Pacific Graphics 2020
E. Eisemann, A. Jacobson, and F.-L Zhang
(Guest Editors)

Volume 39 (2020), Number 7

Diversifying Semantic Image Synthesis and Editing via Class- and Layer-wise VAEs

Y. Endo and Y. Kanamori

University of Tsukuba, Japan

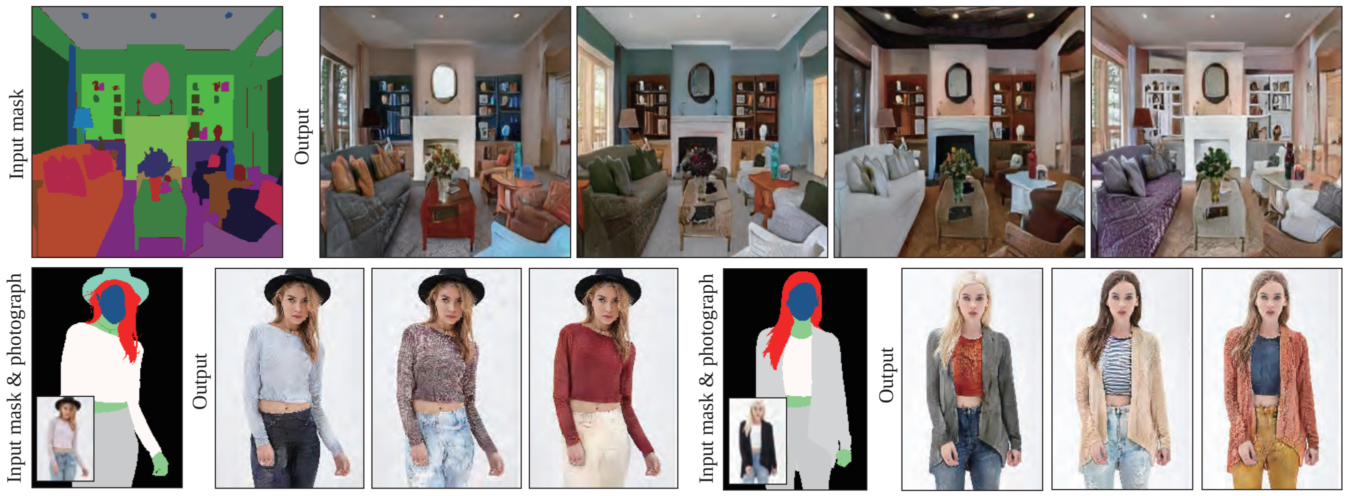

Figure 1: *Results of multimodal semantic image synthesis and editing using our method. Our method yields highly diverse images from a single semantic mask (top), and also enables appearance editing for specific semantic objects, e.g., the clothes in the fashion images (bottom).*

Abstract

Semantic image synthesis is a process for generating photorealistic images from a single semantic mask. To enrich the diversity of multimodal image synthesis, previous methods have controlled the global appearance of an output image by learning a single latent space. However, a single latent code is often insufficient for capturing various object styles because object appearance depends on multiple factors. To handle individual factors that determine object styles, we propose a class- and layer-wise extension to the variational autoencoder (VAE) framework that allows flexible control over each object class at the local to global levels by learning multiple latent spaces. Furthermore, we demonstrate that our method generates images that are both plausible and more diverse compared to state-of-the-art methods via extensive experiments with real and synthetic datasets in three different domains. We also show that our method enables a wide range of applications in image synthesis and editing tasks.

CCS Concepts

• ***Computing methodologies*** → ***Artificial intelligence; Image manipulation;***

1. Introduction

Semantic image synthesis is a fascinating technique for converting a semantic mask into natural images. This technique has been gaining considerable attention because of its broad applications such as design exploration and content creation. Ideally, for these practical applications, semantic image synthesis should generate realistic and diverse images from a single semantic mask. Thus we seek *multimodal* image synthesis, which entails a one-to-many mapping problem.

Semantic image synthesis has advanced significantly since the emergence of *conditional generative adversarial networks* (cGANs) [IZZE17]. Early cGAN techniques were limited to *unimodal image synthesis* (i.e., one-to-one mapping from a semantic mask to an output image) [WLZ*18]. Multimodal image synthesis uses several approaches, such as the *variational autoencoder* (VAE) [KW14], combinations of VAE and GAN (VAE-GAN) [LSLW16; ZZP*17b; PLWZ19; LYS*19], and *implicit maximum likelihood estimation* (IMLE) [LM18; LZM19]. Nonetheless, the variety of output images is still limited because a single latent code controls the entire content in each image. Consider the interior design of indoor scenes (e.g., the top row of Figure 1). We should be able to change the design of the furniture (e.g., a sofa, desk, and shelf) according to users' preferences. A single latent representation is thus insufficient to capture such style variations adequately.

In this study, we enhance the VAE-GAN framework to introduce flexible control over each semantic class while supporting a large number of class labels. Unlike previous approaches where a single latent code is extracted only from the last layer of the encoder, our key idea is to learn the class-wise latent space from each layer of the encoder to capture multiple factors that affect the class-wise object appearance. Next, class-wise latent codes are extracted from the intermediate layers of the encoder using the semantic mask, spatially replicated according to the mask's spatial structure, and then fed to the corresponding generator blocks. Hence, we can synthesize the diverse image content for each class. The layer-wise latent codes allow flexible image editing at local (e.g., textures) to global (e.g., entire color tones) levels, as proposed in *StyleGAN* [KLA19; KLA*19]. The linear functions for sampling the latent codes are shared among the class-wise latent codes in each layer. Therefore our approach is independent of and thus scalable with the number of class labels. To suppress the degradation of generalization performance, we introduce additional regularization for latent codes during training.

Our class- and layer-wise VAEs are a simple yet powerful extension for modeling various styles more efficiently than state-of-the-art methods. We demonstrate that our method can generate images that are both plausible and more diverse than the competing methods via experiments with the following datasets: *ADE20K* [ZZP*17a], *DeepFashion* [LLQ*16], and *GTA-5* [RVRK16]. In addition, we show that our method has interesting applications in image synthesis and editing tasks, such as the exploration of interior or clothing design, as illustrated in Figure 1.

2. Related Work

2.1. Conventional Image Editing

Various image editing techniques have been proposed in the field of computer graphics. Image analogies [HJO*01] can apply a transformation between a pair of aligned images to another image. Certain methods use reference images in a database to edit objects and textures in a target image [CCT*09; HE07; LHE*07]. These non-parametric methods can produce photorealistic results if the reference images retrieved from the database match target images. However, the generalization performance for various inputs is low, and the cost of running the database is high.

2.2. Generative Adversarial Networks

A parametric approach for image generation has evolved since the advent of *generative adversarial networks* (GANs) [GPM*14], and the quality of the generated images has greatly improved. The recent GAN-based technique known as *StyleGAN* [KLA19; KLA*19] can produce high quality and diverse images via feature normalization using *adaptive instance normalization* (AdaIN) [HB17] and latent code regularization. While GANs were originally designed to generate various outputs from random noise, they have been adapted to a variety of tasks by conditioning them on input data, i.e., *conditional GANs* (cGANs).

2.3. Image-to-Image Translation

A successful example of cGANs is the image-to-image translation, in which an input image is converted to another image domain. *Pix2pix* [IZZE17] is a seminal work of image-to-image translation that showcased image domains such as sketches, maps, and semantic masks. Moreover, pix2pix was extended to support higher resolution images [WLZ*18] and unsupervised settings [ZPIE17; CCK*18]. The encoder-decoder architectures for image-to-image translation also play an important role in various specific tasks in computer graphics [PHS*18; BSP*19; EKD*17; EKM17; ZLW*18; KE18; SBT*19].

2.4. Semantic Image Synthesis

Synthesizing realistic and diverse images from a semantic mask is an ill-posed and challenging problem, which we address in this study. Although pix2pix supports semantic image synthesis, it suffers from *mode collapse*; that is, pix2pix obtains only a single plausible image even when using latent noise as additional input to control the output. To solve this problem, various methods for multimodal semantic image synthesis have been proposed. The cascaded refinement network (CRN) [CK17] outputs a fixed number of images while encouraging their diversity via the diversity loss in the training phase. Ghosh et al. achieved multimodal image synthesis with multiple generators trained for different modes [GKN*18]. However, these methods can only generate a fixed number of images from a single input image.

To handle a variable number of modes while avoiding mode collapse, the variational autoencoder (VAE) [KW14] is used jointly with the GAN; this is also known as the VAE-GAN [LSLW16]. BicycleGAN [ZZP*17b] employed a conditional version of the VAE-GAN with a latent regressor [DKD17; DKD17; CDH*16] for multimodal image synthesis. Most recently, Li et al. [LZM19] used *conditional implicit maximum likelihood estimation* (cIMLE) [LM18] instead of GANs. Unlike BicycleGAN, which uses multiple networks such as generators, discriminators, and encoders, they stabilized the training using a single network. *GauGAN* [PLWZ19] improved the quality of semantic image synthesis significantly using normalization suitable for semantic masks. Its VAE-GAN variant facilitates high-quality multimodal image synthesis. Subsequent studies have also improved the quality of image synthesis [LYS*19]. However, all these methods use a single latent code for a single input mask. By contrast, our method enables diverse image synthesis and flexible editing by learning the latent spaces per class and per layer.

