CGI2022 manuscript No.
(will be inserted by the editor)

Controlling StyleGANs Using Rough Scribbles via One-shot Learning

Yuki Endo · Yoshihiro Kanamori
University of Tsukuba

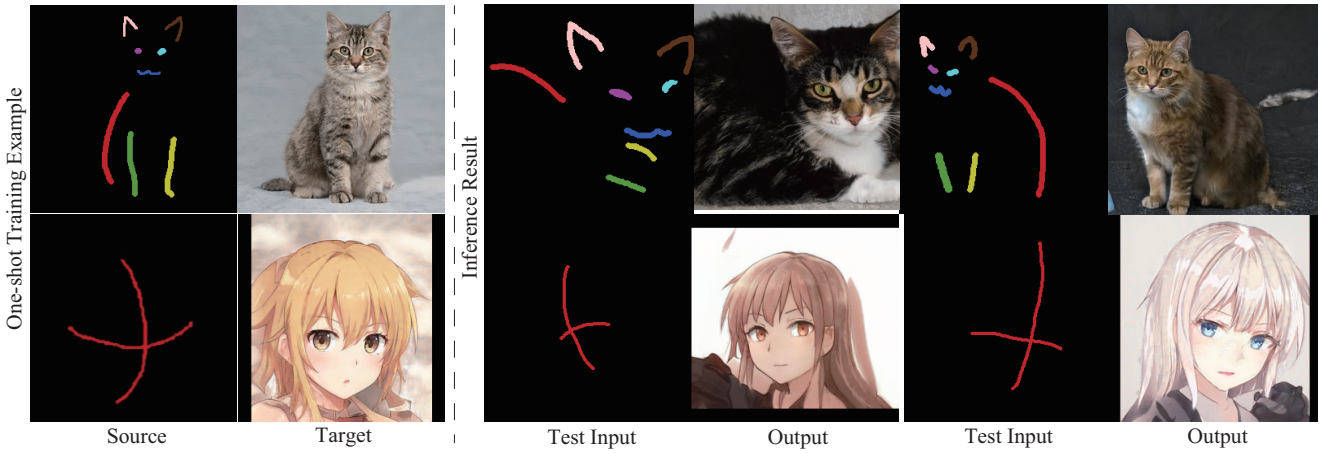

Fig. 1: Our method can synthesize photorealistic images from rough semantic scribbles using a single training pair and a pre-trained StyleGAN model.

Abstract This paper tackles the challenging problem of one-shot semantic image synthesis from rough sparse annotations, which we call "*semantic scribbles*." Namely, from only a single training pair annotated with semantic scribbles, we generate realistic and diverse images with layout control over, e.g., facial part layouts and body poses. We present a training strategy that performs pseudo labeling for semantic scribbles using the StyleGAN prior. Our key idea is to construct a simple mapping between StyleGAN features and each semantic class from a single example of semantic scribbles. With such mappings, we can generate an unlimited number of pseudo semantic scribbles from random noise to train an encoder for controlling a pre-trained StyleGAN generator. Even with our rough pseudo semantic scribbles obtained via one-shot supervision, our method can synthesize high-quality images thanks to our GAN inversion framework. We further offer optimization-based post-processing to refine the pixel alignment of synthesized images. Qualitative and quantitative results on various datasets demonstrate improvement over previous approaches in one-shot settings.

Keywords Generative adversarial networks · Image editing · GAN inversion

1 Introduction

Recent advances in generative adversarial networks (GANs) have enabled us to easily create realistic and diverse images [5,18,19]. This success, in turn, has encouraged researchers to find how to control GANs' outputs [8,14,29,30,37]. They have analyzed the latent space of GANs to reveal the effect of latent code manipulation on output images. Through the disentangled latent space, users can control various attributes

(e.g., face orientation, mouth shape, and hair color) in the output images. However, attribute manipulation is not necessarily intuitive against, in particular, layout-related attributes (e.g., facial part layout and body pose), which we focus on in this paper.

Layout control in image synthesis is possible with image-to-image (I2I) translation [13], where layouts are specified with semantic masks or sketches. A drawback is that most existing techniques require substantial training data in source and target domains for high-quality outputs. Even worse, fine-grained annotations of pixel-wise labels in both training and testing times are quite costly. Although some public datasets like those for human faces may be sufficient to train I2I models, various datasets in other domains such as for animals and cartoons are not well organized.

