GauGAN
Semantic Image Synthesis with Spatially-Adaptive Normalization

Taesung Park[1,2*] Ming-Yu Liu[2] Ting-Chun Wang[2] Jun-Yan Zhu[2,3]

[1]UC Berkeley [2]NVIDIA [2,3]MIT CSAIL

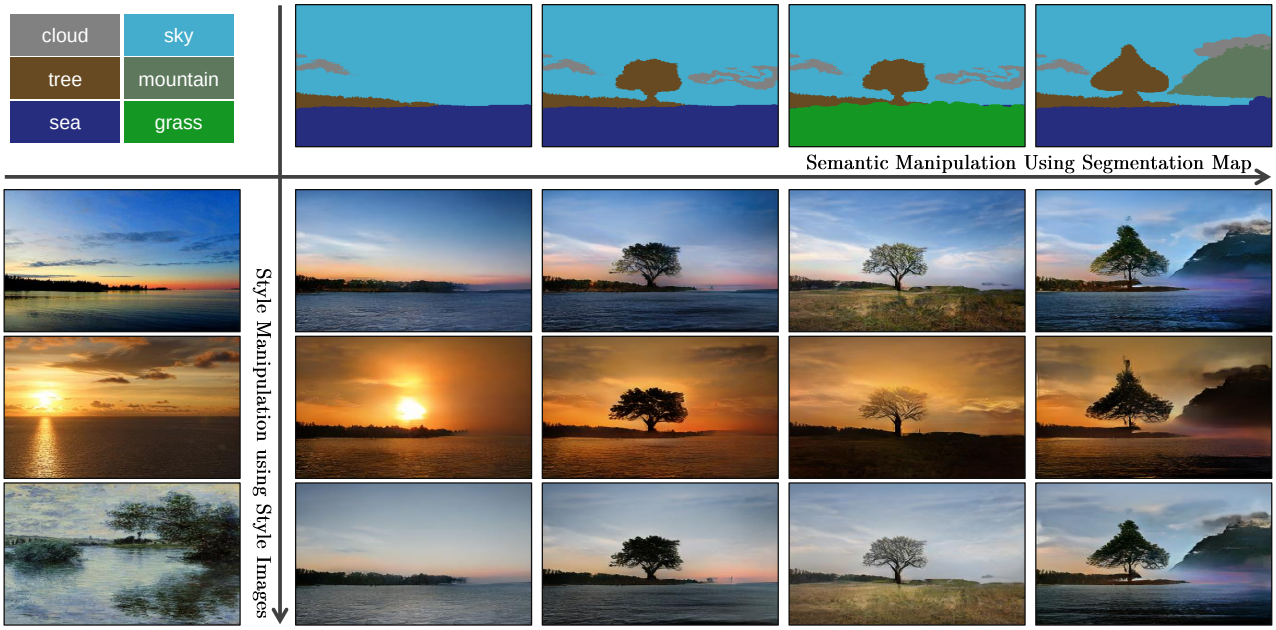


Figure 1: Our model allows user control over both semantic and style as synthesizing an image. The semantic (e.g., the existence of a tree) is controlled via a label map (the top row), while the style is controlled via the reference style image (the leftmost column). Please visit our website for interactive image synthesis demos.

Abstract

We propose spatially-adaptive normalization, a simple but effective layer for synthesizing photorealistic images given an input semantic layout. Previous methods directly feed the semantic layout as input to the deep network, which is then processed through stacks of convolution, normalization, and nonlinearity layers. We show that this is suboptimal as the normalization layers tend to "wash away" semantic information. To address the issue, we propose using the input layout for modulating the activations in normalization layers through a spatially-adaptive, learned transformation. Experiments on several challenging datasets demonstrate the advantage of the proposed method over existing approaches, regarding both visual fidelity and alignment with input layouts. Finally, our model allows user control over both semantic and style. Code is available at https://github.com/NVlabs/SPADE.

1. Introduction

Conditional image synthesis refers to the task of generating photorealistic images conditioning on certain input data. Seminal work computes the output image by stitching pieces from a single image (e.g., Image Analogies [16]) or using an image collection [7, 14, 23, 30, 35]. Recent methods directly learn the mapping using neural networks [3, 6, 22, 47, 48, 54, 55, 56]. The latter methods are faster and require no external database of images.

We are interested in a specific form of conditional image synthesis, which is converting a semantic segmentation mask to a photorealistic image. This form has a wide range of applications such as content generation and image editing [6, 22, 48]. We refer to this form as semantic image synthesis. In this paper, we show that the conventional network architecture [22, 48], which is built by stacking convolutional, normalization, and nonlinearity layers, is at best

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

suboptimal because their normalization layers tend to "wash away" information contained in the 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.

We conduct experiments on several challenging datasets including the COCO-Stuff [4, 32], the ADE20K [58], and the Cityscapes [9]. We show that with the help of our spatially-adaptive normalization layer, a compact network can synthesize significantly better results compared to several state-of-the-art methods. Additionally, an extensive ablation study demonstrates the effectiveness of the proposed normalization layer against several variants for the semantic image synthesis task. Finally, our method supports multi-modal and style-guided image synthesis, enabling controllable, diverse outputs, as shown in Figure 1. Also, please see our SIGGRAPH 2019 Real-Time Live demo and try our online demo by yourself.

2. Related Work

Deep generative models can learn to synthesize images. Recent methods include generative adversarial networks (GANs) [13] and variational autoencoder (VAE) [28]. Our work is built on GANs but aims for the conditional image synthesis task. The GANs consist of a generator and a discriminator where the goal of the generator is to produce realistic images so that the discriminator cannot tell the synthesized images apart from the real ones.

Conditional image synthesis exists in many forms that differ in the type of input data. For example, class-conditional models [3, 36, 37, 39, 41] learn to synthesize images given category labels. Researchers have explored various models for generating images based on text [18,44,52,55]. Another widely-used form is image-to-image translation based on a type of conditional GANs [20, 22, 24, 25, 33, 57, 59, 60], where both input and output are images. Compared to earlier non-parametric methods [7, 16, 23], learning-based methods typically run faster during test time and produce more realistic results. In this work, we focus on converting segmentation masks to photorealistic images. We assume the training dataset contains registered segmentation masks and images. With the proposed spatially-adaptive normalization, our compact network achieves better results compared to leading methods.

Unconditional normalization layers have been an important component in modern deep networks and can be found in various classifiers, including the Local Response Normalization in the AlexNet [29] and the Batch Normalization (BatchNorm) in the Inception-v2 network [21]. Other popular normalization layers include the Instance Normalization (InstanceNorm) [46], the Layer Normalization [2], the Group Normalization [50], and the Weight Normalization [45]. We label these normalization layers as unconditional as they do not depend on external data in contrast to the conditional normalization layers discussed below.

Conditional normalization layers include the Conditional Batch Normalization (Conditional BatchNorm) [11] and Adaptive Instance Normalization (AdaIN) [19]. Both were first used in the style transfer task and later adopted in various vision tasks [3, 8, 10, 20, 26, 36, 39, 42, 49, 54]. Different from the earlier normalization techniques, conditional normalization layers require external data and generally operate as follows. First, layer activations are normalized to zero mean and unit deviation. Then the normalized activations are *denormalized* by modulating the activation using a learned affine transformation whose parameters are inferred from external data. For style transfer tasks [11, 19], the affine parameters are used to control the global style of the output, and hence are uniform across spatial coordinates. In contrast, our proposed normalization layer applies a spatially-varying affine transformation, making it suitable for image synthesis from semantic masks. Wang *et al*. proposed a closely related method for image super-resolution [49]. Both methods are built on spatially-adaptive modulation layers that condition on semantic inputs. While they aim to incorporate semantic information into super-resolution, our goal is to design a generator for style and semantics disentanglement. We focus on providing the semantic information in the context of modulating normalized activations. We use semantic maps in different scales, which enables coarse-to-fine generation. The reader is encouraged to review their work for more details.

3. Semantic Image Synthesis

Let $\mathbf{m} \in \mathbb{L}^{H \times W}$ be a semantic segmentation mask where $\mathbb{L}$ is a set of integers denoting the semantic labels, and H and W are the image height and width. Each entry in $\mathbf{m}$ denotes the semantic label of a pixel. We aim to learn a mapping function that can convert an input segmentation mask $\mathbf{m}$ to a photorealistic image.

Spatially-adaptive denormalization. Let $\mathbf{h}^i$ denote the activations of the i-th layer of a deep convolutional network for a batch of N samples. Let C^i be the number of channels in the layer. Let H^i and W^i be the height and width of the activation map in the layer. We propose a new conditional normalization method called the SPatially-Adaptive (DE)normalization[1] (SPADE). Similar to the Batch Normalization [21], the activation is normalized in the channel-wise manner and then modulated with learned scale and bias. Figure 2 illustrates the SPADE design. The activation

[1]Conditional normalization [11, 19] uses external data to denormalize the normalized activations; i.e., the denormalization part is conditional.

They propose SPatially-Adaptive (DE)normalization called SPADE where its modulation parameters (γ,β) are adaptive to the input segmentation mask and thus results better output.

BatchNorm only picks high frequency component which reduce robustness and it is applied on mini-batch. If the mini-batch size is greater than it results good outputs whereas, less mini-batch size not work well. And that's why InstanceNorm and BatchNorm not works in this case where semantic information is erased in some level and results not realistic image.

SPADE solve that issue: Semantic information washed away through stacks of convolution, normalization, and nonlinearity layers similar to batchNorm, activation is normalized in the channelwise manner, and then modulated with learned γ,β which are adaptive to the input.

Unlike prior conditional normalization, γ,β are not vectors, but tensors with spatial dimensions

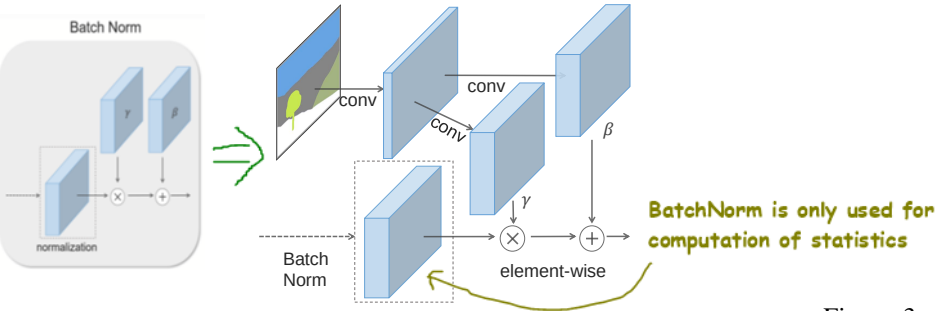


Figure 2: In the SPADE, the mask is first projected onto an embedding space and then convolved to produce the modulation parameters γ and β. Unlike prior conditional normalization methods, γ and β are not vectors, but tensors with spatial dimensions. The produced γ and β are multiplied and added to the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

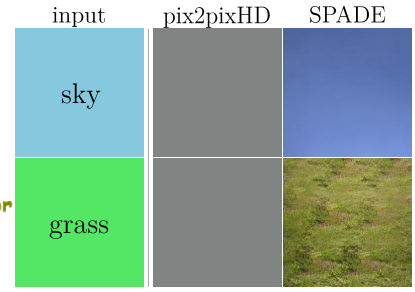


Figure 3: Comparing results given uniform segmentation maps: while the SPADE generator produces plausible textures, the pix2pixHD generator [48] produces two identical outputs due to the loss of the semantic information after the normalization layer.

value at site $(n \in N, c \in C^i, y \in H^i, x \in W^i)$ is

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

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

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

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

The variables $\gamma^i_{c,y,x}(\mathbf{m})$ and $\beta^i_{c,y,x}(\mathbf{m})$ in (1) are the learned modulation parameters of the normalization layer. In contrast to the BatchNorm [21], they depend on the input segmentation mask and vary with respect to the location (y, x). We use the symbol $\gamma^i_{c,y,x}$ and $\beta^i_{c,y,x}$ to denote the functions that convert $\mathbf{m}$ to the scaling and bias values at the site (c, y, x) in the i-th activation map. We implement the functions $\gamma^i_{c,y,x}$ and $\beta^i_{c,y,x}$ using a simple two-layer convolutional network, whose design is in the appendix.