A concurrent study [ZXYB20] diversifies multimodal image synthesis with a similar motivation to our work; namely, leveraging group convolutions. Their method showed good performance for datasets with a small number of semantic classes (e.g., eight in a simplified version of *DeepFashion*), but it struggled to handle a large number of classes. By contrast, our method is scalable regarding the number of classes (see Section 1) and significantly outperforms the aforementioned method in datasets with many classes (see Section 4).

3. Method

Our method adopts the VAE-GAN framework. For the generator, we employ a modified version of GauGAN [PLWZ19], which is a state-of-the-art method capable of generating high-quality images from a semantic mask. In this section, we first briefly review GauGAN and then explain our class- and layer-wise VAEs.

3.1. Background

GauGAN [PLWZ19] introduces a normalization module called *spatially adaptive denormalization* (SPADE). The SPADE module normalizes an input feature map from the previous layer and then updates the feature map using modulation parameters, which are calculated using convolutions for the input semantic mask. Unlike other normalization layers, the modulation parameters have spatially different values and are used via pixel-wise multiplication and summation. Thus, SPADE successfully retains the spatial information of the semantic mask in the feature map. We have omitted the details of the SPADE module because they are not relevant to our method.

Figure 2 (a) illustrates the SPADE generator, which is a lightweight network consisting of a decoder only. Given the semantic mask $\mathbf{M} \in \{0,1\}^{C\times W\times H}$ (where C, W, and H are the total number of classes, width, and height) in a one-hot representation, the SPADE generator G calculates an output image as follows:

$$G(\mathbf{M}) = g_K(\mathbf{M}_K, g_{K-1}(\mathbf{M}_{K-1}, \cdots g_1(\mathbf{M}_1))), \tag{1}$$

where g_k is a kth block, and K is the number of blocks. $\mathbf{M}_k \in \{0,1\}^{C\times W_k\times H_k}$ is a downsampled semantic mask whose width W_k and height H_k are the same as those of the corresponding feature map. In the case of unimodal image synthesis, the first block g_1 is a simple convolutional layer that takes $\mathbf{M}_1$ as an input. Meanwhile, g_k ($k \geq 2$) consists of the SPADE ResNet block, which takes an intermediate feature map as well as $\mathbf{M}_k$ as inputs.

In the case of multimodal image synthesis based on GauGAN-VAE [PLWZ19], a latent code $\mathbf{z}_1 \in \mathbb{R}^{N_1}$ is used as an additional input as follows:

$$G(\mathbf{M}, \mathbf{z}_1) = g_K(\mathbf{M}_K, g_{K-1}(\mathbf{M}_{K-1}, \cdots g_1(\mathbf{z}_1))). \tag{2}$$

In this case, g_1 consists of linear and reshaping functions, which acquires a feature map from the input vector. The latent code $\mathbf{z}_1$ is sampled from the prior at the test time:

$$\mathbf{z}_1 \sim \mathcal{N}(\mathbf{0}, I), \tag{3}$$

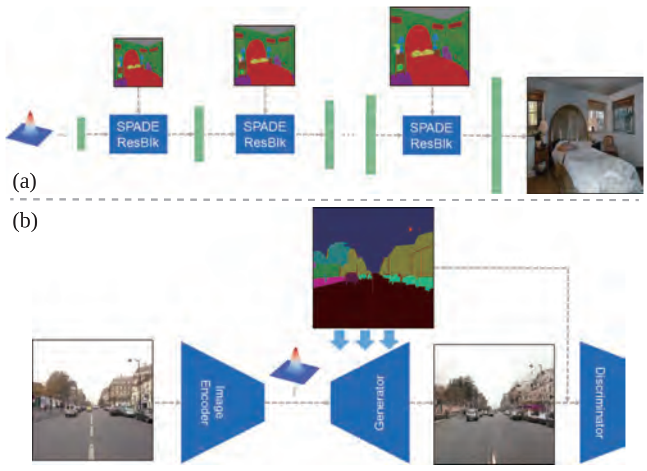

Figure 2: *Network architecture of the SPADE generator (a) and the training pipeline (b).*

where I is an indentity matrix.

At the training time shown in Figure 2 (b), given a ground-truth RGB image $\mathbf{I} \in \mathbf{R}^{3\times W\times H}$, the latent code $\mathbf{z}_1$ is sampled from the encoder E via the reparameterization trick [KW14]:

$$\mathbf{z}_1 \sim E(\mathbf{I}) = \mathcal{N}(\mu(\mathbf{I}), \sigma^2(\mathbf{I})), \tag{4}$$

$$\mu(\mathbf{I}) = f^{\mu}(e_L(e_{L-1}(\cdots e_1(\mathbf{I})))), \tag{5}$$

$$\sigma^2(\mathbf{I}) = f^{\sigma^2}(e_L(e_{L-1}(\cdots e_1(\mathbf{I})))), \tag{6}$$

where L denotes the number of encoder blocks, and e_l is the lth block comprising the convolutional, instance normalization, and LeakyReLU layers. The functions f^{μ} and f^{σ^2} are linear layers and compute the mean and the variance from the feature map.

To train the generator G and the encoder E, we minimize the following loss function:

$$\mathcal{L}(G,D,E) = \mathcal{L}_{GAN}(G,D,E) + \lambda_{FM}\mathcal{L}_{FM}(G,E) + \lambda_{VGG}\mathcal{L}_{VGG}(G,E) + \lambda_{KL}\mathcal{L}_{KL}(E), \tag{7}$$

$$G^*, E^* = \arg\min_{G,E}\max_{D} \mathcal{L}(G,D,E), \tag{8}$$

where D is the discriminator; $\mathcal{L}_{GAN}$, $\mathcal{L}_{FM}$, and $\mathcal{L}_{VGG}$ are adversarial [GPM*14], GAN feature matching [WLZ*18], and perceptual losses [JAF16], respectively. In addition, $\mathcal{L}_{KL}$ is the regularization term for the VAE:

$$\mathcal{L}_{KL}(E) = \mathbb{E}_{\mathbf{I}\sim p(\mathbf{I})}[\mathcal{D}_{KL}(E(\mathbf{I})\|\mathcal{N}(0,I))], \tag{9}$$

where $\mathcal{D}_{KL}$ is the Kullback-Leibler divergence (KLD), and λ_{FM}, λ_{VGG}, and λ_{KL} are the empirically determined weights.

3.2. Class- and Layer-wise VAEs

In addition to the VAE-GAN framework explained earlier, we introduce the class- and layer-wise style controls using multiple latent codes. Figure 3 illustrates our generator and the entire training pipeline. During training, our encoder extracts class-wise latent codes from the feature maps obtained from the final layer and from

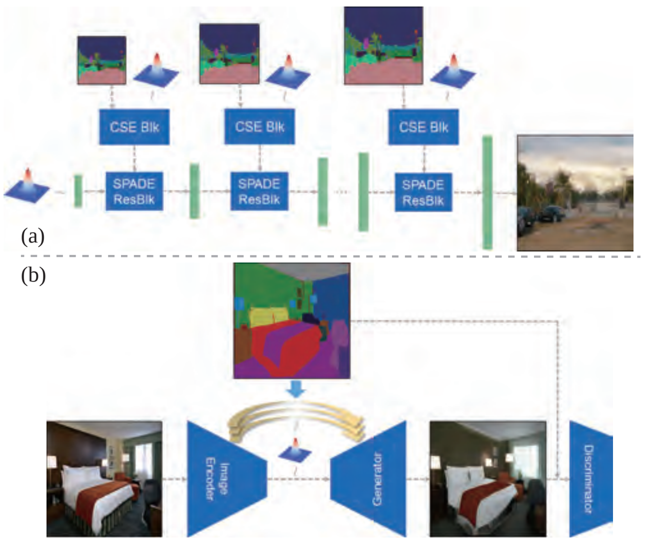

Figure 3: *Network architecture of our generator (a) and the training pipeline (b).*

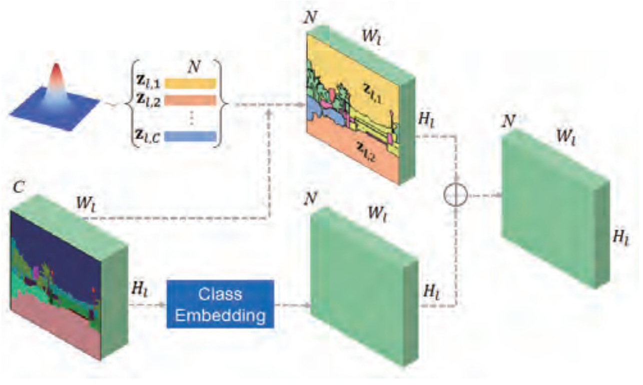

Figure 4: *Class-wise semantics embedding (CSE) block. In the upper path, we replace the one-hot vectors in the semantic mask with the corresponding latent codes. In the lower path, we embed the semantic mask in the same dimension as the counterpart in the upper path. The final feature map is obtained by their element-wise addition.*

each individual layer. Our generator receives latent codes at the first block and at each subsequent block, as shown in the bottom of Figure 3. Our *class-wise semantics embedding* (CSE) block (Figure 4) plays the key role in exploiting the class-wise latent codes at the generator blocks. We explain the generator and the encoder in the following sections.

