
    return loss
```

Datasets

COCO-stuff: 118,000 training / 5,000 testing. 182 semantic classes. 굉장히 다양한 분포이기 때문에 기존 모델들은 image synthesis 성능이 좋지 않음.

ADE20K: 20,210 training / 2,000 testing. COCO-stuff 와 비슷하게 150 classes.

ADE20K-outdoor: ADK20K 중 outdoor scenes 만 모아 놓음.

Cityscapes: 3,000 training / 500 testing. Image Synthesis 학습 및 비교에 대표적인 dataset.

Flickr Landscapes: 40,000 training / 1,000 testing. Flickr 에서 직접 수집한 dataset. 이 데이터셋을 사용할 때에는 Pre-trained DeepLabV2 model 을 사용해 segmentation mask 를 생성함.

Figure 5: Visual comparison of semantic image synthesis results on the COCO-Stuff dataset. Our method successfully synthesizes realistic details from semantic labels.

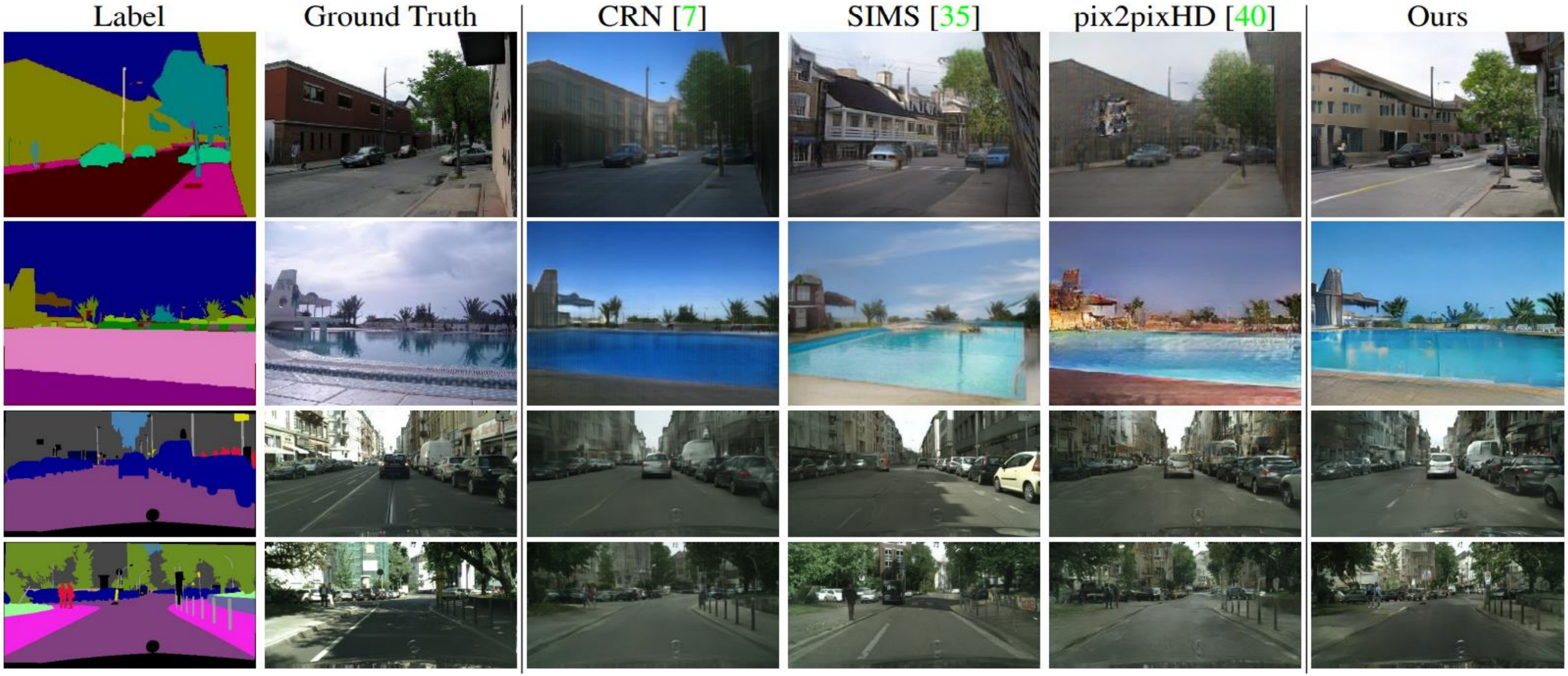
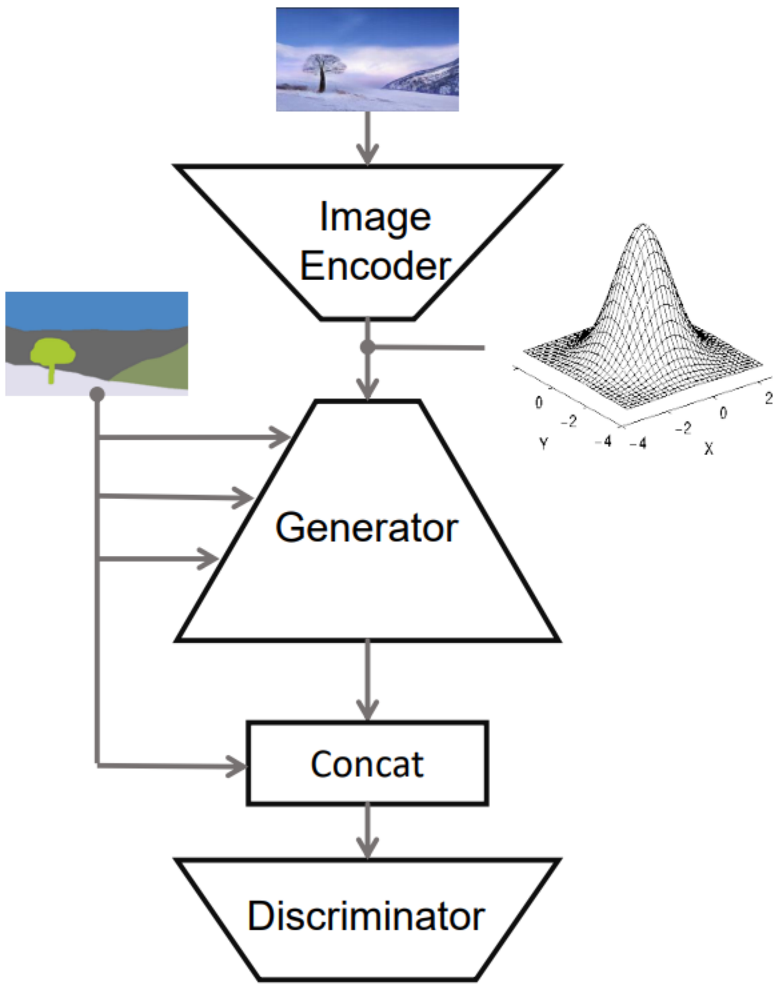


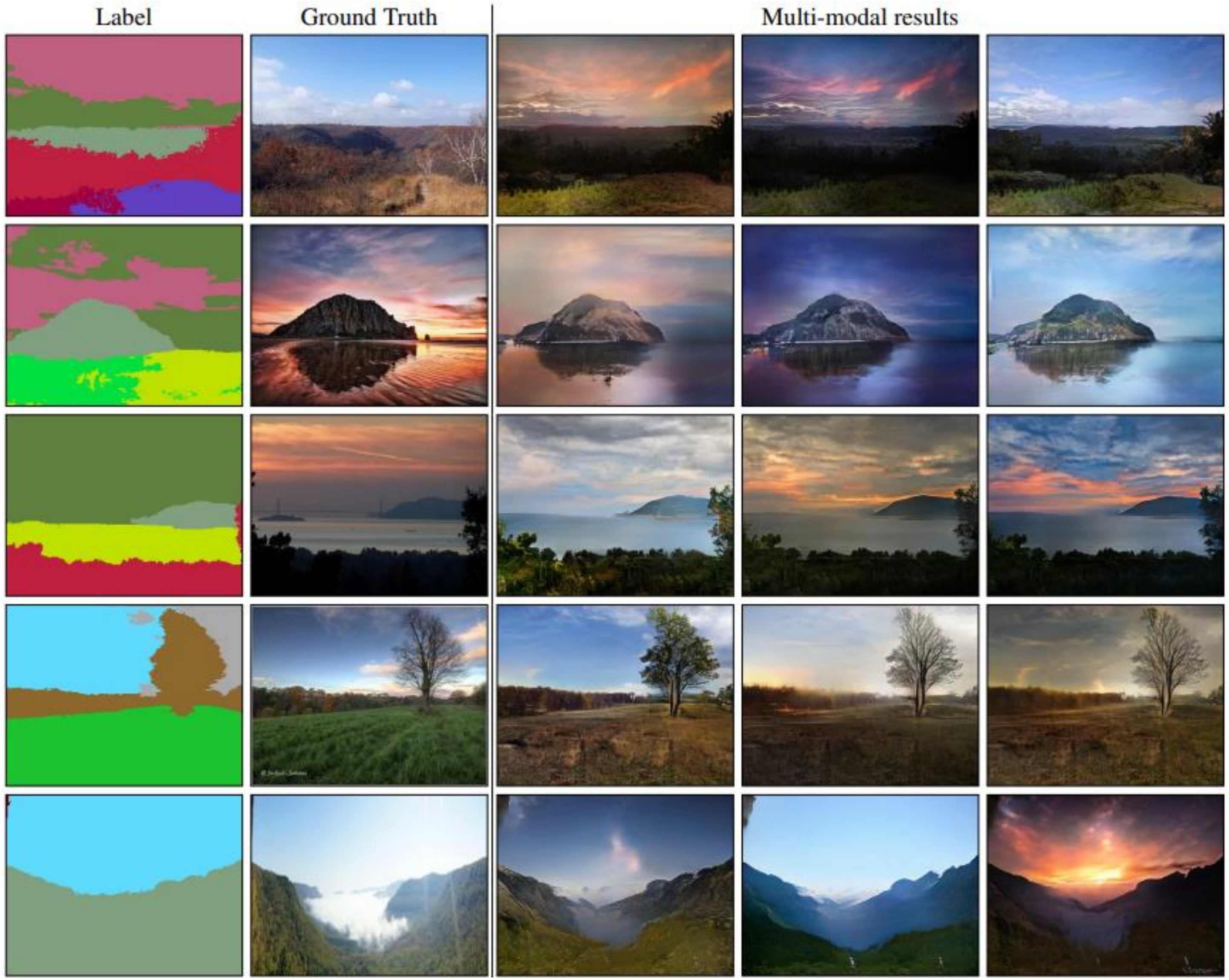
Figure 6: Visual comparison of semantic image synthesis results on the ADE20K outdoor and Cityscapes datasets. Our method produces realistic images while respecting the spatial semantic layout at the same time.

Method	COCO-Stuff			ADE20K			ADE20K-outdoor			Cityscapes		
	mIoU	accu	FID	mIoU	accu	FID	mIoU	accu	FID	mIoU	accu	FID
CRN [7]	23.7	40.4	70.4	22.4	68.8	73.3	16.5	68.6	99.0	52.4	77.1	104.7
SIMS [35]	N/A	N/A	N/A	N/A	N/A	N/A	13.1	74.7	67.7	47.2	75.5	**49.7**
pix2pixHD [40]	14.6	45.8	111.5	20.3	69.2	81.8	17.4	71.6	97.8	58.3	81.4	95.0
Ours	**37.4**	**67.9**	**22.6**	**38.5**	**79.9**	**33.9**	**30.8**	**82.9**	**63.3**	**62.3**	**81.9**	71.8

Table 1: Our method outperforms current leading methods in semantic segmentation scores (mean IoU and overall pixel accuracy) and FID [15] on all the benchmark datasets. For mIoU and pixel accuracy, higher is better. For FID, lower is better.

4. EXPERIMENTS





5. CONCLUSION

Affine transformation 을 학습하면서 semantic layout 정보를 잃어버리지 않는
Spatially-adaptive normalization layer 를 제안한다.

해당 방법을 통해 다양한 scene에 대한 image synthesis 성능을 높일 수 있다.

Multi-modal synthesis 및 guided image synthesis 에 대해 연구를 더 할 생각이다.