In fact, SPADE is related to, and is a generalization of several existing normalization layers. First, replacing the segmentation mask $\mathbf{m}$ with the image class label and making the modulation parameters spatially-invariant (i.e., $\gamma^i_{c,y_1,x_1} \equiv \gamma^i_{c,y_2,x_2}$ and $\beta^i_{c,y_1,x_1} \equiv \beta^i_{c,y_2,x_2}$ for any $y_1, y_2 \in \{1, 2, ..., H^i\}$ and $x_1, x_2 \in \{1, 2, ..., W^i\}$), we arrive at the form of the Conditional BatchNorm [11]. Indeed, for any spatially-invariant conditional data, our method reduces to the Conditional BatchNorm. Similarly, we can arrive at the AdaIN [19] by replacing $\mathbf{m}$ with a real image, making the modulation parameters spatially-invariant, and setting $N = 1$. As the modulation parameters are adaptive to the input segmentation mask, the proposed SPADE is better suited for semantic image synthesis.

SPADE generator. With the SPADE, there is no need to feed the segmentation map to the first layer of the generator, since the learned modulation parameters have encoded enough information about the label layout. Therefore, we discard encoder part of the generator, which is commonly used in recent architectures [22, 48]. This simplification results in a more lightweight network. Furthermore, similarly to existing class-conditional generators [36, 39, 54], the new generator can take a random vector as input, enabling a simple and natural way for multi-modal synthesis [20, 60].

Encoder is removed results light-weight network

Use of class-Condn Generators similar to pix2pix for multi-modal synthesis

Figure 4 illustrates our generator architecture, which employs several ResNet blocks [15] with upsampling layers. The modulation parameters of all the normalization layers are learned using the SPADE. Since each residual block operates at a different scale, we downsample the semantic mask to match the spatial resolution.

We train the generator with the same multi-scale discriminator and loss function used in pix2pixHD [48] except that we replace the least squared loss term [34] with the hinge loss term [31, 38, 54]. We test several ResNet-based discriminators used in recent unconditional GANs [1, 36, 39] but observe similar results at the cost of a higher GPU memory requirement. Adding the SPADE to the discriminator also yields a similar performance. For the loss function, we observe that removing any loss term in the pix2pixHD loss function lead to degraded generation results.

Use of pix2pixHD with hinge loss to train the generator.

Why does the SPADE work better? A short answer is that it can better preserve semantic information against common normalization layers. Specifically, while normalization layers such as the InstanceNorm [46] are essential pieces in almost all the state-of-the-art conditional image synthesis models [48], they tend to wash away semantic information when applied to uniform or flat segmentation masks.

Let us consider a simple module that first applies convolution to a segmentation mask and then normalization. Furthermore, let us assume that a segmentation mask with a single label is given as input to the module (e.g., all the

SPADE works well since it can well preserve semantic information against common normalization layers whereas, other normalization layers like InstanceNorm when applied to uniform of flat segmentation masks then, they tend to corrupt semantic information.

In SPADE generator, Removes the Encoder part of pix2pixHD which makes model parameters less and smooth training.

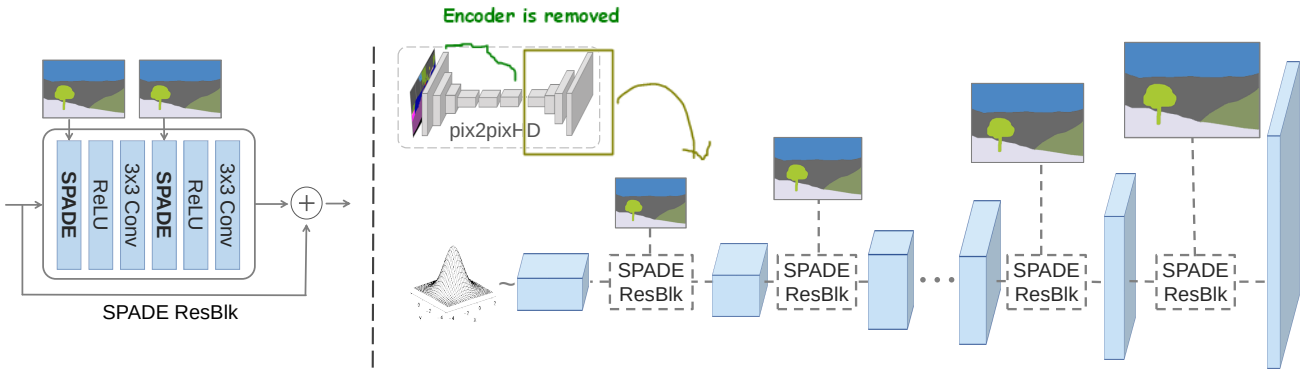


Figure 4: In the SPADE generator, each normalization layer uses the segmentation mask to modulate the layer activations. *(left)* Structure of one residual block with the SPADE. *(right)* The generator contains a series of the SPADE residual blocks with upsampling layers. Our architecture achieves better performance with a smaller number of parameters by removing the downsampling layers of leading image-to-image translation networks such as the pix2pixHD model [48].

pixels have the same label such as sky or grass). Under this setting, the convolution outputs are again uniform, with different labels having different uniform values. Now, after we apply InstanceNorm to the output, the normalized activation will become all zeros no matter what the input semantic label is given. Therefore, semantic information is totally lost. This limitation applies to a wide range of generator architectures, including pix2pixHD and its variant that concatenates the semantic mask at all intermediate layers, as long as a network applies convolution and then normalization to the semantic mask. In Figure 3, we empirically show this is precisely the case for pix2pixHD. Because a segmentation mask consists of a few uniform regions in general, the issue of information loss emerges when applying normalization.

In contrast, the segmentation mask in the SPADE Generator is fed through spatially adaptive modulation *without* normalization. Only activations from the previous layer are normalized. Hence, the SPADE generator can better preserve semantic information. It enjoys the benefit of normalization without losing the semantic input information.

Multi-modal synthesis. By using a random vector as the input of the generator, our architecture provides a simple way for multi-modal synthesis [20, 60]. Namely, one can attach an encoder that processes a real image into a random vector, which will be then fed to the generator. The encoder and generator form a VAE [28], in which the encoder tries to capture the style of the image, while the generator combines the encoded style and the segmentation mask information via the SPADEs to reconstruct the original image. The encoder also serves as a style guidance network at test time to capture the style of target images, as used in Figure 1. For training, we add a KL-Divergence loss term [28].

4. Experiments

Implementation details. We apply the Spectral Norm [38] to all the layers in both generator and discriminator. The learning rates for the generator and discriminator are 0.0001 and 0.0004, respectively [17]. We use the ADAM solver [27] with $\beta_1 = 0$ and $\beta_2 = 0.999$. All the experiments are conducted on an NVIDIA DGX1 with 8 32GB V100 GPUs. We use synchronized BatchNorm, i.e., these statistics are collected from all the GPUs.

Datasets. We conduct experiments on several datasets.

- *COCO-Stuff* [4] is derived from the COCO dataset [32]. It has 118,000 training images and 5,000 validation images captured from diverse scenes. It has 182 semantic classes. Due to its vast diversity, existing image synthesis models perform poorly on this dataset.
- *ADE20K* [58] consists of 20,210 training and 2,000 validation images. Similarly to the COCO, the dataset contains challenging scenes with 150 semantic classes.
- *ADE20K-outdoor* is a subset of the ADE20K dataset that only contains outdoor scenes, used in Qi *et al*. [43].
- *Cityscapes* dataset [9] contains street scene images in German cities. The training and validation set sizes are 3,000 and 500, respectively. Recent work has achieved photorealistic semantic image synthesis results [43, 47] on the Cityscapes dataset.
- *Flickr Landscapes.* We collect 41,000 photos from Flickr and use 1,000 samples for the validation set. To avoid expensive manual annotation, we use a well-trained DeepLabV2 [5] to compute input segmentation masks.

We train the competing semantic image synthesis methods on the same training set and report their results on the same validation set for each dataset.

Performance metrics. We adopt the evaluation protocol from previous work [6, 48]. Specifically, we run a semantic segmentation model on the synthesized images and compare how well the predicted segmentation mask matches the ground truth input. Intuitively, if the output images are realistic, a well-trained semantic segmentation model should be able to predict the ground truth label. For measuring the segmentation accuracy, we use both the mean Intersection-

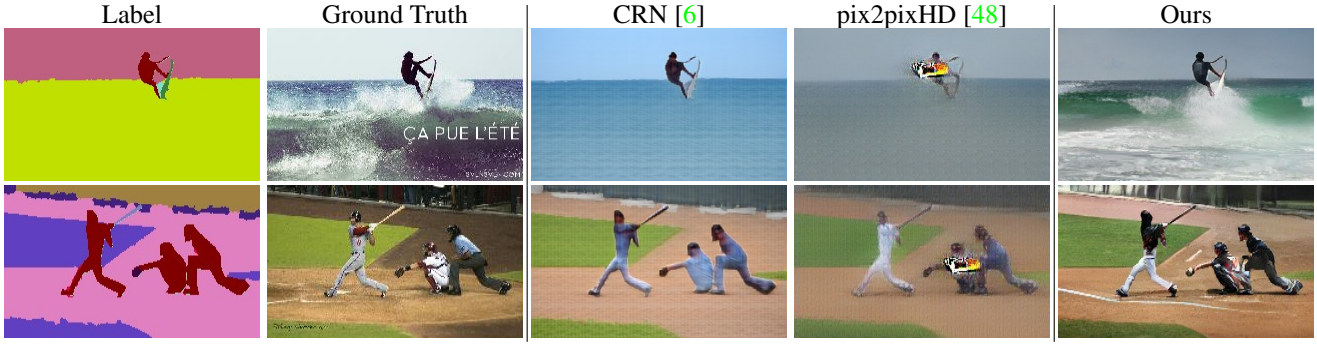


Figure 5: Visual comparison of semantic image synthesis results on the COCO-Stuff dataset. Our method successfully synthesizes realistic details from semantic labels.

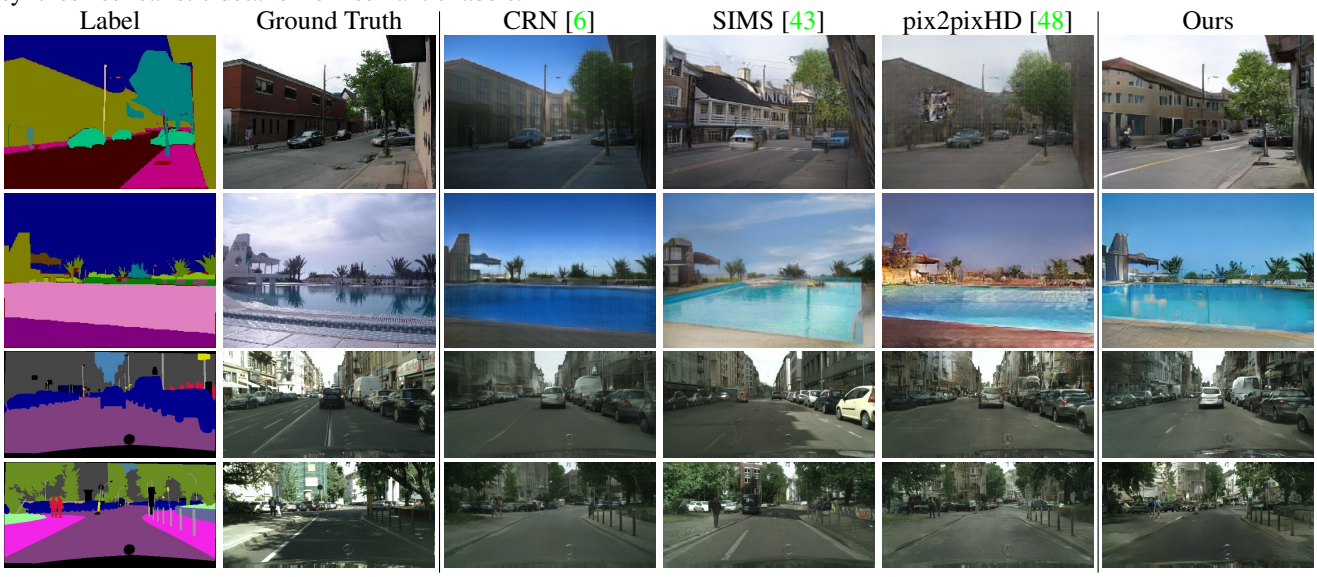


Figure 6: Visual comparison of semantic image synthesis results on the ADE20K outdoor and Cityscapes datasets. Our method produces realistic images while respecting the spatial semantic layout at the same time.