3.2.1. Generator with CSE blocks

First, we define a set of latent codes fed to the generator as $\mathcal{Z} = \{\mathbf{z}_1, \mathbf{Z}_2, \mathbf{Z}_3 \cdots, \mathbf{Z}_L\}$, where $L \leq K$. Here, $\mathbf{z}_1 \in \mathbb{R}^{N_1}$ (where N_1 is the number of $\mathbf{z}_1$ channels) is a vector that controls the entire appearance of an output image, similar to the GauGAN-VAE [PLWZ19]. $\mathbf{Z}_l = (\mathbf{z}_{l,1}, \mathbf{z}_{l,2}, \cdots, \mathbf{z}_{l,C}) \in \mathbb{R}^{N \times C}$ (where N is also the number of channels) is a 2D tensor consisting of latent codes for C classes and is fed to the generator block l corresponding to the encoder's block l. The generator G gives an output image from a semantic mask $\mathbf{M}$ and a set of latent codes $\mathcal{Z}$:

$$G(\mathbf{M}, \mathcal{Z}) = g_K(s(\mathbf{M}_L, \mathbf{Z}_L), g_{K-1}(s(\mathbf{M}_{L-1}, \mathbf{Z}_{L-1}), \cdots g_1(\mathbf{z}_1))), \quad (10)$$

where s is the CSE block, which generates a feature map by combining a downsampled semantic mask $\mathbf{M}_l$ and the class-wise latent codes $\mathbf{Z}_l$. The feature maps computed in s are fed to the SPADE ResNet blocks. Similar to $\mathbf{z}_1$, $\{\mathbf{z}_{l,c}\}_{c \in \{1,2,\ldots,C\}}$ are sampled from the standard normal distribution at the test time:

$$\mathbf{z}_{l,c} \sim \mathcal{N}(\mathbf{0}, I). \quad (11)$$

In cases where the number of generator blocks K is larger than the number of encoder blocks L, latent codes for the $(K - L)$ generator blocks are missing. This holds true for our experiments because we used the same number of blocks as GauGAN (i.e., $L < K$) for fair comparison. In this case, we suggest two options: simply replacing s with $\mathbf{M}_l$ or reusing the same latent code at the $(K - L)$ blocks. In our experiments, we adopted a combination of these two options, as shown in the supplementary material.

Figure 4 shows the design of the CSE block. The feature map extracted by the CSE block is given as follows:

$$s(\mathbf{M}_l, \mathbf{Z}_l) = \mathcal{T}^{-1}(\mathbf{Z}_l \mathcal{T}(\mathbf{M}_l) + \mathbf{W}\mathcal{T}(\mathbf{M}_l)), \quad (12)$$

where $\mathcal{T}$ is a reshaping function, i.e., $\mathcal{T} : \mathbb{R}^{C \times W_l \times H_l} \to \mathbb{R}^{C \times W_l H_l}$. In the first term (the upper part of Figure 4), we replace the class-wise one-hot vectors with the corresponding class-wise latent codes. Note that the latent codes in the first term are random vectors and thus do not contain the objects' class information (e.g., chair and floor) that they represent. In the second term (the lower part of Figure 4), to balance with the first term's dense representation, we embed the sparse (i.e., one-hot) vectors in the semantic mask into dense vectors of the same dimension as the latent code by multiplying the trainable weight $\mathbf{W} \in \mathbb{R}^{N \times C}$. Exploiting the sparsity of $\mathbf{M}_l$'s one-hot representation, we can quickly calculate the matrix multiplications in the first and second terms. We can obtain the final feature map via element-wise summation of these two feature maps.

3.2.2. Encoder for learning latent codes

In this section, we describe the encoder for extracting latent codes for each class and layer. Figure 5 illustrates our encoder design. Recall that the GauGAN-VAE encoder samples the single latent code $\mathbf{z}_1$ dependent on the entire feature map at the final layer. In addition to that, our method samples a latent code $\mathbf{z}_{l,c}$ for each class c and each block l:

$$\mathbf{z}_{l,c} \sim E_{l,c}(\mathbf{I}) = \mathcal{N}(\mu_{l,c}(\mathbf{I}), \sigma^2_{l,c}(\mathbf{I})). \quad (13)$$

We compute the mean and the variance from the feature map of each block on a region $\Omega_{l,c}$ that contains the class label c in the semantic mask $\mathbf{M}_l$:

$$\mu_{l,c}(\mathbf{I}) = f^{\mu}_l(r(\Omega_{l,c}, e_{L-l+1}(e_{L-l}(\cdots e_1(\mathbf{I}))))), \quad (14)$$

$$\sigma^2_{l,c}(\mathbf{I}) = f^{\sigma^2}_l(r(\Omega_{l,c}, e_{L-l+1}(e_{L-l}(\cdots e_1(\mathbf{I}))))), \quad (15)$$

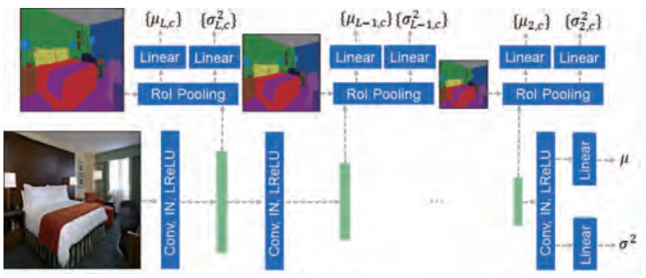

Figure 5: *Encoder architecture. The means and variances of the latent code distributions are calculated from a feature map at each block. The RoI pooling allows us to extract vectors with the same dimension from regions of different sizes.*

where f_l^{μ} and $f_l^{\sigma^2}$ are the linear functions for the block l. These linear functions are shared among the classes c in each block l when we sample the latent codes $\mathbf{z}_{l,c}$, as explained in Section 1. Although the number of pixels in the region $\Omega_{l,c}$ differs depending on class c, the extracted feature vectors for each class should have the same dimension. To manage this, we use a region-of-interest (RoI) pooling r that extracts a fixed-sized vector from a feature map $\mathbf{H}$ (3D tensor):

$$r(\Omega_{l,c}, \mathbf{H}) = (H'_{1,c}, H'_{2,c}, \cdots, H'_{j,c}, \cdots, H'_{n,c})^T, \quad (16)$$

$$H'_{j,c} = \max_{(x,y)\in\Omega_{l,c}} H_{j,x,y}, \quad (17)$$

where n, x, and y are the number of output channels, x coordinate and y coordinate in the region $\Omega_{l,c}$.

For training the networks, we use a modified version of Equation (7). Specifically, we redefine the regularization term $\mathcal{L}_{KL}$ to handle multiple outputs from our encoder:

$$\mathcal{L}_{KL}(E) = \mathbb{E}_{\mathbf{I},\mathbf{M}\sim p(\mathbf{I},\mathbf{M})}[\mathcal{D}_{KL}(E(\mathbf{I})\|\mathcal{N}(0,I)) + \frac{1}{|\mathcal{C}(\mathbf{M})|}\Sigma_{l=2}^{L}\Sigma_{c\in\mathcal{C}(\mathbf{M})}\mathcal{D}_{KL}(E_{l,c}(\mathbf{I})\|\mathcal{N}(0,I))], \quad (18)$$

where $\mathcal{C}(\mathbf{M})$ is a function that returns a set of unique classes existing in the semantic mask $\mathbf{M}$. The first term is for a latent code $\mathbf{z}_1$ that is dependent on the global appearance of the input image; the second term is for latent codes $\{\mathbf{Z}_{l,c}\}$ that are dependent on each class and each block. To prevent the regularization from being biased toward images containing many unique classes, we normalize the sum of the KLD with the number of unique classes. For the hyper-parameters, we used the same values as in the GauGAN-VAE [PLWZ19]; specifically, $\lambda_{FM} = 10$, $\lambda_{VGG} = 10$ and $\lambda_{KL} = 0.05$. The details of our network architectures can be found in the supplementary materials.

4. Experiments

4.1. Datasets

We used the following three datasets in our experiments.

- The *ADE20K* [ZZP*17a] dataset contains images of indoor and outdoor scenes labeled with 151 semantic classes (including the "unknown" class), such as window, bed, sky, and tree. It consists of 20,210 training images and 2,000 validation images. In the training set, we resized the images to 286×286 and applied random cropping of 256×256 for each iteration. In the validation set, we resized the images to 256×256.
- *DeepFashion* [LLQ*16] (In-shop Clothes Retrieval Benchmark) is a fashion-image dataset with 16 semantic classes (i.e., top, pants, hair, etc.). We excluded the photographs that do not contain face labels. Our training and test sets contained 6,343 and 6,390 images, respectively. We squared the images by padding them with white rectangles at the left and right sides and then resized them to 256×256. Note that we removed these paddings from the resultant images for better visual quality.
- The *GTA-5* [RVRK16] dataset includes rendered images of street scenes. Although Cityscapes is a popular street photograph dataset [COR*16], GTA-5 has much greater variations in terms of weather conditions and object appearance [LZM19]. We used the post-processed version of the GTA-5 dataset published in the recent work on multimodal semantic image synthesis [LZM19]. This dataset contains 12,403 training images and 6,382 test images with 20 semantic classes (19 classes that are common in GTA-5 and Cityscapes, plus an "others" class), including sky, road, and building. The image size is 512×256.