In this paper, we present the first method for controlling GANs' outputs via one-shot learning in a one using rough sparse annotations as input. We term these input annotations "*semantic scribbles*," and they can be used to specify output layouts with sparse annotations like brush strokes for body parts and cross lines for the face. Imagine that you have a large dataset of unlabeled images, but only a single annotated pair is available (see Figure 1). In this scenario, we would utilize *StyleGAN* [18,19], pre-trained using the unlabeled dataset. Namely, we would achieve high-quality image synthesis by exploring StyleGAN's latent space via *GAN inversion* [38]. What is challenging here is that, although common GAN inversion techniques [1,2] assume that test inputs belong to the same domain as a GAN's training data (e.g., facial photographs), our test and training data are in different domains, i.e., semantic layouts and photographs. How to invert the input in a different domain into a GAN's latent space is an open question, especially in a one-shot scenario.

To bridge the domain gaps, we construct a mapping between the semantics predefined in the one-shot example and StyleGAN's latent space. Inspired by the fact that pixels with the same semantics tend to have similar StyleGAN features [9], we generate pseudo semantic scribbles from random noise in StyleGAN's latent space via simple nearest-neighbor matching. This way, we can draw an unlimited number of training pairs by only feeding random noise to a pre-trained StyleGAN generator. After integrating an encoder on top of a fixed StyleGAN generator, we then train the encoder for controlling the generator using the pseudo-labeled data in a supervised fashion. We further offer optimization-based post-processing to refine the pixel alignment of synthesized images. Our approach integrates semantic layout control into pre-trained StyleGAN models publicly available on the Web [25], via pseudo labeling even from a single annotated pair.

In summary, our major contributions are as follows:

- We explore a novel problem of controlling GANs' outputs using semantic scribbles in a one-shot setting, where users can synthesize high-quality, various images in target domains even from a single and rough semantic layout provided during training.
- We propose a simple yet effective framework for training a StyleGAN encoder for scribble-based image synthesis in a one-shot scenario, via pseudo sampling and labeling based on the StyleGAN prior.
- We propose a post-processing method, which is optimization-based GAN inversion that refines our encoder-based results.

As demonstrated in Figure 1 and the experiments in Section 4, our method is the first to control StyleGAN in a one-shot scenario using semantic scribbles, which can handle various layouts such as complicated poses as well as face orientation.

2 Related Work

This section introduces related work for latent space manipulation and image-to-image translation. We also discuss one- or few-shot approaches for controlling GANs.

2.1 Latent space manipulation

Recent techniques attempt to manipulate disentangled latent spaces of pre-trained GANs for image editing. Here, we briefly introduce some of them; please refer to a survey paper [38] for more information. A typical choice for pre-trained GANs is StyleGAN [18,19], which enables coarse-to-fine editing using multiple layer-wise latent codes. Chiu et al. developed a framework that allows users to search 1D subspaces for efficient user exploration from a high-dimensional latent space [8]. Jahanian et al. computed trajectories in a latent space for simple image transformations in a self-supervised manner [14]. InterFaceGAN [29] can control the pose and expression of faces by finding semantic boundaries via the training of a linear SVM. Shen et al. and Härkönen et al. found interpretable paths in a latent space through closed-form analysis [30] and principal components analysis (PCA) [12]. Although these methods enable image editing via operation in a latent space, users cannot directly control output layouts.

Image2StyleGAN [1,2] is a GAN inversion method that can control GANs' outputs by inverting given images into latent space via optimization. Roich et al. [27]

proposed a technique that not only optimizes latent codes but also tunes a generator to improve editability of inverted real images. However, inverting semantic layouts into a latent space defined by photographs is not straightforward because how to measure the discrepancy between the two different domains (i.e., semantic scribbles and photographs) is an open question, especially in a one-shot scenario [38]. Recently, the *Pixel2Style2Pixel* (pSp) [26] encoder has enabled GAN inversion without optimization for test inputs. Because this method does not need to compute losses between inputs and a GAN's outputs, it can also handle semantic layouts as input. To improve editability for the encoder-based approach, Tov et al. [34] introduced regularization and adversarial losses for latent codes into encoder training. In addition, Alaluf et al. [3] proposed a method that improves reconstruction quality of inverted images by iteratively refining latent codes from the encoder. However, the encoder-based approach requires many training pairs to improve the generalizability of the encoder, as demonstrated in Section 4.