	COCO-Stuff			ADE20K			ADE20K-outdoor			Cityscapes		
Method	mIoU	accu	FID	mIoU	accu	FID	mIoU	accu	FID	mIoU	accu	FID
CRN [6]	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 [43]	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 [48]	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 the current leading methods in semantic segmentation (mIoU and accu) and FID [17] scores on all the benchmark datasets. For the mIoU and accu, higher is better. For the FID, lower is better.

over-Union (mIoU) and the pixel accuracy (accu). We use the state-of-the-art segmentation networks for each dataset: DeepLabV2 [5, 40] for COCO-Stuff, UperNet101 [51] for ADE20K, and DRN-D-105 [53] for Cityscapes. In addition to the mIoU and the accu segmentation performance metrics, we use the Fréchet Inception Distance (FID) [17] to measure the distance between the distribution of synthesized results and the distribution of real images.

Baselines. We compare our method with 3 leading semantic image synthesis models: the pix2pixHD model [48], the cascaded refinement network (CRN) [6], and the semi-parametric image synthesis method (SIMS) [43]. The pix2pixHD is the current state-of-the-art GAN-based conditional image synthesis framework. The CRN uses a deep network that repeatedly refines the output from low to high resolution, while the SIMS takes a semi-parametric approach that composites real segments from a training set and refines the boundaries. Both the CRN and SIMS are mainly trained using image reconstruction loss. For a fair comparison, we train the CRN and pix2pixHD models using the implementations provided by the authors. As image synthesis using the SIMS requires many queries to the training

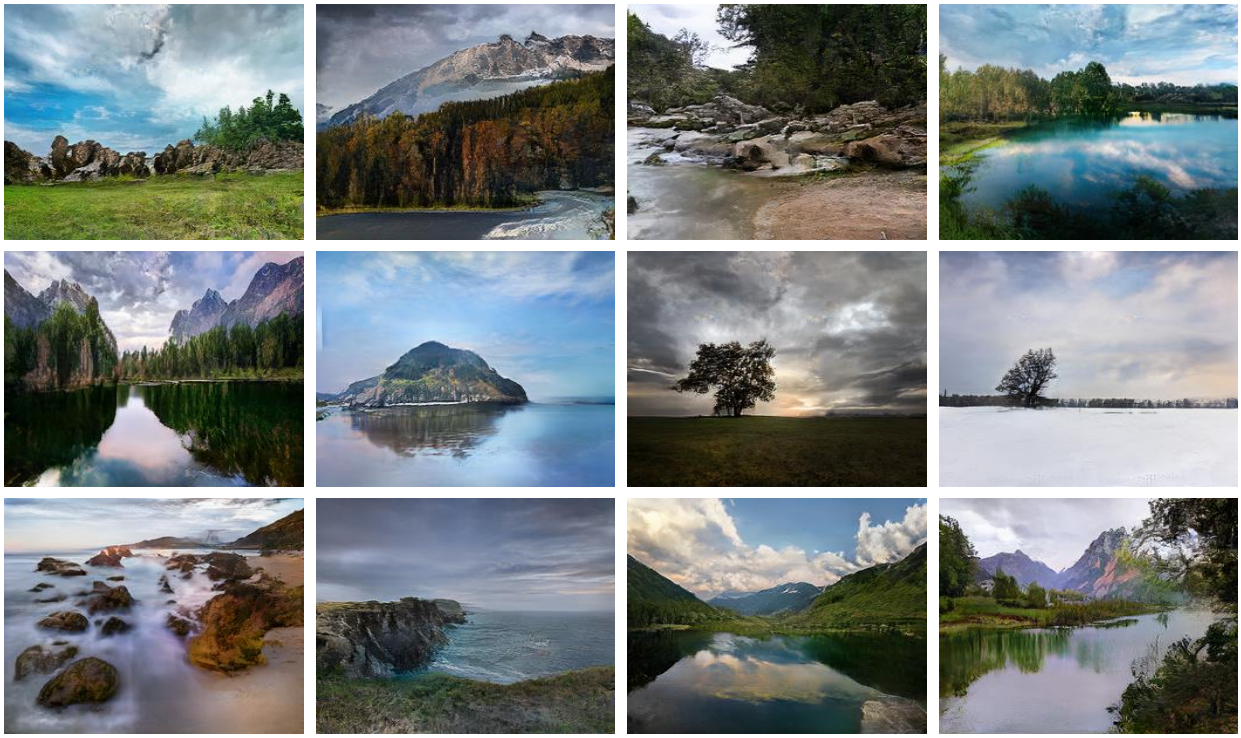


Figure 7: Semantic image synthesis results on the Flickr Landscapes dataset. The images were generated from semantic layout of photographs on the Flickr website.

dataset, it is computationally prohibitive for a large dataset such as the COCO-stuff and the full ADE20K. Therefore, we use the results provided by the authors when available.

Quantitative comparisons. As shown in Table 1, our method outperforms the current state-of-the-art methods by a large margin in all the datasets. For the COCO-Stuff, our method achieves an mIoU score of 35.2, which is about 1.5 times better than the previous leading method. Our FID is also 2.2 times better than the previous leading method. We note that the SIMS model produces a lower FID score but has poor segmentation performances on the Cityscapes dataset. This is because the SIMS synthesizes an image by first stitching image patches from the training dataset. As using the real image patches, the resulting image distribution can better match the distribution of real images. However, because there is no guarantee that a perfect query (e.g., a person in a particular pose) exists in the dataset, it tends to copy objects that do not match the input segments.

Qualitative results. In Figures 5 and 6, we provide qualitative comparisons of the competing methods. We find that our method produces results with much better visual quality and fewer visible artifacts, especially for diverse scenes in the COCO-Stuff and ADE20K dataset. When the training dataset size is small, the SIMS model also renders images with good visual quality. However, the depicted content often deviates from the input segmentation mask (e.g., the shape of the swimming pool in the second row of Figure 6).

Dataset	Ours vs. CRN	Ours vs. pix2pixHD	Ours vs. SIMS
COCO-Stuff	79.76	86.64	N/A
ADE20K	76.66	83.74	N/A
ADE20K-outdoor	66.04	79.34	85.70
Cityscapes	63.60	53.64	51.52

Table 2: User preference study. The numbers indicate the percentage of users who favor the results of the proposed method over those of the competing method.

In Figures 7 and 8, we show more example results from the Flickr Landscape and COCO-Stuff datasets. The proposed method can generate diverse scenes with high image fidelity. More results are included in the appendix.

Human evaluation. We use the Amazon Mechanical Turk (AMT) to compare the perceived visual fidelity of our method against existing approaches. Specifically, we give the AMT workers an input segmentation mask and two synthesis outputs from different methods and ask them to choose the output image that looks more like a corresponding image of the segmentation mask. The workers are given unlimited time to make the selection. For each comparison, we randomly generate 500 questions for each dataset, and each question is answered by 5 different workers. For quality control, only workers with a lifetime task approval rate greater than 98% can participate in our study.

Table 2 shows the evaluation results. We find that users

Figure 8: Semantic image synthesis results on COCO-Stuff. Our method successfully generates realistic images in diverse scenes ranging from animals to sports activities.

Method	#param	COCO.	ADE.	City.
decoder w/ SPADE (Ours)	96M	**35.2**	38.5	62.3
compact decoder w/ SPADE	61M	**35.2**	38.0	**62.5**
decoder w/ Concat	79M	31.9	33.6	61.1
pix2pixHD++ w/ SPADE	237M	34.4	**39.0**	62.2
pix2pixHD++ w/ Concat	195M	32.9	38.9	57.1
pix2pixHD++	183M	32.7	38.3	58.8
compact pix2pixHD++	103M	31.6	37.3	57.6
pix2pixHD [48]	183M	14.6	20.3	58.3

Table 3: The mIoU scores are boosted when the SPADE is used, for both the decoder architecture (Figure 4) and encoder-decoder architecture of pix2pixHD++ (our improved baseline over pix2pixHD [48]). On the other hand, simply concatenating semantic input at every layer fails to do so. Moreover, our compact model with smaller depth at all layers outperforms all the baselines.

Method	COCO	ADE20K	Cityscapes
segmap input	35.2	38.5	62.3
random input	35.3	38.3	61.6
kernelsize 5x5	35.0	39.3	61.8
kernelsize 3x3	35.2	38.5	62.3
kernelsize 1x1	32.7	35.9	59.9
#params 141M	35.3	38.3	62.5
#params 96M	35.2	38.5	62.3
#params 61M	35.2	38.0	62.5
Sync BatchNorm	35.0	39.3	61.8
BatchNorm	33.7	37.9	61.8
InstanceNorm	33.9	37.4	58.7

Table 4: The SPADE generator works with different configurations. We change the input of the generator, the convolutional kernel size acting on the segmentation map, the capacity of the network, and the parameter-free normalization method. The settings used in the paper are boldfaced.

strongly favor our results on all the datasets, especially on the challenging COCO-Stuff and ADE20K datasets. For the Cityscapes, even when all the competing methods achieve high image fidelity, users still prefer our results.

Effectiveness of the SPADE. For quantifying importance of the SPADE, we introduce a strong baseline called pix2pixHD++, which combines all the techniques we find useful for enhancing the performance of pix2pixHD except the SPADE. We also train models that receive the segmentation mask input at all the intermediate layers via feature concatenation in the channel direction, which is termed as pix2pixHD++ w/ Concat. Finally, the model that combines the strong baseline with the SPADE is denoted as pix2pixHD++ w/ SPADE.

As shown in Table 3, the architectures with the proposed SPADE consistently outperforms its counterparts, in both the decoder-style architecture described in Figure 4 and more traditional encoder-decoder architecture used in the pix2pixHD. We also find that concatenating segmentation masks at all intermediate layers, a reasonable alternative to the SPADE, does not achieve the same performance as SPADE. Furthermore, the decoder-style SPADE generator works better than the strong baselines even with a smaller number of parameters.

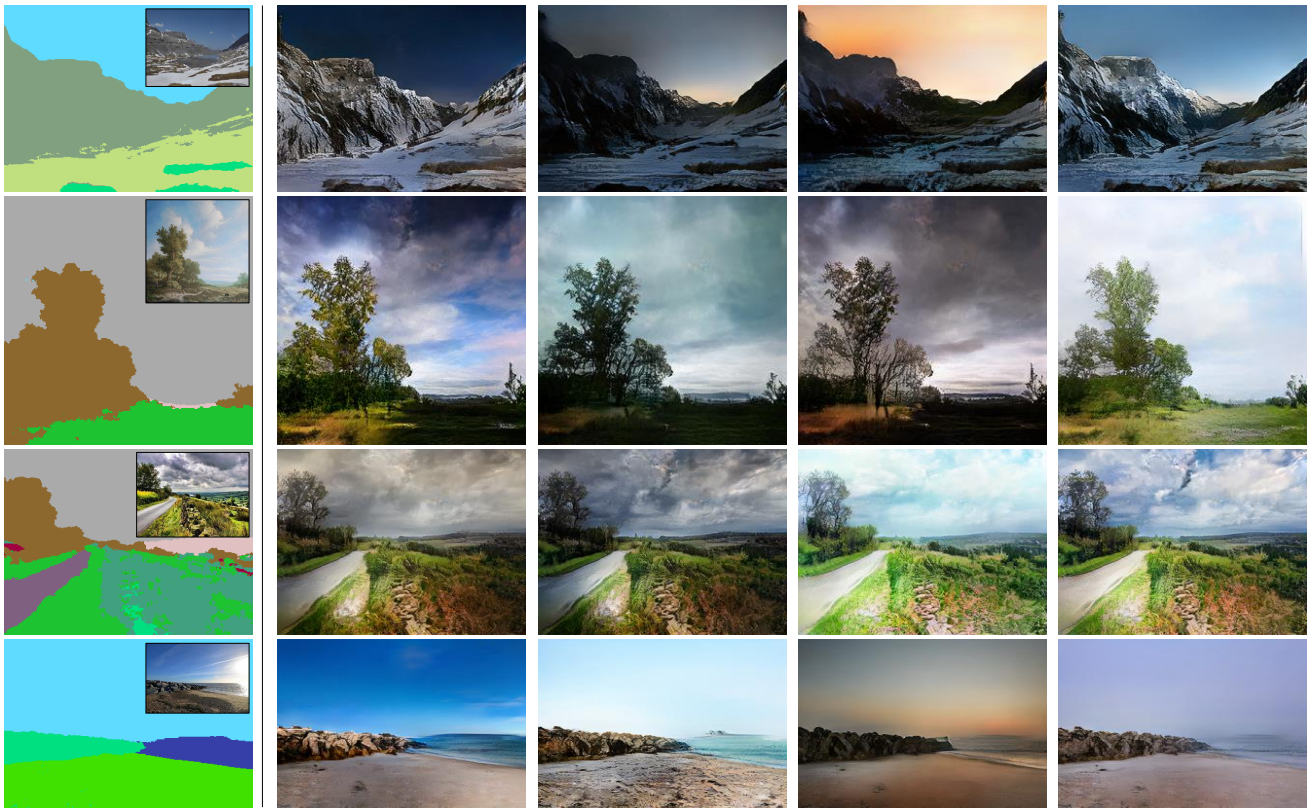


Figure 9: Our model attains multimodal synthesis capability when trained with the image encoder. During deployment, by using different random noise, our model synthesizes outputs with diverse appearances but all having the same semantic layouts depicted in the input mask. For reference, the ground truth image is shown inside the input segmentation mask.