4.2. Experimental Settings

We implemented our method with PyTorch and conducted experiments on GeForce GTX 1080 Ti and Quadro RTX 6000. We used the same training parameters as in the GauGAN [PLWZ19] and GroupDNet papers [ZXYB20] except for the batch size. For optimization, we used Adam with the momentum term $\beta = (0.0, 0.9)$ and set the learning rates of the generator and encoder to 0.0004 and the discriminator to 0.0001. Due to our limited GPU resources, we set the batch size to 4 (the GauGAN paper uses a larger size). For the dimensions of the latent codes, we set $N_1 = 256$ and $N = C$ (where C is the number of classes in the dataset). In the ADE20K and DeepFashion datasets, we performed 100 and 60 epochs of training. These are followed by a further 100 and 40 epochs with linear decays to zero for the learning rate. In the GTA-5 dataset, 20 epochs of training were sufficient to learn plausible image translation. In addition, we applied random horizontal flipping to the images for all datasets during training.

4.2.1. Compared methods

We compared our method with the GauGAN [PLWZ19], GauGAN-VAE [PLWZ19], conditional implicit maximum likelihood estimation (cIMLE) [LZM19], and concurrent GroupDNet approaches [ZXYB20]. For GauGAN, we generated results using the model (published by the authors) pre-trained on ADE20K. For GauGAN-VAE, we performed the training with the same parameter settings as our method because their pre-trained models are not publicly available. For the cIMLE model, we generated results using the model pre-trained on GTA-5. To train our model and GauGAN-VAE using the GTA-5 dataset, we adopt the rebalancing schemes for the dataset and the loss function used in cIMLE, for fair comparison. For GroupDNet, we used the pre-trained model for the ADE20K dataset. However, we trained their model from scratch for the DeepFashion dataset because their pre-processed test set and

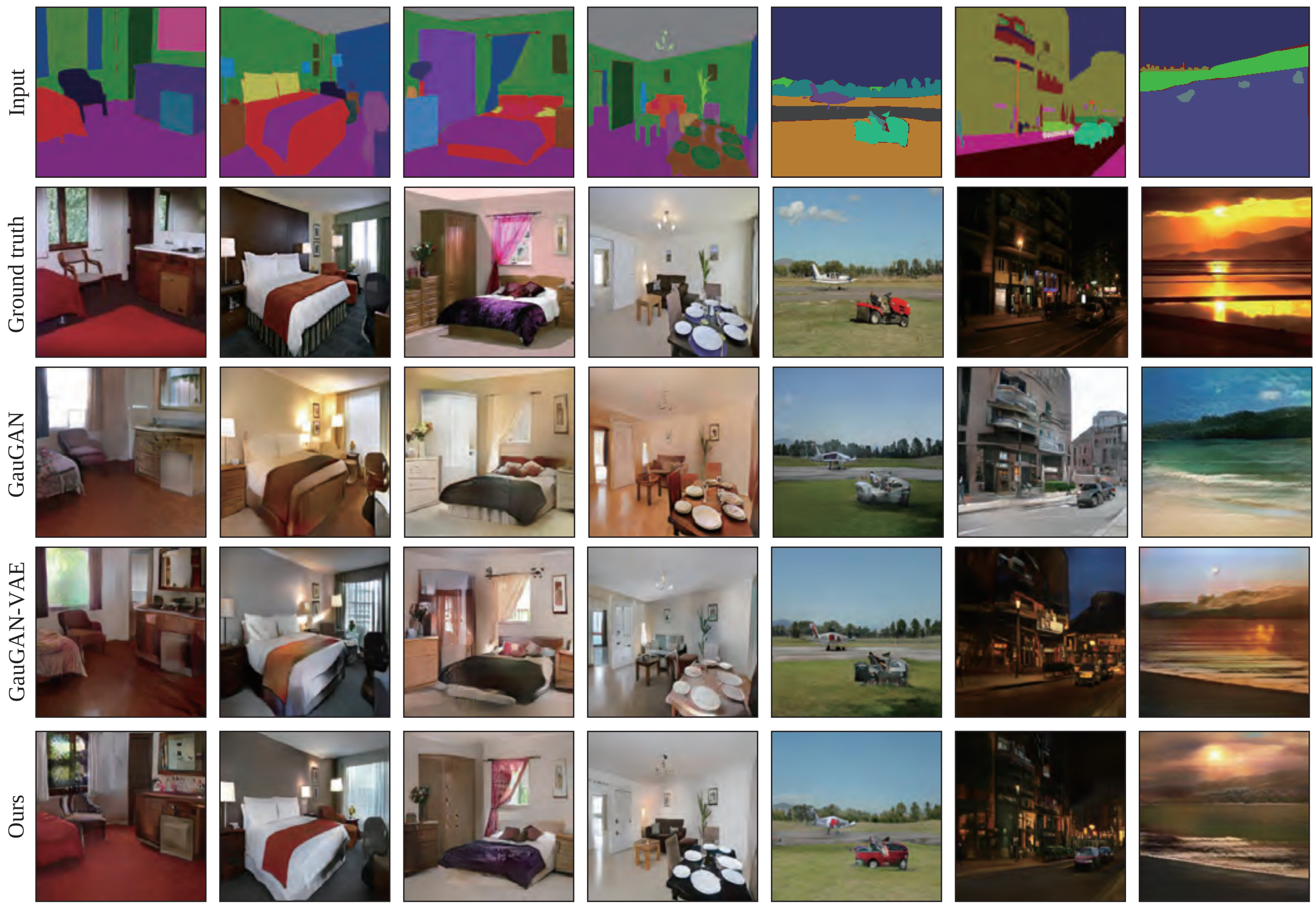

Figure 6: *Qualitative comparison of reconstruction performance using latent codes extracted from the ground-truth images in the ADE20K dataset. Our method can reproduce styles for each object in the ground-truth images more faithfully than the baseline methods. To see the differences more clearly, please zoom in on the electronic version of this paper.*

segmentation masks are not publicly available. The inputs for our results are not included in the training datasets.

4.3. Results and Analysis

In this section, we discuss the experimental results to answer two research questions (RQs) as follows.

RQ1. Can our model capture the style of each class using latent codes better?

If our model can adequately capture the class-wise styles, we should be able to reproduce the styles of a ground-truth image faithfully using the appropriate latent codes. To evaluate this, we used the trained encoder as a style guidance network to extract the "ground-truth" latent codes from the ground-truth images. We attempted to reproduce the ground-truth images in the test set by extracting the target styles from the images. The generated images $G(E(\mathbf{M}, \mathbf{I}))$ are compared with the ground-truth images $\mathbf{I}$. As an evaluation metric, we used the *Fréchet inception distance* (FID), which measures the distance between two image sets generated from a GAN distribution and a true distribution. Furthermore, to measure the similarity between the generated and ground-truth images one by one, we used the *learned perceptual image patch similarity* (LPIPS) [ZIE*18] and computed the average for all images in the test set (hereinafter called the *GT-similarity*).

Table 1: *Quantitative comparison of reconstruction performance using the latent codes extracted from the ground-truth images in the ADE20K dataset. The* ***red*** *and* ***blue*** *font colors denote the best and second-best scores, respectively.*

Methods	FID ↓	GT-similarity↓
GauGAN	33.9	0.538
GauGAN-VAE	**32.9**	**0.493**
Ours	**31.2**	**0.462**

Table 1 shows a quantitative comparison of various methods in the ADE20K dataset; the results show that our method yields better values in the two metrics. Because GauGAN [PLWZ19] does not