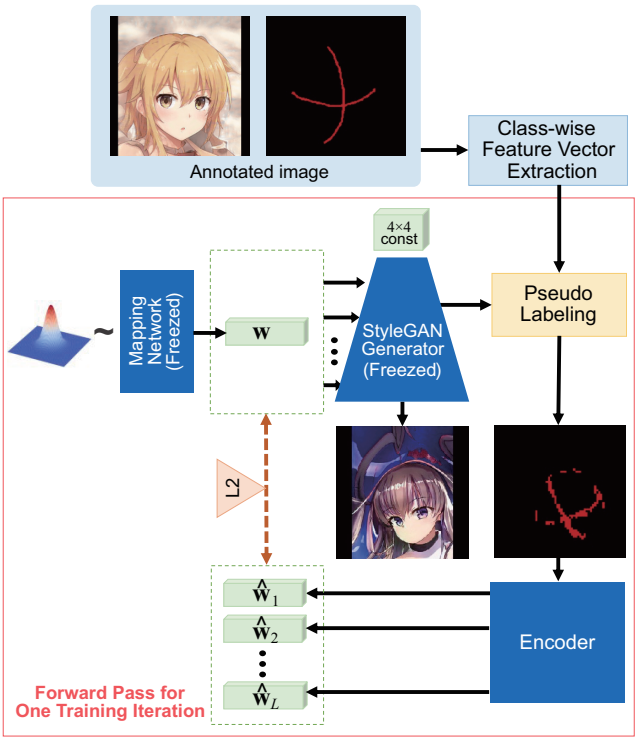

Fig. 2: Training iteration of the StyleGAN encoder. We first generate images from noise vectors via the mapping network and the StyleGAN generator. We then compute pseudo semantic scribbles using *class-wise feature vectors*. We train the encoder based on L2 loss between latent codes.

2.2 Image-to-image translation

There are various I2I translation methods suitable for interactive image synthesis using semantic layouts as input; the goals are, e.g., to improve image quality [13,6, 23,24,33,32], generate multi-modal outputs [24,21,45, 10,41], and simplify input annotations using bounding boxes [44,31,22]. There are also image synthesis methods using sketch and color scribbles as input [28,15]. However, these methods require a large amount of training data for both source and target domains and thus are unsuitable for our one-shot scenario. Moreover, although fine-grained annotations allow us to specify fine details, the annotation cost is very high. Meanwhile, our focus is on roughly specifying the layouts of GANs' outputs using semantic scribbles.

There are also methods that use sketches as input instead of semantic masks [39]. SketchyGAN [7] is a variant of GANs that can generate images for various classes from freehand sketches. SketchyCOCO [11] enabled image generation for more complicated scenes. While these methods require a large amount of paired data for training the networks, Wang et al. [36] proposed a method that can generate diverse images by learning a small number of sketches. However, this method is not suitable for interactive editing because it requires re-training the model for about 30K iterations every time the test input changes. Our method enables interactive synthesis for various test inputs after training our model. It can also handle novel and arbitrary annotations such as cross lines.

2.3 One-/Few-shot GAN control

To the best of our knowledge, there is no other one-shot method dedicated to controlling GANs' outputs using semantic layouts. Meanwhile, several few-shot techniques were proposed recently. DatasetGAN [43] and RepurposeGAN [35] demonstrated high-quality semantic segmentation by leveraging pre-trained StyleGAN models and a few annotations. They use feature maps from the hidden layers of StyleGAN to learn segmentation models. However, they are not suited for the class-imbalanced setting in our one-shot scenario, where only sparse sets of pixels are labeled with scribbles. SemanticGAN [20] trains GANs that generate both images and semantic masks using many unlabeled images and dozens of segmentation maps. Although this method can generate high-quality segmentation maps for out-of-domain images, it does not work well if we have only one-shot scribbles for training. Therefore, these techniques are not suitable to replace our pseudo labeling in our framework, as demonstrated in Section 4.

3 Method

Our goal is to accomplish scribble-based image synthesis via training with N_u unlabeled images and a single labeled image both in the same target domain. A single training pair consists of one-hot semantic scribbles $\mathbf{x} \in \{0,1\}^{C\times W\times H}$ (where C, W, and H are the number of classes, width, and height) and its ground-truth (GT) RGB image $\mathbf{y} \in \mathbb{R}^{3\times W\times H}$. Each scribble has a unique class label, whereas unoccupied pixels have an "*unknown*" class label. Hereafter, we denote the labeled dataset as $\mathcal{D}_l = \{\mathbf{x}, \mathbf{y}\}$ and the unlabeled dataset as $\mathcal{D}_u = \{\mathbf{y}_j\}_{j=1}^{N_u}$.

The core of our method is to find appropriate mappings between semantics defined by a single labeled pair $\mathcal{D}_l$ and StyleGAN's latent space defined by an unlabeled dataset $\mathcal{D}_u$. Our StyleGAN encoder learns the mappings to extract latent codes from semantic scribbles. The overall training procedure (Figure 2 and Subsection 3.2) involves iterating three steps; (i) first, we randomly sample images from a StyleGAN generator pre-trained with $\mathcal{D}_u$, (ii) then perform pseudo labeling with $\mathcal{D}_l$ for the sampled images, and (iii) update the StyleGAN encoder parameters. In pseudo labeling (Subsection 3.1), we first extract feature vectors representing each semantic class and then find matchings with StyleGAN's feature maps. Such matchings enable *pseudo labeling*, i.e., to obtain pseudo semantic scribbles from random noise in StyleGAN's latent space, which are then used to train an encoder to extract latent codes for controlling the pre-trained StyleGAN generator.