Variations of SPADE generator. Table 4 reports the performance of several variations of our generator. First, we compare two types of input to the generator where one is the random noise while the other is the downsampled segmentation map. We find that both of the variants render similar performance and conclude that the modulation by SPADE alone provides sufficient signal about the input mask. Second, we vary the type of parameter-free normalization layers before applying the modulation parameters. We observe that the SPADE works reliably across different normalization methods. Next, we vary the convolutional kernel size acting on the label map, and find that kernel size of 1x1 hurts performance, likely because it prohibits utilizing the context of the label. Lastly, we modify the capacity of the generator by changing the number of convolutional filters. We present more variations and ablations in the appendix.

Multi-modal synthesis. In Figure 9, we show the multimodal image synthesis results on the Flickr Landscape dataset. For the same input segmentation mask, we sample different noise inputs to achieve different outputs. More results are included in the appendix.

Semantic manipulation and guided image synthesis. In Figure 1, we show an application where a user draws different segmentation masks, and our model renders the corresponding landscape images. Moreover, our model allows users to choose an external style image to control the global appearances of the output image. We achieve it by replacing the input noise with the embedding vector of the style image computed by the image encoder.

5. Conclusion

We have proposed the spatially-adaptive normalization, which utilizes the input semantic layout while performing the affine transformation in the normalization layers. The proposed normalization leads to the first semantic image synthesis model that can produce photorealistic outputs for diverse scenes including indoor, outdoor, landscape, and street scenes. We further demonstrate its application for multi-modal synthesis and guided image synthesis.

Acknowledgments. We thank Alexei A. Efros, Bryan Catanzaro, Andrew Tao, and Jan Kautz for insightful advice. We thank Chris Hebert, Gavriil Klimov, and Brad Nemire for their help in constructing the demo apps. Taesung Park contributed to the work during his internship at NVIDIA. His Ph.D. is supported by a Samsung Scholarship.

References

[1] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning (ICML)*, 2017. 3

[2] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 2

[3] A. Brock, J. Donahue, and K. Simonyan. Large scale gan training for high fidelity natural image synthesis. In *International Conference on Learning Representations (ICLR)*, 2019. 1, 2

[4] H. Caesar, J. Uijlings, and V. Ferrari. Coco-stuff: Thing and stuff classes in context. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2, 4

[5] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 40(4):834–848, 2018. 4, 5

[6] Q. Chen and V. Koltun. Photographic image synthesis with cascaded refinement networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2017. 1, 4, 5, 13, 14, 15, 16, 17, 18

[7] T. Chen, M.-M. Cheng, P. Tan, A. Shamir, and S.-M. Hu. Sketch2photo: internet image montage. *ACM Transactions on Graphics (TOG)*, 28(5):124, 2009. 1, 2

[8] T. Chen, M. Lucic, N. Houlsby, and S. Gelly. On self modulation for generative adversarial networks. In *International Conference on Learning Representations*, 2019. 2

[9] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2, 4

[10] H. De Vries, F. Strub, J. Mary, H. Larochelle, O. Pietquin, and A. C. Courville. Modulating early visual processing by language. In *Advances in Neural Information Processing Systems*, 2017. 2

[11] V. Dumoulin, J. Shlens, and M. Kudlur. A learned representation for artistic style. In *International Conference on Learning Representations (ICLR)*, 2016. 2, 3

[12] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010. 12, 13

[13] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, 2014. 2

[14] J. Hays and A. A. Efros. Scene completion using millions of photographs. In *ACM SIGGRAPH*, 2007. 1

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 3

[16] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin. Image analogies. 2001. 1, 2

[17] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *Advances in Neural Information Processing Systems*, 2017. 4, 5, 13

[18] S. Hong, D. Yang, J. Choi, and H. Lee. Inferring semantic layout for hierarchical text-to-image synthesis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[19] X. Huang and S. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *IEEE International Conference on Computer Vision (ICCV)*, 2017. 2, 3

[20] X. Huang, M.-Y. Liu, S. Belongie, and J. Kautz. Multimodal unsupervised image-to-image translation. *European Conference on Computer Vision (ECCV)*, 2018. 2, 3, 4

[21] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, 2015. 2, 3

[22] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 2, 3, 11, 12

[23] M. Johnson, G. J. Brostow, J. Shotton, O. Arandjelovic, V. Kwatra, and R. Cipolla. Semantic photo synthesis. In *Computer Graphics Forum*, volume 25, pages 407–413, 2006. 1, 2

[24] L. Karacan, Z. Akata, A. Erdem, and E. Erdem. Learning to generate images of outdoor scenes from attributes and semantic layouts. *arXiv preprint arXiv:1612.00215*, 2016. 2

[25] L. Karacan, Z. Akata, A. Erdem, and E. Erdem. Manipulating attributes of natural scenes via hallucination. *arXiv preprint arXiv:1808.07413*, 2018. 2

[26] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[27] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015. 4

[28] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR)*, 2014. 2, 4, 11, 12

[29] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2012. 2

[30] J.-F. Lalonde, D. Hoiem, A. A. Efros, C. Rother, J. Winn, and A. Criminisi. Photo clip art. In *ACM transactions on graphics (TOG)*, volume 26, page 3. ACM, 2007. 1

[31] J. H. Lim and J. C. Ye. Geometric gan. *arXiv preprint arXiv:1705.02894*, 2017. 3, 11

[32] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision (ECCV)*, 2014. 2, 4

[33] M.-Y. Liu, T. Breuel, and J. Kautz. Unsupervised image-to-image translation networks. In *Advances in Neural Information Processing Systems*, 2017. 2

[34] X. Mao, Q. Li, H. Xie, Y. R. Lau, Z. Wang, and S. P. Smolley. Least squares generative adversarial networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2017. 3, 11

[35] T. B. Mathias Eitz, Kristian Hildebrand and M. Alexa. Photosketch: A sketch based image query and compositing system. In *ACM SIGGRAPH 2009 Talk Program*, 2009. 1

[36] L. Mescheder, A. Geiger, and S. Nowozin. Which training methods for gans do actually converge? In *International Conference on Machine Learning (ICML)*, 2018. 2, 3, 11

[37] M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014. 2

[38] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2018. 3, 4, 11

[39] T. Miyato and M. Koyama. cGANs with projection discriminator. In *International Conference on Learning Representations (ICLR)*, 2018. 2, 3, 11

[40] K. Nakashima. Deeplab-pytorch. https://github.com/kazuto1011/deeplab-pytorch, 2018. 5

[41] A. Odena, C. Olah, and J. Shlens. Conditional image synthesis with auxiliary classifier GANs. In *International Conference on Machine Learning (ICML)*, 2017. 2

[42] E. Perez, H. De Vries, F. Strub, V. Dumoulin, and A. Courville. Learning visual reasoning without strong priors. In *International Conference on Machine Learning (ICML)*, 2017. 2

[43] X. Qi, Q. Chen, J. Jia, and V. Koltun. Semi-parametric image synthesis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 4, 5, 13, 17, 18

[44] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. Generative adversarial text to image synthesis. In *International Conference on Machine Learning (ICML)*, 2016. 2

[45] T. Salimans and D. P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems*, 2016. 2

[46] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Instance normalization: The missing ingredient for fast stylization. arxiv 2016. *arXiv preprint arXiv:1607.08022*, 2016. 2, 3

[47] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, G. Liu, A. Tao, J. Kautz, and B. Catanzaro. Video-to-video synthesis. In *Advances in Neural Information Processing Systems*, 2018. 1, 4

[48] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 3, 4, 5, 7, 11, 12, 13, 14, 15, 16, 17, 18

[49] X. Wang, K. Yu, C. Dong, and C. Change Loy. Recovering realistic texture in image super-resolution by deep spatial feature transform. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 606–615, 2018. 2

[50] Y. Wu and K. He. Group normalization. In *European Conference on Computer Vision (ECCV)*, 2018. 2

[51] T. Xiao, Y. Liu, B. Zhou, Y. Jiang, and J. Sun. Unified perceptual parsing for scene understanding. In *European Conference on Computer Vision (ECCV)*, 2018. 5

[52] T. Xu, P. Zhang, Q. Huang, H. Zhang, Z. Gan, X. Huang, and X. He. Attngan: Fine-grained text to image generation with attentional generative adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[53] F. Yu, V. Koltun, and T. Funkhouser. Dilated residual networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 5

[54] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena. Self-attention generative adversarial networks. In *International Conference on Machine Learning (ICML)*, 2019. 1, 2, 3, 11

[55] H. Zhang, T. Xu, H. Li, S. Zhang, X. Huang, X. Wang, and D. Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2017. 1, 2

[56] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. Metaxas. Stackgan++: Realistic image synthesis with stacked generative adversarial networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2018. 1

[57] B. Zhao, L. Meng, W. Yin, and L. Sigal. Image generation from layout. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[58] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba. Scene parsing through ade20k dataset. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2, 4

[59] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2017. 2

[60] J.-Y. Zhu, R. Zhang, D. Pathak, T. Darrell, A. A. Efros, O. Wang, and E. Shechtman. Toward multimodal image-to-image translation. In *Advances in Neural Information Processing Systems*, 2017. 2, 3, 4

A. Additional Implementation Details

Generator. The architecture of the generator consists of a series of the proposed SPADE ResBlks with nearest neighbor upsampling. We train our network using 8 GPUs simultaneously and use the synchronized version of the BatchNorm. We apply the Spectral Norm [38] to all the convolutional layers in the generator. The architectures of the proposed SPADE and SPADE ResBlk are given in Figure 10 and Figure 11, respectively. The architecture of the generator is shown in Figure 12.

Discriminator. The architecture of the discriminator follows the one used in the pix2pixHD method [48], which uses a multi-scale design with the InstanceNorm (IN). The only difference is that we apply the Spectral Norm to all the convolutional layers of the discriminator. The details of the discriminator architecture is shown in Figure 13.

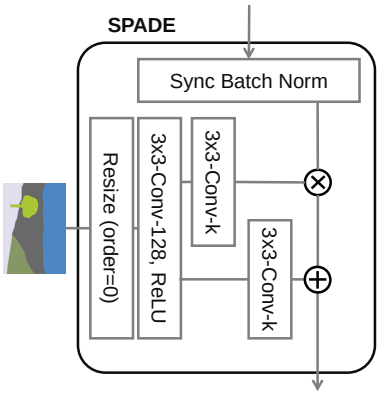


Figure 10: SPADE Design. The term 3x3-Conv-k denotes a 3-by-3 convolutional layer with k convolutional filters. The segmentation map is resized to match the resolution of the corresponding feature map using nearest-neighbor downsampling.