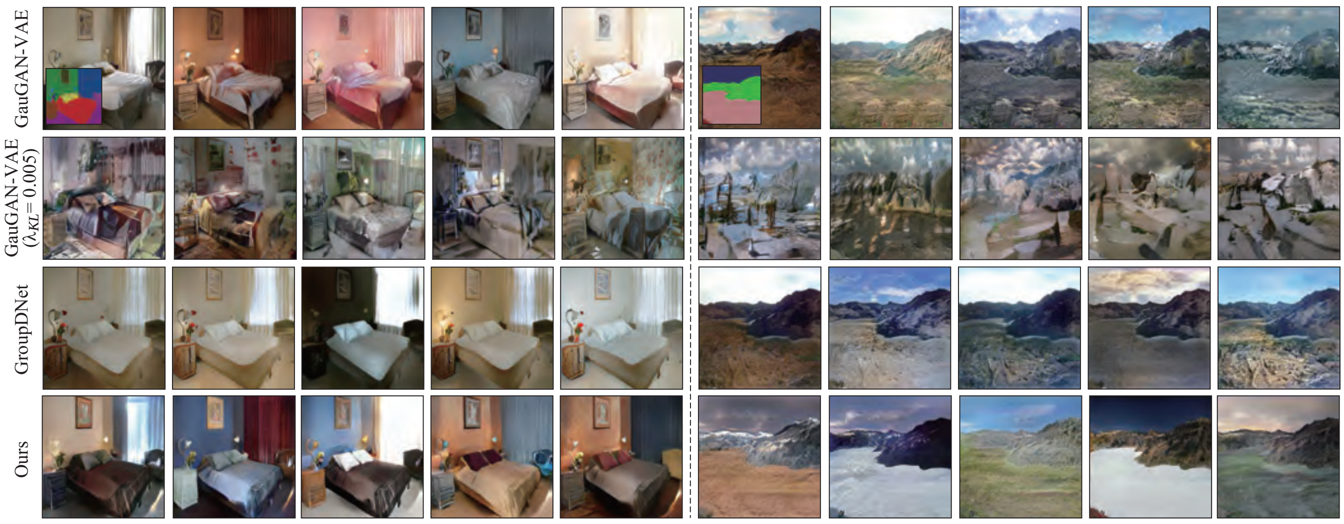

Figure 7: *Comparison of the results generated using the randomly-sampled latent codes with the ADE20K dataset. The inset at each upper left region is the input semantic mask. Our method can produce more diverse images than the baselines, without quality degradation. In GauGAN-VAE, the weak regularization ($\lambda_{KL} = 0.0005$) for latent codes increases the diversity of the generated images. However, it produces more artifacts due to the poor generalization performance.*

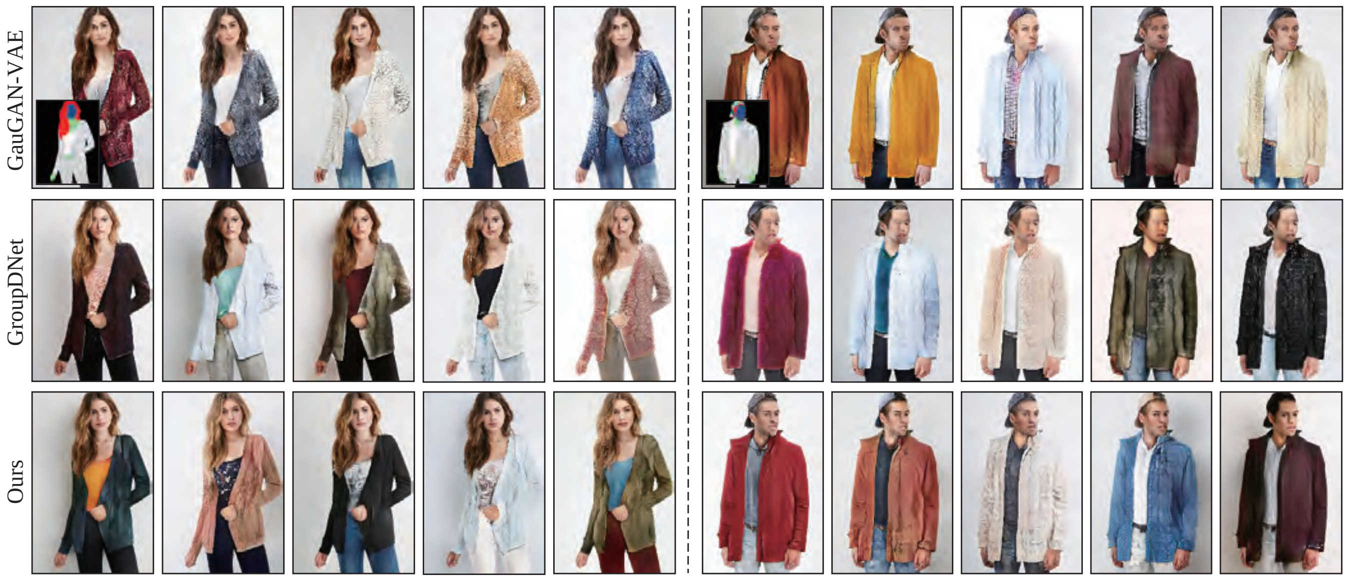

Figure 8: *Comparison of the results generated using the randomly-sampled latent codes with the DeepFashion dataset.*

accept user-specified styles, the GT-similarity score degrades even if the FID score is somewhat satisfactory. On the other hand, our method shows greater improvement in GT-similarity compared to GauGAN-VAE [PLWZ19]. These results indicate that our model can better capture the style of each class using latent codes. In the qualitative comparison shown in Figure 6, our results are visually more faithful to the ground-truth images than those of the compared methods. Specifically, individual objects have similar styles to the ground-truth images; for example, floor, wall, and bed images in the indoor scenes, and car, sky, and sunset in the outdoor scenes. Although the ground-truth images are not available in actual applications, our method can generate plausible images with specific styles if we have other reference images containing common classes (see Section 4.4.1).

RQ2. Can our model generate diverse and perceptually plausible images?

We evaluated the results generated using the latent codes randomly sampled from the prior without specifying specific style images. For diversity, we used the LPIPS score, following the work of Li

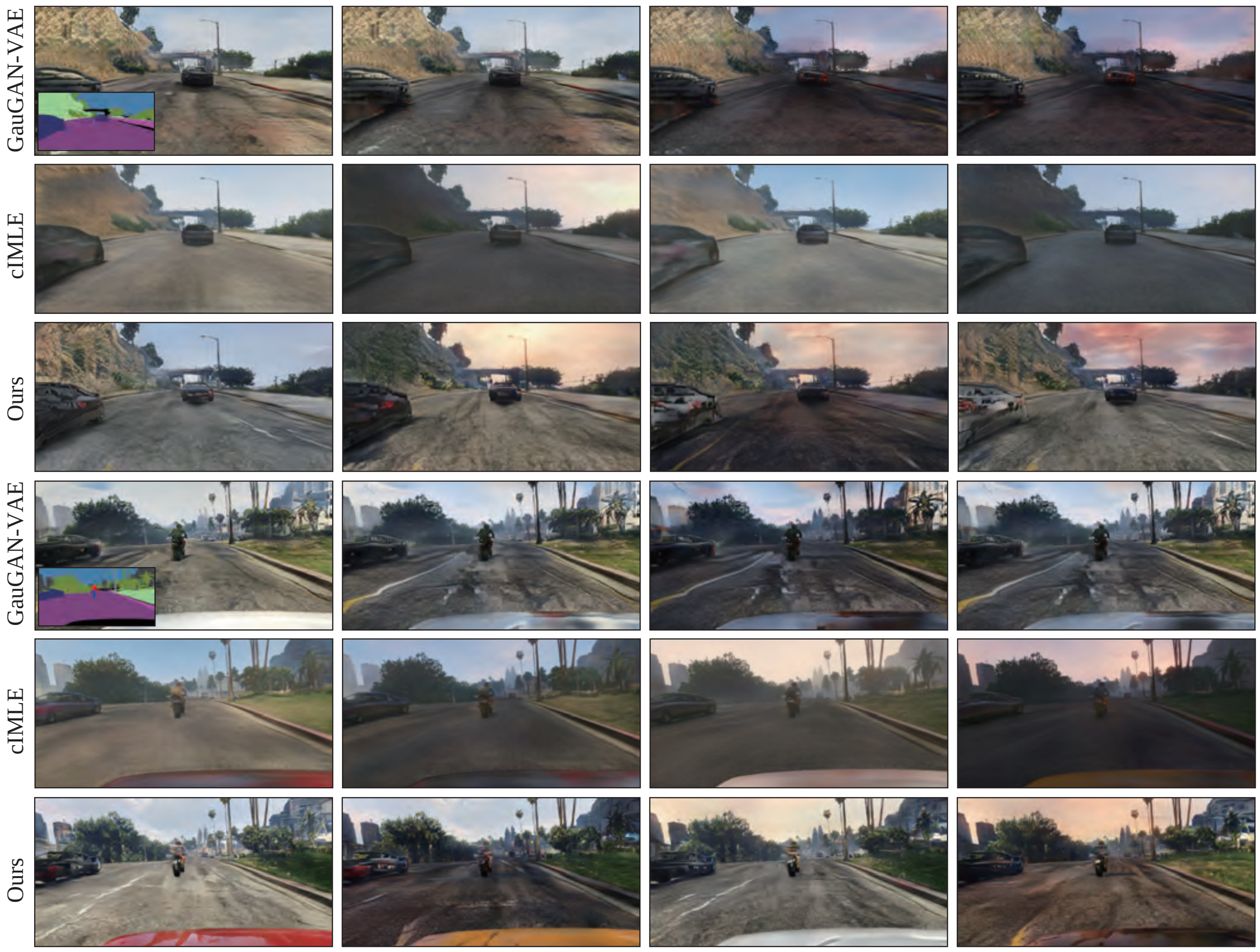

Figure 9: *Comparison of the results generated using the randomly-sampled latent codes with the GTA-5 dataset.*