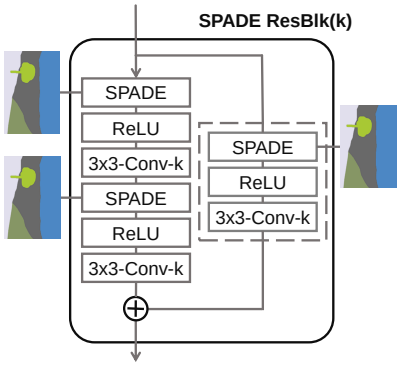


Figure 11: SPADE ResBlk. The residual block design largely follows that in Mescheder *et al*. [36] and Miyato *et al*. [39]. We note that for the case that the number of channels before and after the residual block is different, the skip connection is also learned (dashed box in the figure).

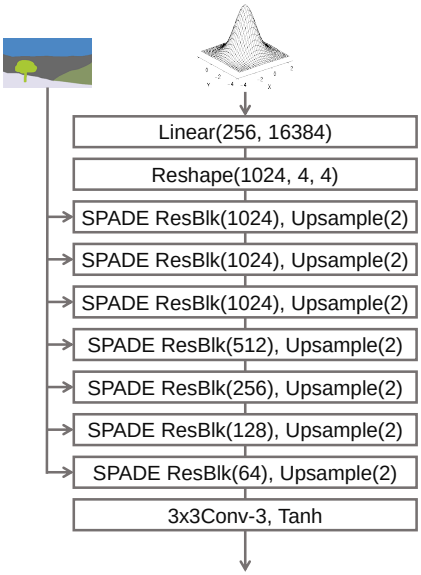


Figure 12: SPADE Generator. Different from prior image generators [22, 48], the semantic segmentation mask is passed to the generator through the proposed SPADE ResBlks in Figure 11.

Image Encoder. The image encoder consists of 6 stride-2 convolutional layers followed by two linear layers to produce the mean and variance of the output distribution as shown in Figure 14.

Learning objective. We use the learning objective function in the pix2pixHD work [48] except that we replace its LSGAN loss [34] term with the Hinge loss term [31, 38, 54]. We use the same weighting among the loss terms in the objective function as that in the pix2pixHD work.

When training the proposed framework with the image encoder for multi-modal synthesis and style-guided image synthesis, we include a KL Divergence loss:

$$\mathcal{L}_{\text{KLD}} = \mathcal{D}_{\text{KL}}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))$$

where the prior distribution $p(\mathbf{z})$ is a standard Gaussian distribution and the variational distribution q is fully determined by a mean vector and a variance vector [28]. We use the reparamterization trick [28] for back-propagating the gradient from the generator to the image encoder. The weight for the KL Divergence loss is 0.05.

In Figure 15, we overview the training data flow. The image encoder encodes a real image to a mean vector and a variance vector. They are used to compute the noise input to the generator via the reparameterization trick [28]. The generator also takes the segmentation mask of the input image as input with the proposed SPADE ResBlks. The

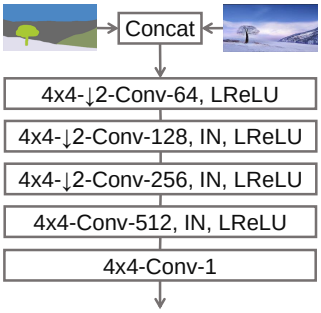


Figure 13: Our discriminator design largely follows that in the pix2pixHD [48]. It takes the concatenation the segmentation map and the image as input. It is based on the PatchGAN [22]. Hence, the last layer of the discriminator is a convolutional layer.

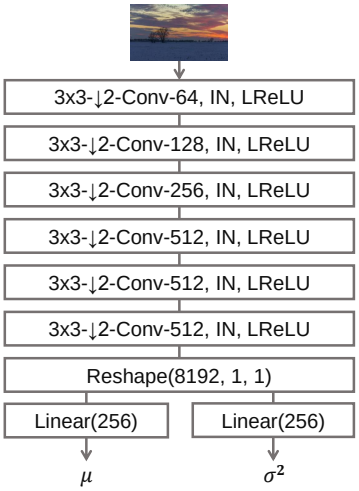


Figure 14: The image encoder consists a series of convolutional layers with stride 2 followed by two linear layers that output a mean vector μ and a variance vector σ.

discriminator takes concatenation of the segmentation mask and the output image from the generator as input and aims to classify that as fake.

Training details. We perform 200 epochs of training on the Cityscapes and ADE20K datasets, 100 epochs of training on the COCO-Stuff dataset, and 50 epochs of training on the Flickr Landscapes dataset. The image sizes are 256×256, except the Cityscapes at 512×256. We linearly decay the learning rate to 0 from epoch 100 to 200 for the Cityscapes and ADE20K datasets. The batch size is 32. We initialize the network weights using thes Glorot initialization [12].

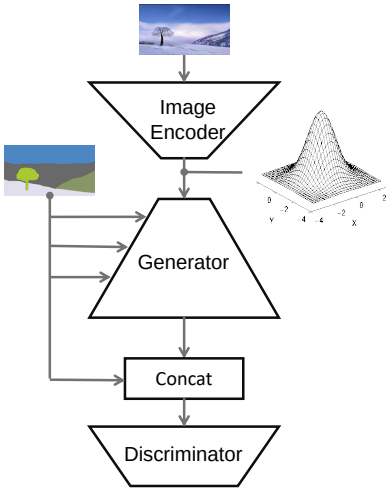


Figure 15: The image encoder encodes a real image to a latent representation for generating a mean vector and a variance vector. They are used to compute the noise input to the generator via the reparameterization trick [28]. The generator also takes the segmentation mask of the input image as input via the proposed SPADE ResBlks. The discriminator takes concatenation of the segmentation mask and the output image from the generator as input and aims to classify that as fake.

B. Additional Ablation Study

Method	COCO.	ADE.	City.
Ours	35.2	38.5	62.3
Ours w/o Perceptual loss	24.7	30.1	57.4
Ours w/o GAN feature matching loss	33.2	38.0	62.2
Ours w/ a deeper discriminator	34.9	38.3	60.9
pix2pixHD++ w/ SPADE	34.4	39.0	62.2
pix2pixHD++	32.7	38.3	58.8
pix2pixHD++ w/o Sync BatchNorm	27.4	31.8	51.1
pix2pixHD++ w/o Sync BatchNorm, and w/o Spectral Norm	26.0	31.9	52.3
pix2pixHD [48]	14.6	20.3	58.3

Table 5: Additional ablation study results using the mIoU metric: the table shows that both the perceptual loss and GAN feature matching loss terms are important. Making the discriminator deeper does not lead to a performance boost. The table also shows that all the components (Synchronized BatchNorm, Spectral Norm, TTUR, the Hinge loss objective, and the SPADE) used in the proposed method helps our strong baseline, pix2pixHD++.

Table 5 provides additional ablation study results analyzing the contribution of individual components in the proposed method. We first find that both of the perceptual loss and GAN feature matching loss inherited from the learning objective function of the pix2pixHD [48] are important. Removing any of them leads to a performance drop. We also find that increasing the depth of the discriminator by inserting one more convolutional layer to the top of the pix2pixHD discriminator does not improve the results.

In Table 5, we also analyze the effectiveness of each component used in our strong baseline, the pix2pixHD++ method, derived from the pix2pixHD method. We found that the Spectral Norm, synchronized BatchNorm, TTUR [17], and the hinge loss objective all contribute to the performance boost. Adding the SPADE to the strong baseline further improves the performance. Note that the pix2pixHD++ w/o Sync BatchNorm and w/o Spectral Norm still differs from the pix2pixHD in that it uses the hinge loss objective, TTUR, a large batch size, and the Glorot initialization [12].

C. Additional Results

In Figure 16, 17, and 18, we show additional synthesis results from the proposed method on the COCO-Stuff and ADE20K datasets with comparisons to those from the CRN [6] and pix2pixHD [48] methods.

In Figure 19 and 20, we show additional synthesis results from the proposed method on the ADE20K-outdoor and Cityscapes datasets with comparison to those from the CRN [6], SIMS [43], and pix2pixHD [48] methods.

In Figure 21, we show additional multi-modal synthesis results from the proposed method. As sampling different $\mathbf{z}$ from a standard multivariate Gaussian distribution, we synthesize images of diverse appearances.

In the accompanying video, we demonstrate our semantic image synthesis interface. We show how a user can create photorealistic landscape images by painting semantic labels on a canvas. We also show how a user can synthesize images of diverse appearances for the same semantic segmentation mask as well as transfer the appearance of a provided style image to the synthesized one.

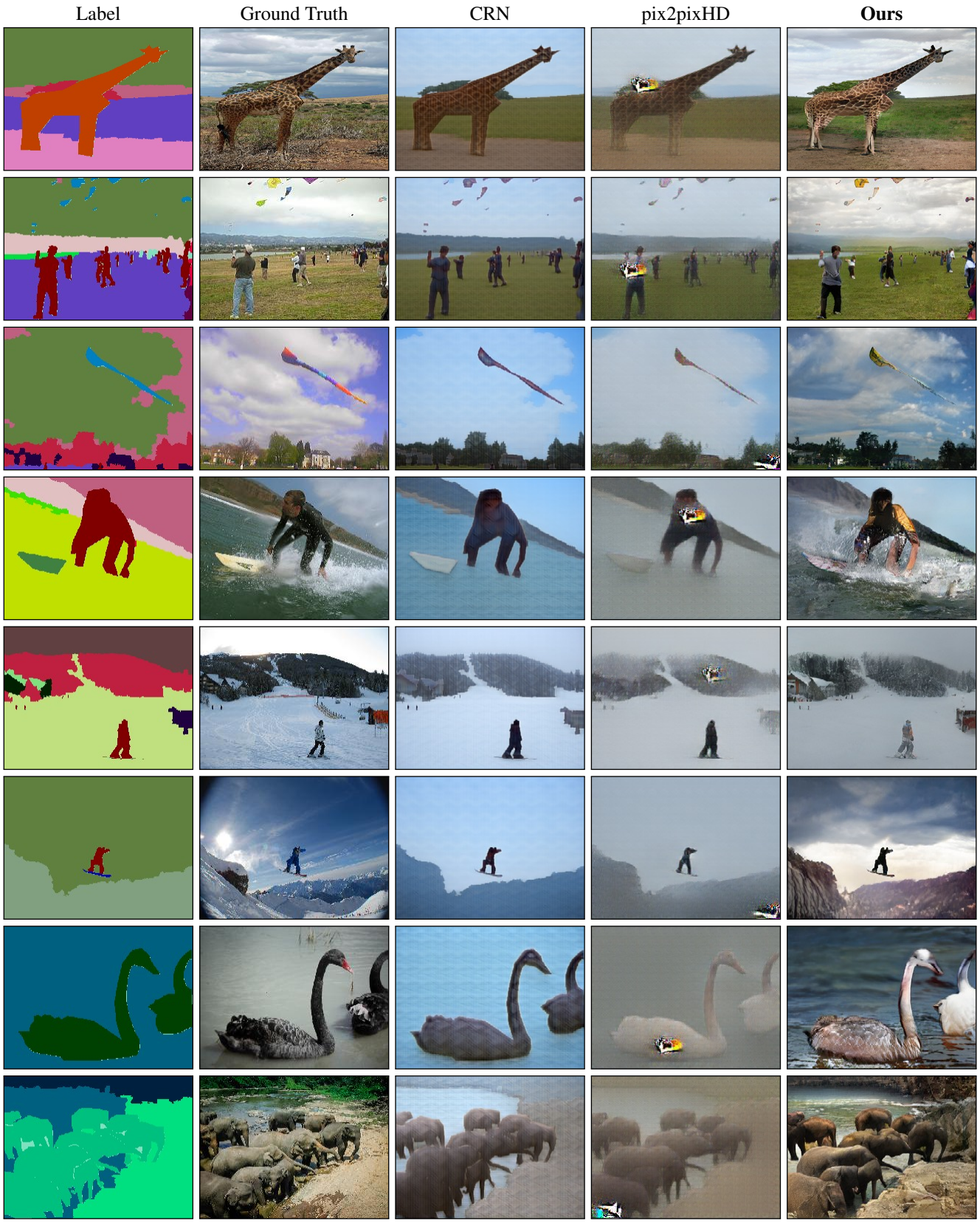


Figure 16: Additional results with comparison to those from the CRN [6] and pix2pixHD [48] methods on the COCO-Stuff dataset.

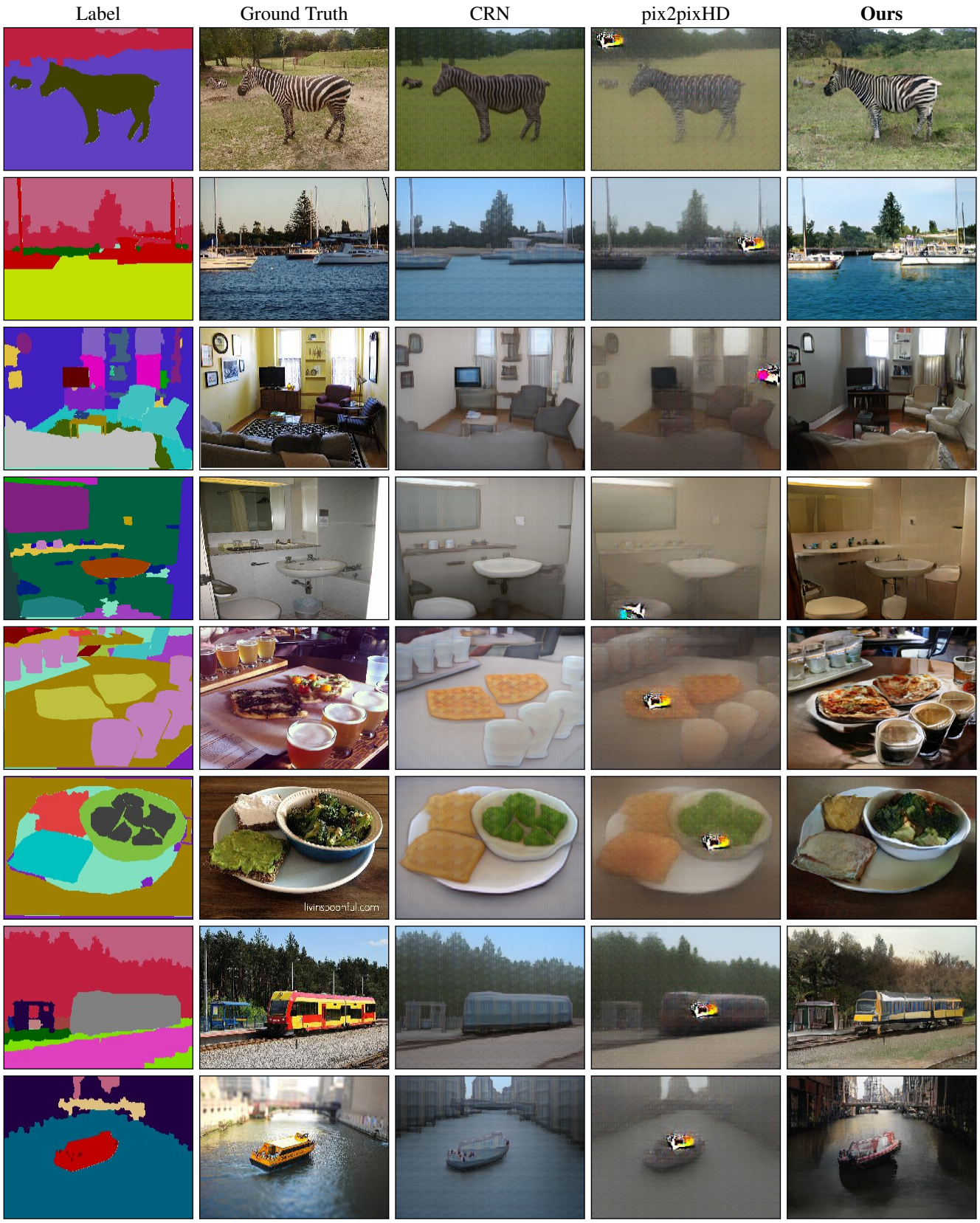


Figure 17: Additional results with comparison to those from the CRN [6] and pix2pixHD [48] methods on the COCO-Stuff dataset.

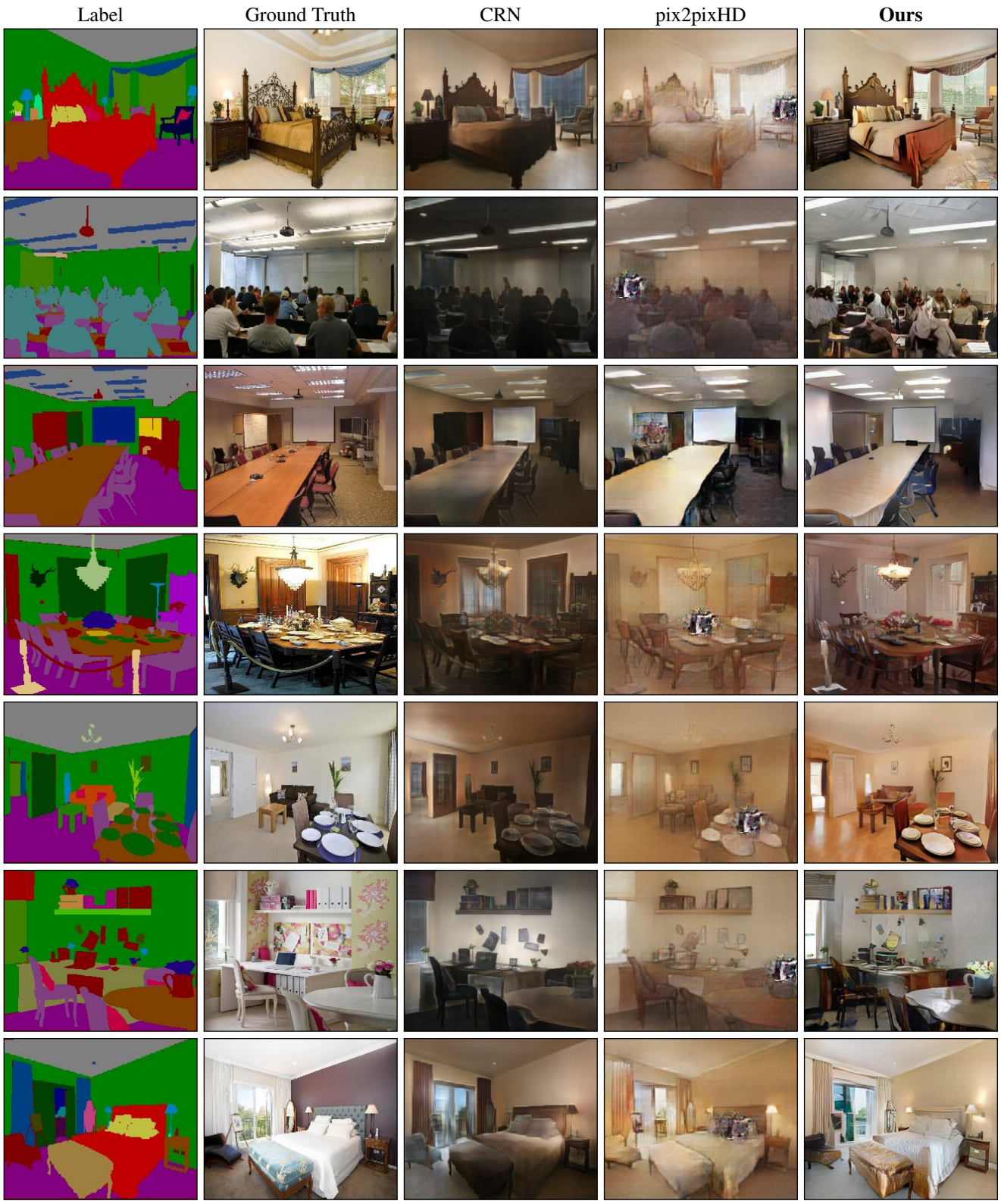


Figure 18: Additional results with comparison to those from the CRN [6] and pix2pixHD [48] methods on the ADE20K dataset.

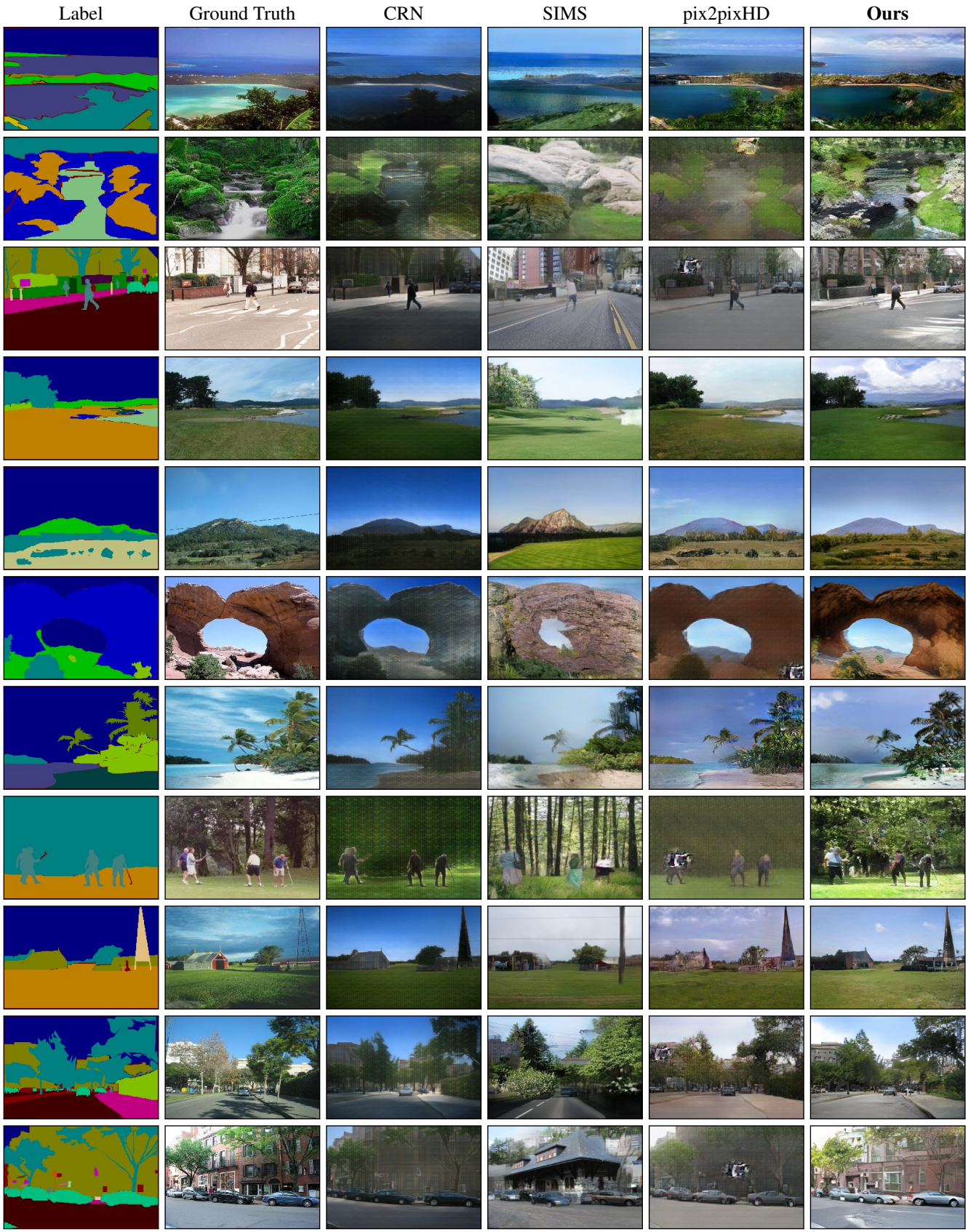


Figure 19: Additional results with comparison to those from the CRN [6], SIMS [43], and pix2pixHD [48] methods on the ADE20K-outdoor dataset.