Table 2: *Quantitative comparison of the results generated using the randomly-sampled latent codes with the ADE20K dataset. GauGAN-VAE* means GauGAN-VAE with weakened regularization ($\lambda_{KL} = 0.0005$) for latent codes.*

Methods	Diversity ↑	Top 1 GT-similarity ↓
GauGAN-VAE	0.294	**0.490**
GauGAN-VAE*	**0.507**	0.583
GroupDNet	0.177	0.521
Ours	**0.415**	**0.503**

Table 3: *Quantitative comparison of the results generated using the randomly-sampled latent codes with the DeepFashion dataset.*

Methods	Diversity ↑	Top 1 GT-similarity ↓
GauGAN-VAE	0.203	**0.155**
GroupDNet	**0.218**	0.159
Ours	**0.216**	**0.158**

et al. [LZM19]. We generated 40 images for each test image using randomly sampled latent codes. Next, we computed the LPIPS values for each pair in 40 images and took the average of them. By calculating this for all the test images and averaging them, we obtained the *Diversity* score; higher scores indicate greater diversity. As for visual plausibility, we can evaluate some results subjectively via qualitative comparison. However, objective evaluation is challenging because not all the randomly generated images have corresponding ground-truth images. To deal with this, we selected the image closest to the ground-truth image regarding the L_1 distance among the generated images, also following the work of Li et al.. For all test images, we computed the LPIPS scores between the selected images and the ground-truth images, and used the average of those top 1 similarities as an objective plausibility score (hereinafter called the *Top 1 GT-similarity*).

Table 2 shows quantitative results for the ADE20K dataset.

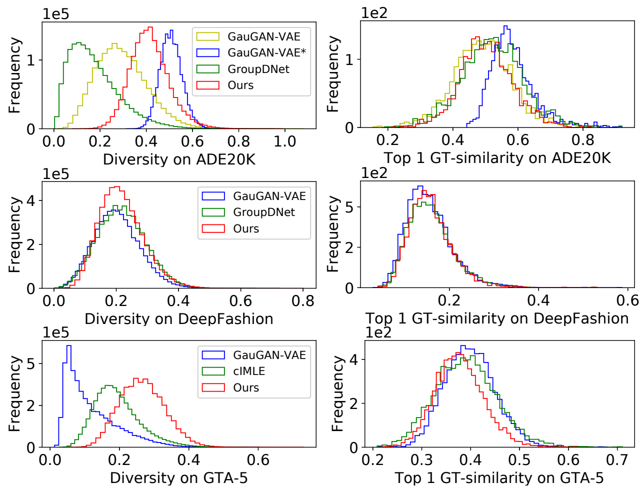

Figure 10: *Histograms of Diversity (larger is better) and Top 1 GT-similarity (smaller is better) on each dataset.*

Table 4: *Quantitative comparison of the results generated using the randomly-sampled latent codes with the GTA-5 dataset.*

Methods	Diversity ↑	Top 1 GT-similarity ↓
GauGAN-VAE	0.118	0.399
cIMLE	**0.194**	**0.390**
Ours	**0.266**	**0.378**

The Diversity scores indicate that our method can generate more diverse images than both GauGAN-VAE [PLWZ19] and GroupDNet [ZXYB20]. Conversely, there is no significant difference in Top 1 GT-similarity. We also tested GauGAN-VAE with weakened regularization of $\mathcal{L}_{KL}$ to make the results comparatively more diverse. Although the diversity was improved, the generated image degraded severely, as shown in the Top 1 GT-similarity score and qualitative results in Figure 7. This degradation is because of the large discrepancy between the prior and the distribution that the trained generator assumes. Furthermore, though GauGAN-VAE globally changes the appearance of the images, our method reproduces local and class-wise variations while retaining image quality. For the DeepFashion dataset (Table 3 and Figure 8), there is no significant difference between each method in terms of image quality. As for diversity, both GroupDNet and our model are modestly better than GauGAN-VAE. GauGAN-VAE can reproduce diverse appearances in large classes, such as jackets, but cannot handle small classes such as pants, inner shirts, and hair. GroupDNet obtained diverse appearances for each item, but it suffered from a large number of classes as demonstrated in the ADE20K results. By contrast, our method works well in both cases. In Table 4 and Figure 9, we also ran comparisons with cIMLE [LZM19] using the GTA-5 dataset. The results demonstrate that cIMLE can generate more diverse images than GauGAN-VAE. However, as shown in the qualitative results, cIMLE handles global color variations only. The results for cIMLE are also smooth, while the GAN-based methods give finer outputs. By contrast, our method showed good performance both quantitatively and qualitatively. To visualize the detailed statistical trends, we further show the histograms of evaluation scores in Figure 10. We can see that each distribution is unimodal, and our results are consistently better or comparable on average. We provide more samples for these results in the supplementary materials.

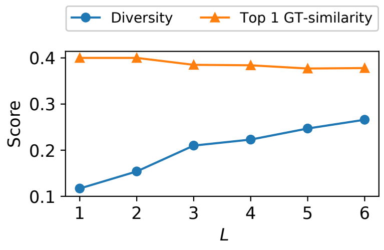

Figure 11: *Evaluation scores with different numbers of encoder layers L on the GTA-5 dataset.*

Table 5: *Ablation study on the GTA-5 dataset.*

Methods	Diversity ↑	Top 1 GT-similarity ↓
W/o class-wise VAE	0.131	0.390
W/o layer-wise VAE	0.242	0.378
Ours-50%	0.210	0.385
Ours-75%	0.247	0.377
Ours	0.266	0.378

4.3.1. Ablation study

We also conducted an ablation study to validate the effectiveness of our class-wise VAE and layer-wise VAE, respectively. Table 5 shows the evaluation scores of our method with and without our VAE modules. *W/o class-wise VAE* means that a single latent code is sampled from each layer, whereas *W/o layer-wise VAE* means that class-wise latent codes are sampled only from the middle layer. The results demonstrate that both VAEs are capable of diversifying generated images without degrading their quality. Compared to *Ours*, we can see that both of our VAE extensions help us diversify generated images without degrading their quality.

In Figure 11, we further investigated the effect of the number of encoder layers L, which is six in *Ours*. For $L < 6$, we adjusted the sizes of the feature maps before the final linear layer to the size of $L = 6$. *Ours-50%* and *Ours-75%* in Table 5 mean that 50% ($L = 3$) and 75% ($L = 5$) of the encoder layers were used. Generally, diversifying generated images often causes degradation of the image quality due to overfitting (like GauGAN-VAE* in Table 2). Nonetheless, these results indicate that increasing the number of layers makes generated images more diverse (i.e., higher Diversity) while keeping their quality (i.e., almost the same but a bit lower Top 1 GT-similarity). The computational cost increases approximately linearly according to the number of layers. For example, the training times for $L = 1$, 2, and 3 were 46, 55, and 66 minutes for one epoch.

4.4. Applications

In this section, we introduce three applications of image synthesis and editing based on our method.

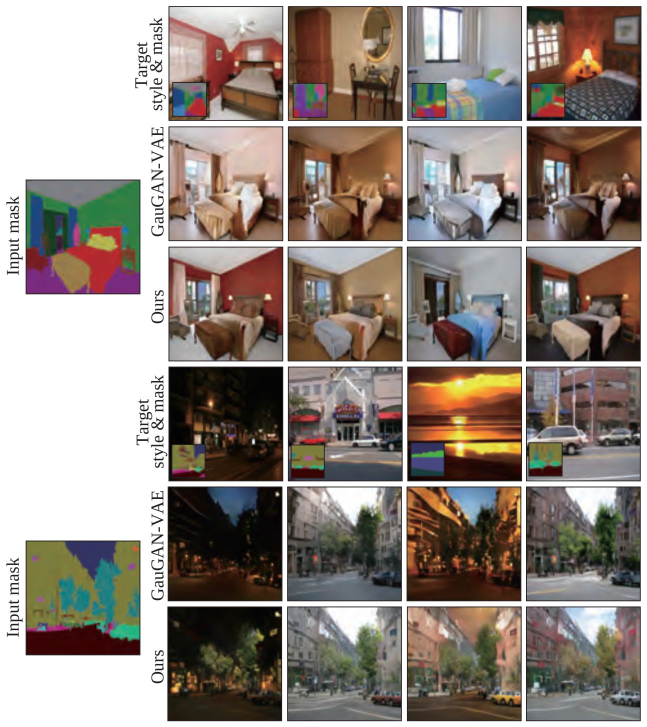

Figure 12: *Style guided synthesis in the ADE20K dataset. Our method can synthesize various images according to the target styles of each object using the target semantic masks (insets). Our results are more faithful to the target styles than the baseline methods.*