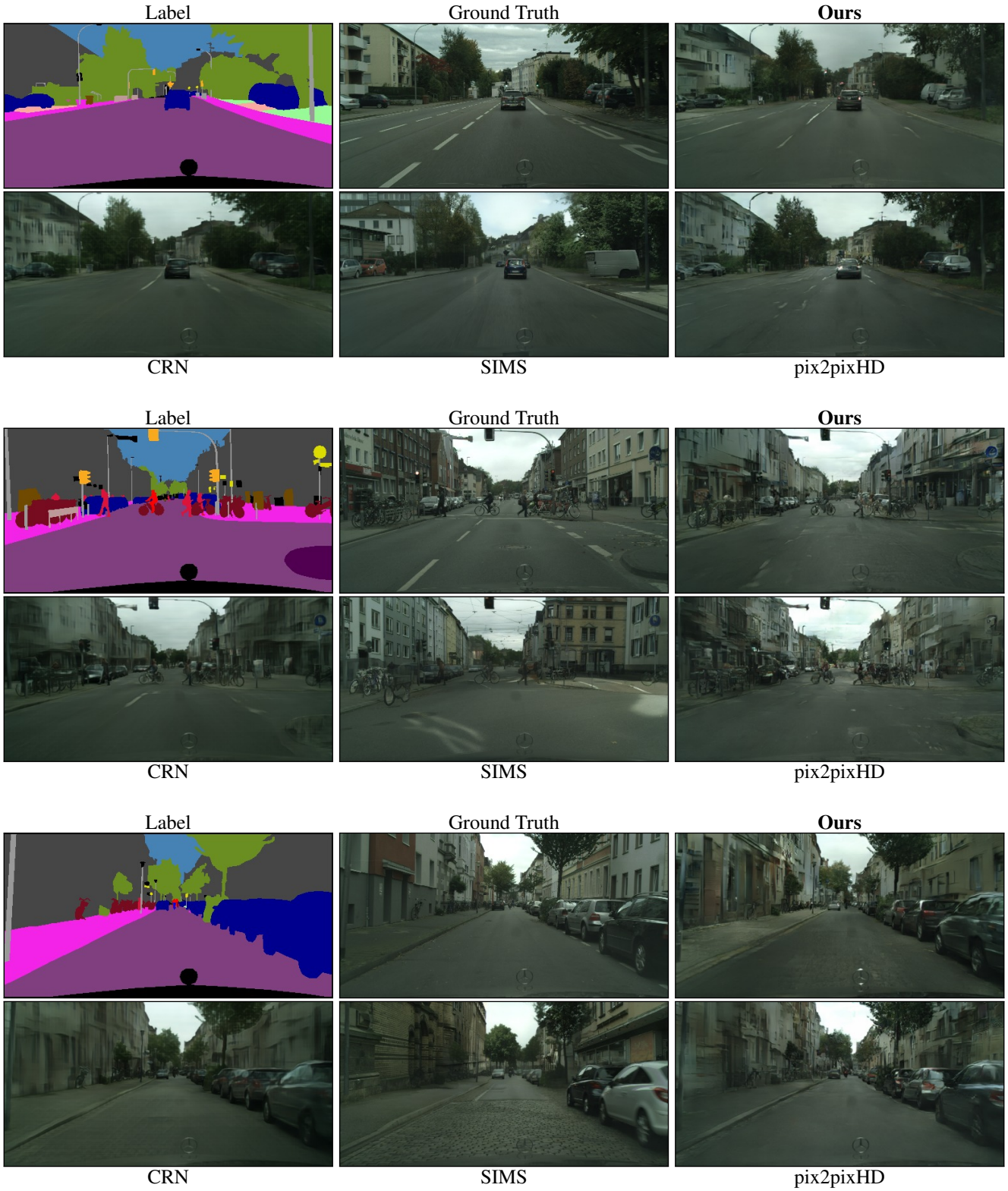


Figure 20: Additional results with comparison to those from the CRN [6], SIMS [43], and pix2pixHD [48] methods on the Cityscapes dataset.

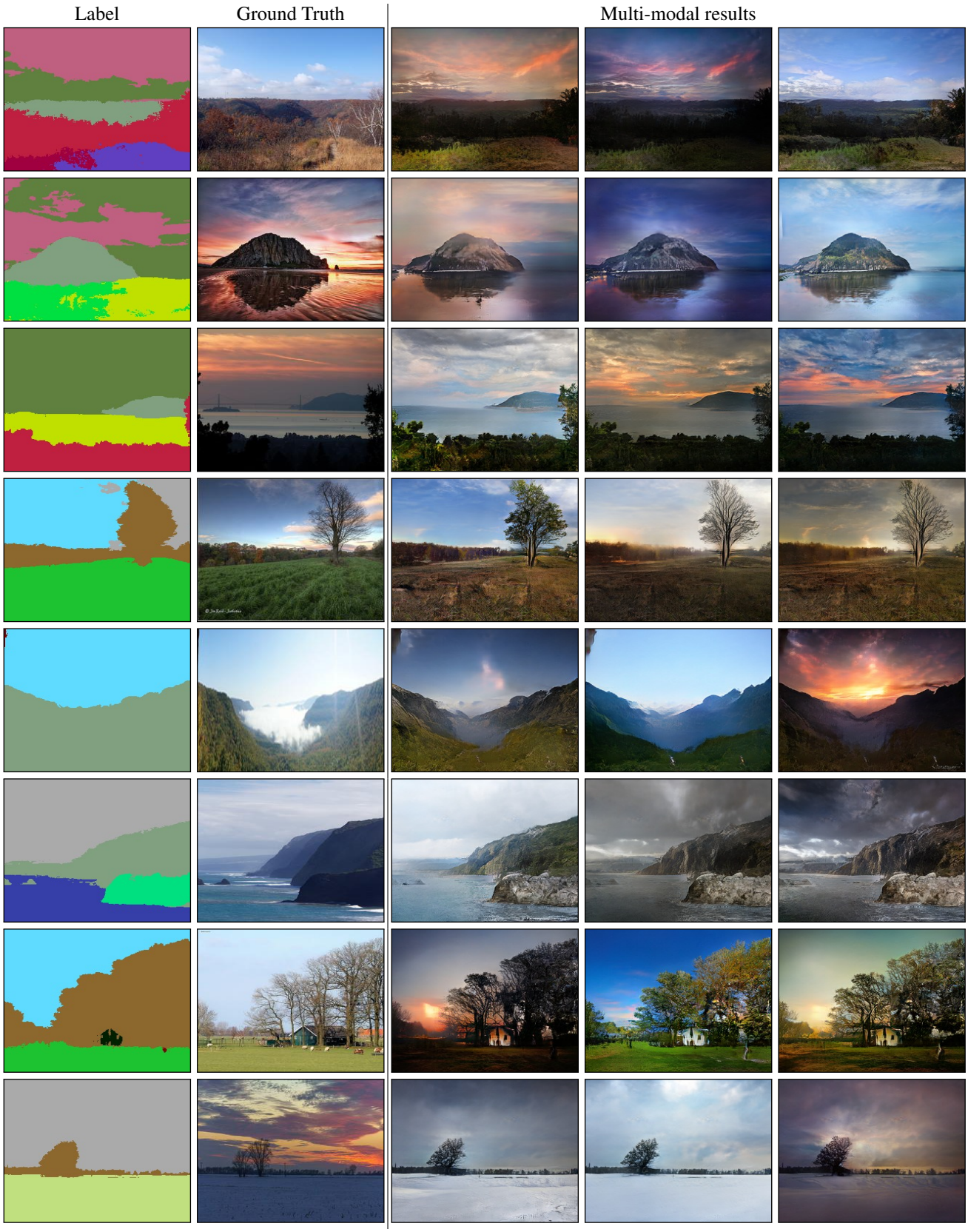


Figure 21: Additional multi-modal synthesis results on the Flickr Landscapes Dataset. By sampling latent vectors from a standard Gaussian distribution, we synthesize images of diverse appearances.

GauGAN Summary

Prepared By: Sushant Gautam
Part of 30 Day GAN paper Reading
https://github.com/sushant097/30-Days-GANs-Paper-Reading

The main idea is to learn mapping function that will map given input image mask(labeled) to realistic image which gives model capability to fill the textures, and other detials in image.

They propose SPatially-Adaptive (DE)normalization called SPADE where its modulation parameters (γ,β) are adaptive to the input segmentation mask and thus results better output.

SPADE solve that issue: Semantic information washed away through stacks of convolution, normalization, and nonlinearity layers similar to batchNorm, activation is normalized in the channelwise manner, and then modulated with learned γ,β which are adaptive to the input.
Unlike prior conditional normalization, γ,β are not vectors, but tensors with spatial dimensions

SPADE Normalization Layer

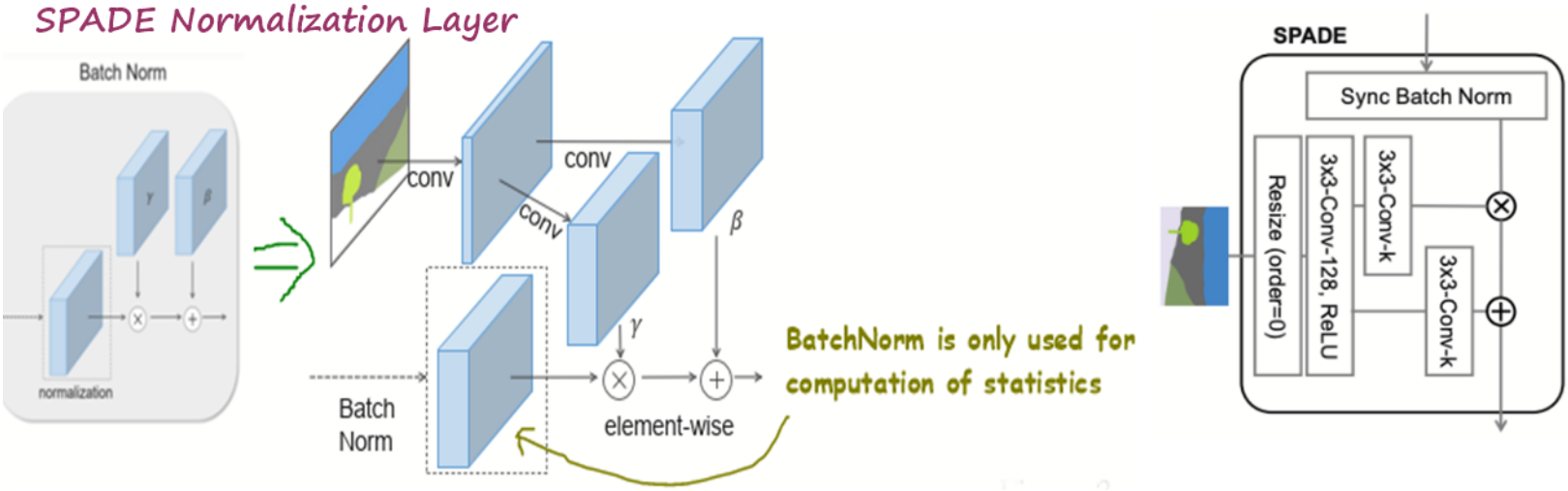


Figure: SPADE

GauGAN has benefit of normalization without losing the semantic input information with SPADE.

Major steps on SPADE:

1. Project segmented mask to embedding space without pre-normalization
2. Pass mask through two layers of convolution to produce γ,β which are tensors with spatial dimension
3. Finally, γ is multiplied as scaling to normalized previous activation map (h) and β is added as element-wise.