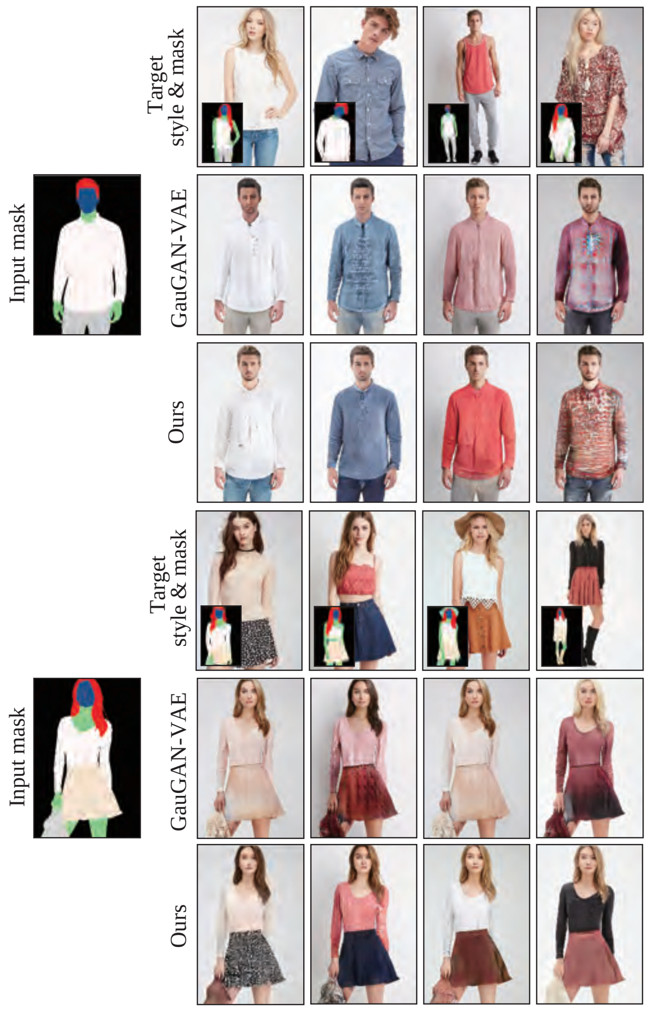

Figure 13: *Style guided synthesis in the DeepFashion dataset.*

4.4.1. Style guided synthesis

From a source mask $\mathbf{M}_s$, our method can synthesize an image in reference to the styles extracted from an arbitrary target image $\mathbf{I}_t$ using the pre-trained encoder. Specifically, we take the latent codes from $E_{l,c}(\mathbf{I}_t)$ for the classes that appear both in the source and target masks $\mathbf{M}_s$ and $\mathbf{M}_t$; i.e., $\mathcal{C}(\mathbf{M}_s) \cap \mathcal{C}(\mathbf{M}_t)$. In addition, we take random samples from $\mathcal{N}(\mathbf{0}, I)$ for the $\mathbf{M}_s$ classes that are not included in $\mathbf{M}_t$; i.e., $\mathcal{C}(\mathbf{M}_s) \cap \overline{\mathcal{C}(\mathbf{M}_t)}$. The results are shown in Figures 12 and 13. Our method reproduces the styles of the individual objects such as walls, sheets, clothes, and hair more faithfully than GauGAN-VAE. This application is similar to image analogies [HJO*01], which applies a transformation between a mask A and an image A' to a mask B. Our method differs from this technique in that classes included in the mask B are not necessarily included in the mask A. Our method can handle missing classes via latent codes sampled from the prior.

4.4.2. Object editing

As a design exploration tool, our method is useful for object appearance editing by manipulating the latent codes for a particular class. The bottom of Figure 1 and the top three rows of Figure 14 show the appearances of pants, hair, etc. edited individually while retaining the styles of the other classes. To maintain a person's identity, we preserved the original faces and skin by copying the pixel values from the input photographs. The bottom of Figure 14 shows that the different latent codes generate the various floor patterns, such as marble or wood, with different glossiness. A recent study by Bau et al. [BSP*19] also attempted a similar task. Their method can synthesize photorealistic images by optimizing latent representation based on the image prior learned by the GANs. While this optimization requires tens of seconds, our method does not require such optimization.

4.4.3. Material editing

We can also edit object materials by manipulating the layer-wise latent codes. In Figure 15, it is evident that moving the latent codes fed to the shallow layer changes the global appearance, e.g., the color of an entire object. Conversely, moving the latent codes fed to the deep layer changes the local appearance features, such as edges and textures. However, the latent codes for some layers did not have as much effect on the appearance variations as we expected. The most likely reason for this is that the degree of the freedom of latent codes was too large. In future work, we would like to determine the

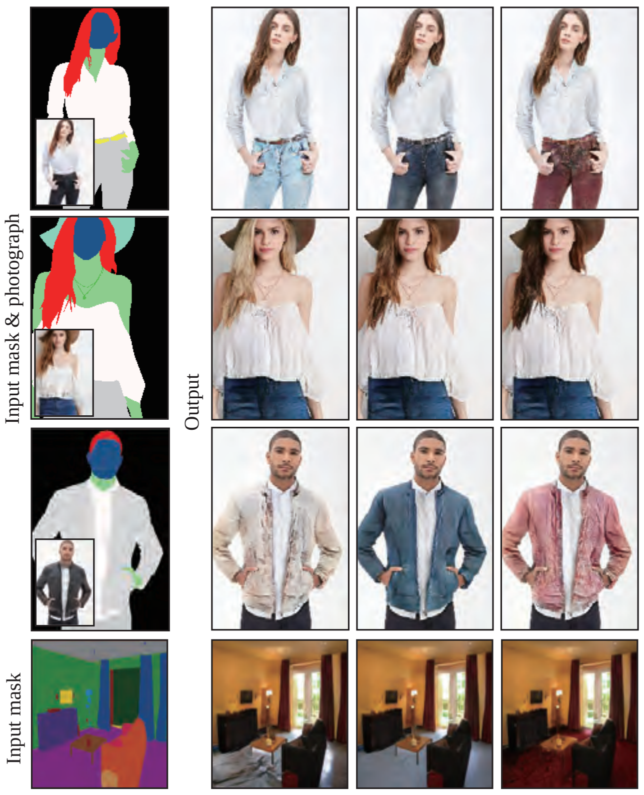

Figure 14: *Object editing. In the top three rows, we changed the regions of specific classes, such as clothes and hair using randomly-sampled latent codes while preserving the other regions. In the bottom row, we generated the image from the semantic mask and edited the floor design using the latent codes.*

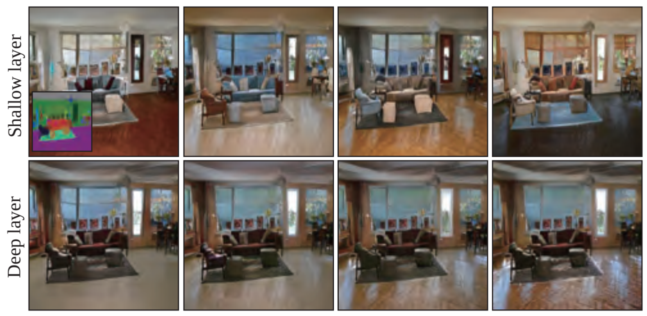

Figure 15: *Material editing. We can control the global appearance (e.g., color of entire objects) by moving the latent codes at the shallow layer in the generator. Likewise, we can control the local appearance (e.g., edges and textures) by moving the latent codes at the deeper layers.*

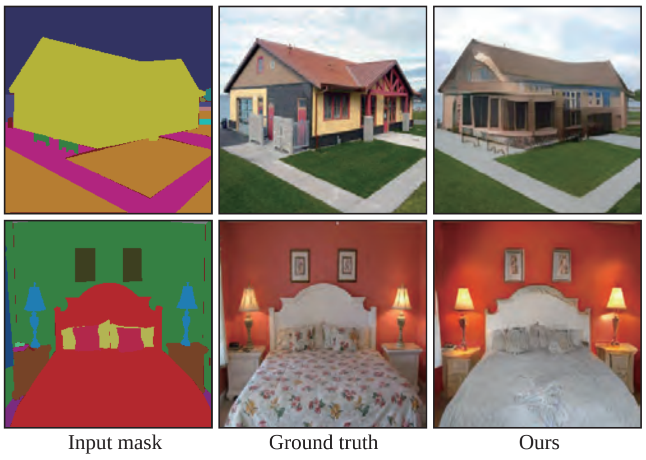

Figure 16: *Our failure cases. Faithfully reproducing complex structures and textures is difficult even with the latent codes extracted from the ground-truth images.*

appropriate dimensions of the latent codes automatically according to the data variation in a target domain.

4.5. Limitations

Although our method significantly improves the diversity of synthesized images while maintaining generalization performance, some limitations remain. First, there is room for improvement in style reproducibility. Figure 16 shows the images reconstructed using the latent codes extracted from the ground-truth images. The results show that complex patterns, such as the window position and the bed texture, are not completely reproduced. Simply increasing the dimensions of the latent codes would improve the expressiveness of the model but could worsen the generalization performance due to overfitting. To solve this problem, we seek a more efficient disentanglement representation in future work. Nevertheless, our method makes a significant contribution by diversifying semantic image synthesis while avoiding output degradation.

Another limitation of our method arises from our assumption that each latent code is independent for each class and layer. This assumption can sometimes cause context mismatch in the object appearance. For example, as shown in the third row and fourth column in Figure 9, it is possible that users might expect the road to be redder in color due to the effect of the sunset. Although our method recognizes scene contexts using the large receptive fields of the decoder, these results might not be sufficient. Users can optionally repeat re-sampling of latent codes until he/she is satisfied, but we would like to develop a better approach that exploits the relationship among latent codes.