Mathematical Equations of SPADE:

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

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

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

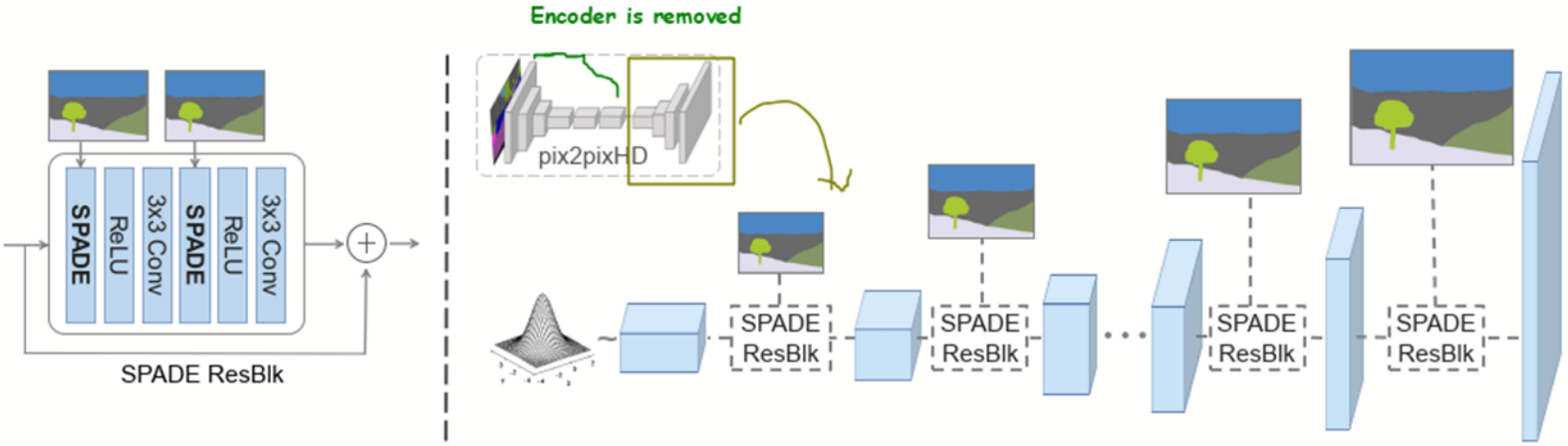
$$\sigma^i_c = \sqrt{\frac{1}{NH^iW^i}\sum_{n,y,x}\left((h^i_{n,c,y,x})^2-(\mu^i_c)^2\right)}. \quad (3)$$

We use the symbol $\gamma^i_{c,y,x}$ and $\beta^i_{c,y,x}$ to denote the functions that convert $\mathbf{m}$ to the scaling and bias values at the site (c,y,x) in the i-th activation map.

The variables $\gamma^i_{c,y,x}(\mathbf{m})$ and $\beta^i_{c,y,x}(\mathbf{m})$ in (1) are the learned modulation parameters of the normalization layer.

SPADE works well since it can well preserve semantic information against common normalization layers whereas, other normalization layers like InstanceNorm when applied to uniform of flat segmentation masks then, they tend to corrupt semantic information.

Architecture of GauGAN:



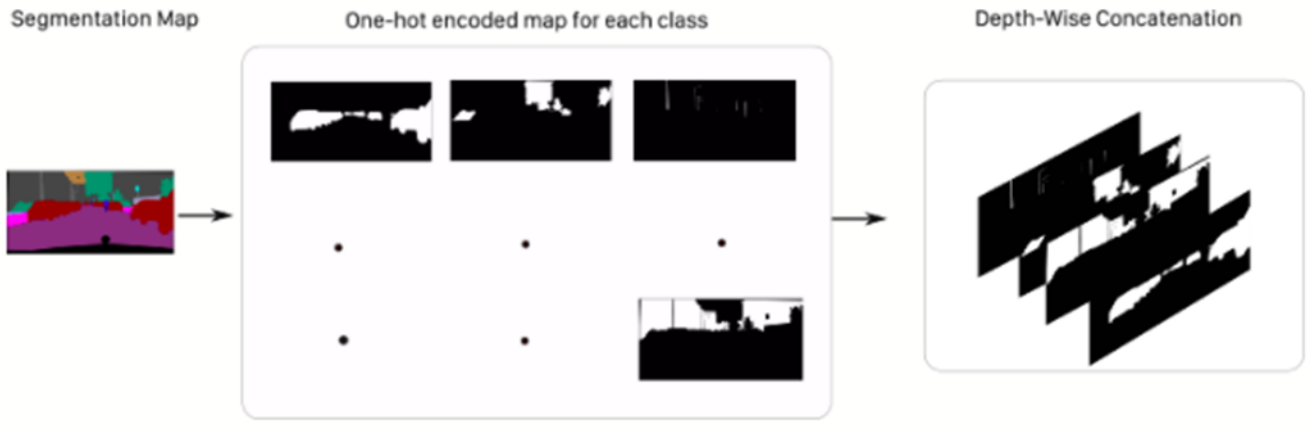
The generator of GauGAN is a fully convolutional decoder consisting of SPADE blocks. It is similar to pix2pixHD architecture but encoder is removed and use of SPADE for normalization.

There are major differences in the architecture of GauGAN's generator and that of Pix2PixHD:

1. No Downsampling layers (Encoding parts) in GauGAN.
2. Provide semantic segmentation map directly as an input to each SPADE block in each level of network as shown in above figure. the segmentation mask in the SPADE Generator is fed through spatially adaptive modulation without normalization. Only activations from the previous layer are normalized. Thus, the SPADE generator can better preserve semantic information.
3. Upsample with nearest neighbour rather than by use of Transpose convolutional layer. As transposed convolutional layer produce artifacts in image and adopt non-learnable upsampling followed by a convolutional layer instea.

Why does SPADE work?

The very first reason is that the input to GauGAN is a semantic map, one-hot encoded.



A different set of batch norm parameters for each pixel is more effective than a single set of batch norm parameters for each channel in the feature map in addressing this task.

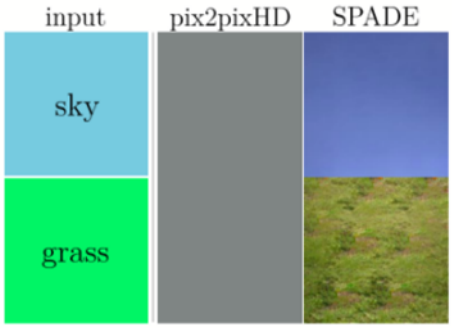


Figure 3: Comparing results given uniform segmentation maps: while SPADE generator produces plausible textures, pix2pixHD [40] produces identical outputs due to the loss of the semantic information after the normalization layer.

Second, the parameters of the Instance Norm layer in Pix2PixHD are non-learnable, and Instance Norm only performs normalization (set to 1 and 0). SPADE has learnable parameters in GauGAN, on the other hand which are adaptive as input.

Discriminator:

Generally, it is a classifier network, with fully connected layers at the end and produces a single output either fake (0) and real (1) in compare to real and generated image.

Discriminator is a multi-scale PatchGAN which is a fully convolutional neural network. It outputs a feature map which is then averaged to get the "realistic-ness" score for the image. Being fully convolutional helps make the GAN process size invariant.

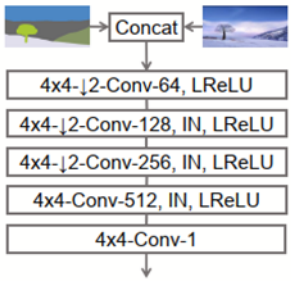


Figure 13: Our discriminator design largely follows that in the pix2pixHD [48]. It takes the concatenation the segmentation map and the image as input. It is based on the PatchGAN [22]. Hence, the last layer of the discriminator is a convolutional layer.

Complete Architecture:

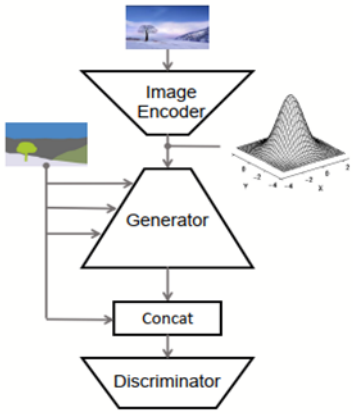


Figure 15: The image encoder encodes a real image to a latent representation for generating a mean vector and a variance vector. They are used to compute the noise input to the generator via the reparameterization trick [28]. The generator also takes the segmentation mask of the input image as input via the proposed SPADE ResBlks. The discriminator takes concatenation of the segmentation mask and the output image from the generator as input and aims to classify that as fake.

Encoder:

Unlike Vanilla GAN, GauGAN doesn't take a random noise vector, but only the semantic map. If we directly pass the semantic map to the generator, than generator output is always deterministic i.e. generator not try to produce diverse image outputs. We don't want our generator only reconstruct the input image. So, for that author use encoder which convert input semantic map to the intermediate tensor through non-linear mapping.

The Encoder basically takes an image, encodes the image into two vectors. These two vectors are used as the mean and standard deviation of a normal Gaussian distribution. A random vector is then sampled from this distribution and then concatenated along with the input Semantic Map as an input to the generator.

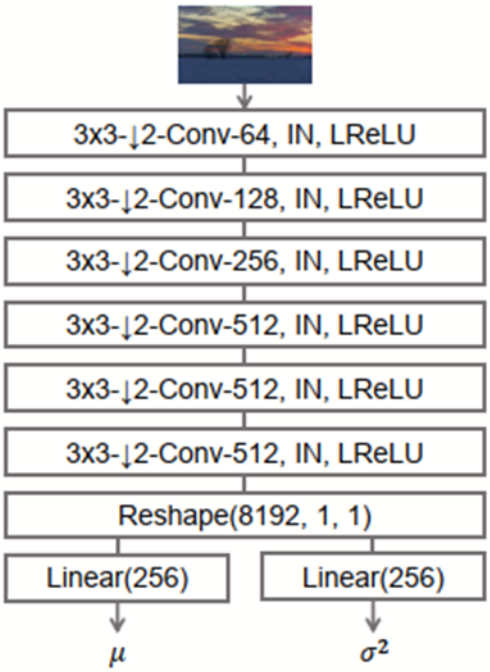


Figure 14: The image encoder consists a series of convolutional layers with stride 2 followed by two linear layers that output a mean vector μ and a variance vector σ.

Loss Function and Training:

Multi-Scale Adversarial Loss:

GauGAN uses hinge loss. We make an image pyramid out of an image generated by the generator, scaling the image to several scales. Then, for each of these scales, we compute the realness score using the discriminator and backpropagate the loss.

$$L_D = -\mathbb{E}_{(x,y)\sim p_{data}}[\min(0, -1 + D(x,y))] - \mathbb{E}_{z\sim p_z, y\sim p_{data}}[\min(0, -1 - D(G(z), y))],$$
$$L_G = -\mathbb{E}_{z\sim p_z, y\sim p_{data}} D(G(z), y),$$

Feature Matching Loss:

This loss constraint the generator to produce images that will not only fool the discriminator but also contains same statistical properties as of original (real) image. For that, we penalize the L1 distance between the discriminator feature maps of the real images and the discriminator feature maps of the fake images. This loss is computed for all the scales of the generated image.

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

where, k denotes which image scale we are using, T is total feature maps from discriminator and Ni is normalizing constant for each feature map that helps to maintain absolute difference between every pair of feature maps despite number of filters in each feature map.

VGG Loss:

The main change is that instead of utilizing the discriminator to compute the feature maps, we utilize a VGG-19 that has been pre-trained on Imagenet to compute the feature maps for real and false images. The L1 distance between these maps is therefore penalized.

$$L_{VGG}(G, D_k) = \mathbb{E}_{s,x} \sum_{i=1}^{5} \frac{1}{2^i}[||VGG(x, M_i) - VGG(G(s), M_i)||_1]$$

$$where\ VGG(x,m)\ is\ the\ feature\ map\ m\ of\ \ VGG19\ when\ x\ is\ the\ input.$$
$$and\ M = \{\text{“}relu1\_1\text{”}, \text{“}relu2\_1\text{”}\text{“}relu3\_1\text{”}\text{“}relu4\_1\text{”}\text{“}relu5\_1\text{”}\}$$

Encoder Loss:

For training they use KL Divergence loss, the proposed framework with the image encoder for multi-modal synthesis and style-guided image synthesis :

$$\mathcal{L}_{\text{KLD}} = \mathcal{D}_{\text{KL}}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))$$

where the prior distribution p (z) is a standard Gaussian distribution and the variational distribution $q(z|x)$ is fully determined by a mean vector and a variance vector. With the Encoder, GauGAN behaves as a sort of Variational AutoEncoder, with the GauGAN playing the part of the decoder.