5. Conclusions

In this study, we have proposed a method for semantic image synthesis and editing via class- and layer-wise VAEs to diversify output images. Our network design allows for the acquisition of latent codes depending on semantic classes and layers. Furthermore, our method is a simple yet effective approach to improve the expressivity of the model and capture diverse styles by learning multiple latent spaces. Through experiments with three challenging datasets, we have demonstrated that our method can synthesize images that are both plausible and diverse from a simple semantic mask. We have also shown applications of our method for style guided synthesis, object editing, and material editing. Future work includes the adaptive adjustment of the appropriate degrees of freedom of the latent codes. In addition, we are interested in obtaining efficient disentanglement representation that considers the correlation between classes.

References

[BSP*19] Bau, David, Strobelt, Hendrik, Peebles, William S., et al. "Semantic photo manipulation with a generative image prior". *ACM Trans. Graph.* 38.4 (2019), 59:1–59:11 2, 10.

[CCK*18] Choi, Yunjey, Choi, Min-Je, Kim, Munyoung, et al. "StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation". *CVPR 2018*. 2018, 8789–8797 2.

[CCT*09] Chen, Tao, Cheng, Ming-Ming, Tan, Ping, et al. "Sketch2Photo: internet image montage". *ACM Trans. Graph.* 28.5 (2009), 124 2.

[CDH*16] Chen, Xi, Duan, Yan, Houthooft, Rein, et al. "InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets". *NIPS 2016*. 2016, 2172–2180 2.

[CK17] Chen, Qifeng and Koltun, Vladlen. "Photographic Image Synthesis with Cascaded Refinement Networks". *ICCV 2017*. 2017, 1520–1529 2.

[COR*16] Cordts, Marius, Omran, Mohamed, Ramos, Sebastian, et al. "The Cityscapes Dataset for Semantic Urban Scene Understanding". *CVPR 2016*. 2016, 3213–3223 5.

[DKD17] Donahue, Jeff, Krähenbühl, Philipp, and Darrell, Trevor. "Adversarial Feature Learning". *ICLR 2017*. 2017 2.

[EKD*17] Eilertsen, Gabriel, Kronander, Joel, Denes, Gyorgy, et al. "HDR image reconstruction from a single exposure using deep CNNs". *ACM Trans. Graph.* 36.6 (2017), 178:1–178:15 2.

[EKM17] Endo, Yuki, Kanamori, Yoshihiro, and Mitani, Jun. "Deep reverse tone mapping". *ACM Trans. Graph.* 36.6 (2017), 177:1–177:10 2.

[GKN*18] Ghosh, Arnab, Kulharia, Viveka, Namboodiri, Vinay P., et al. "Multi-Agent Diverse Generative Adversarial Networks". *CVPR 2018*. 2018, 8513–8521 2.

[GPM*14] Goodfellow, Ian J., Pouget-Abadie, Jean, Mirza, Mehdi, et al. "Generative Adversarial Nets". *NIPS 2014*. 2014, 2672–2680 2, 3.

[HB17] Huang, Xun and Belongie, Serge J. "Arbitrary Style Transfer in Real-Time with Adaptive Instance Normalization". *ICCV 2017*. 2017, 1510–1519 2.

[HE07] Hays, James and Efros, Alexei A. "Scene completion using millions of photographs". *ACM Trans. Graph.* 26.3 (2007), 4 2.

[HJO*01] Hertzmann, Aaron, Jacobs, Charles E., Oliver, Nuria, et al. "Image analogies". *SIGGRAPH 2001*. Ed. by Pocock, Lynn. 2001, 327–340 2, 10.

[IZZE17] Isola, Phillip, Zhu, Jun-Yan, Zhou, Tinghui, and Efros, Alexei A. "Image-to-Image Translation with Conditional Adversarial Networks". *CVPR 2017*. 2017, 5967–5976 2.

[JAF16] Johnson, Justin, Alahi, Alexandre, and Fei-Fei, Li. "Perceptual Losses for Real-Time Style Transfer and Super-Resolution". *ECCV 2016*. Ed. by Leibe, Bastian, Matas, Jiri, Sebe, Nicu, and Welling, Max. Lecture Notes in Computer Science. 2016, 694–711 3.

[KE18] Kanamori, Yoshihiro and Endo, Yuki. "Relighting humans: occlusion-aware inverse rendering for full-body human images". *ACM Trans. Graph.* 37.6 (2018), 270:1–270:11 2.

[KLA*19] Karras, Tero, Laine, Samuli, Aittala, Miika, et al. "Analyzing and Improving the Image Quality of StyleGAN". *CoRR* abs/1912.04958 (2019). arXiv: `1912.04958` 2.

[KLA19] Karras, Tero, Laine, Samuli, and Aila, Timo. "A Style-Based Generator Architecture for Generative Adversarial Networks". *CVPR 2019*. 2019, 4401–4410 2.

[KW14] Kingma, Diederik P. and Welling, Max. "Auto-Encoding Variational Bayes". *ICLR 2014*. 2014 2, 3.

[LHE*07] Lalonde, Jean-François, Hoiem, Derek, Efros, Alexei A., et al. "Photo clip art". *ACM Trans. Graph.* 26.3 (2007), 3 2.

[LLQ*16] Liu, Ziwei, Luo, Ping, Qiu, Shi, et al. "DeepFashion: Powering Robust Clothes Recognition and Retrieval with Rich Annotations". *CVPR 2016*. 2016 2, 5.

[LM18] Li, Ke and Malik, Jitendra. "Implicit Maximum Likelihood Estimation". *CoRR* abs/1809.09087 (2018). arXiv: `1809.09087` 2.

[LSLW16] Larsen, Anders Boesen Lindbo, Sønderby, Søren Kaae, Larochelle, Hugo, and Winther, Ole. "Autoencoding beyond pixels using a learned similarity metric". *ICML 2016*. JMLR Workshop and Conference Proceedings. 2016, 1558–1566 2.

[LYS*19] Liu, Xihui, Yin, Guojun, Shao, Jing, et al. "Learning to Predict Layout-to-image Conditional Convolutions for Semantic Image Synthesis". *NeurIPS 2019*. 2019, 568–578 2.

[LZM19] Li, Ke, Zhang, Tianhao, and Malik, Jitendra. "Diverse Image Synthesis From Semantic Layouts via Conditional IMLE". *ICCV 2019*. 2019, 4219–4228 2, 5, 8, 9.

[PHS*18] Portenier, Tiziano, Hu, Qiyang, Szabó, Attila, et al. "Faceshop: deep sketch-based face image editing". *ACM Trans. Graph.* 37.4 (2018), 99:1–99:13 2.

[PLWZ19] Park, Taesung, Liu, Ming-Yu, Wang, Ting-Chun, and Zhu, Jun-Yan. "Semantic Image Synthesis With Spatially-Adaptive Normalization". *CVPR 2019*. 2019, 2337–2346 2–7, 9.

[RVRK16] Richter, Stephan R., Vineet, Vibhav, Roth, Stefan, and Koltun, Vladlen. "Playing for Data: Ground Truth from Computer Games". *ECCV 2016*. Lecture Notes in Computer Science. 2016, 102–118 2, 5.

[SBT*19] Sun, Tiancheng, Barron, Jonathan T., Tsai, Yun-Ta, et al. "Single image portrait relighting". *ACM Trans. Graph.* 38.4 (2019), 79:1–79:12 2.

[WLZ*18] Wang, Ting-Chun, Liu, Ming-Yu, Zhu, Jun-Yan, et al. "High-Resolution Image Synthesis and Semantic Manipulation With Conditional GANs". *CVPR 2018*. 2018, 8798–8807 2, 3.

[ZIE*18] Zhang, Richard, Isola, Phillip, Efros, Alexei A., et al. "The Unreasonable Effectiveness of Deep Features as a Perceptual Metric". *CVPR 2018*. 2018, 586–595 6.

[ZLW*18] Zhang, Lvmin, Li, Chengze, Wong, Tien-Tsin, et al. "Two-stage sketch colorization". *ACM Trans. Graph.* 37.6 (2018), 261:1–261:14 2.

[ZPIE17] Zhu, Jun-Yan, Park, Taesung, Isola, Phillip, and Efros, Alexei A. "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks". *ICCV 2017*. 2017, 2242–2251 2.

[ZXYB20] Zhu, Zhen, Xu, Zhiliang, You, Ansheng, and Bai, Xiang. "Semantically Multi-modal Image Synthesis". *CVPR 2020*. 2020 3, 5, 9.

[ZZP*17a] Zhou, Bolei, Zhao, Hang, Puig, Xavier, et al. "Scene Parsing through ADE20K Dataset". *CVPR 2017*. 2017, 5122–5130 2, 5.

[ZZP*17b] Zhu, Jun-Yan, Zhang, Richard, Pathak, Deepak, et al. "Toward Multimodal Image-to-Image Translation". *NIPS 2017*. 2017, 465–476 2.