For inference, we extract latent codes from test inputs using the StyleGAN encoder and generate images by feeding the latent codes to the StyleGAN generator. As the StyleGAN encoder, we adopt the pSp encoder [26]. The inference process is the same as that of pSp; from semantic scribbles, the encoder generates latent codes that are then fed to the fixed StyleGAN generator to control the spatial layout. We can optionally change or fix latent codes that control the local details of the output images. In addition, we can refine the generated results via optimization-based GAN inversion in a post-processing stage (Subsection 3.3). Note that naïvely training the pSp encoder with a single ground-truth pair cannot generalize the mapping function between various semantic scribbles and latent codes (see the results in Section 4). Hereafter, we explain the pseudo labeling process, the training procedure with the pseudo semantic scribbles, and the post-processing stage.

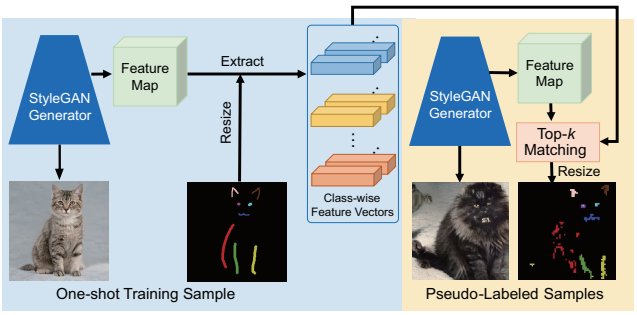

Fig. 3: Pipeline of pseudo labeling. Left: We extract class-wise feature vectors for all labeled pixels. Right: For each feature vector, we take top-k correspondences and assign its class label to corresponding pixels whose similarities are above threshold t.

3.1 Pseudo labeling

Figure 3 illustrates the pseudo labeling process for semantic scribbles. As explained at the beginning of Section 3, each semantic scribble has a unique class label, and unoccupied pixels have an "unknown" label. To obtain pseudo semantic scribbles, we first extract *class-wise feature vectors* corresponding to class labels except for the "unknown" label. We then find matchings between the feature vectors and pixels in images newly sampled from StyleGAN.

Specifically, we extract StyleGAN's feature maps $\mathbf{F} \in \mathbb{R}^{Z\times W'\times H'}$ (where Z, W', and H' are the number of channels, width, and height) corresponding to the semantic scribbles $\mathbf{x}$. There are two ways to prepare the feature maps $\mathbf{F}$. If a pair of semantic scribbles $\mathbf{x}$ and GT RGB image $\mathbf{y}$ is available in $\mathcal{D}_l$, we first invert $\mathbf{y}$ into the StyleGAN's latent space via optimization-based GAN inversion using the L2 and LPIPS [42] losses between $\mathbf{y}$ and the StyleGAN output. We then extract the feature map via forward propagation. Alternatively, we feed one noise vector to the pre-trained StyleGAN generator, extract the feature map and synthesized image, and manually annotate the synthesized image to create semantic scribbles. In all of our results, the feature map $\mathbf{F}$ is at a resolution of 64×64 and extracted from the layer closest to the output layer of the StyleGAN generator. We set the resolution to 64×64 by reference to Collins et al. [9]; they used 32×32 feature maps to get "most semantic" clusters, but we want to obtain modestly high-resolution pseudo semantic scribbles to train the encoder.

After obtaining StyleGAN's feature map $\mathbf{F}$, we extract class-wise feature vectors $\{\mathbf{f}_j^c\}_{j=1}^{N_c}$ for semantic scribbles $\mathbf{x}' \in \{0,1\}^{C\times W'\times H'}$ (where $\mathbf{x}'$ is a downsampled version of $\mathbf{x}$ at the same spatial resolution as $\mathbf{F}$,

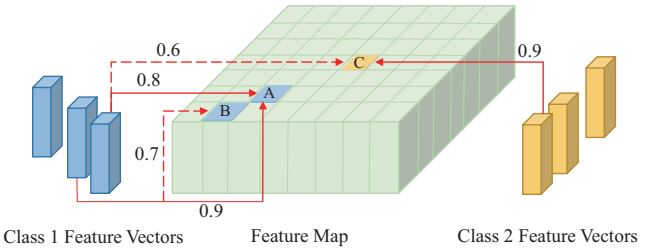

Fig. 4: Matching between class-wise feature vectors and a feature map. The solid and dashed arrows are first and second matchings, respectively, and numbers near arrows show cosine similarity. Nearest-neighbor matching may result in fewer samples in pseudo labeling than in genuine ones due to many-to-one mapping like cell A. K-nearest-neighbor matching can avoid this problem by increasing matchings with cells like B. In addition, if feature vectors with different classes match same cell like C, we assign class label with highest similarity (i.e., class 2 in this case).

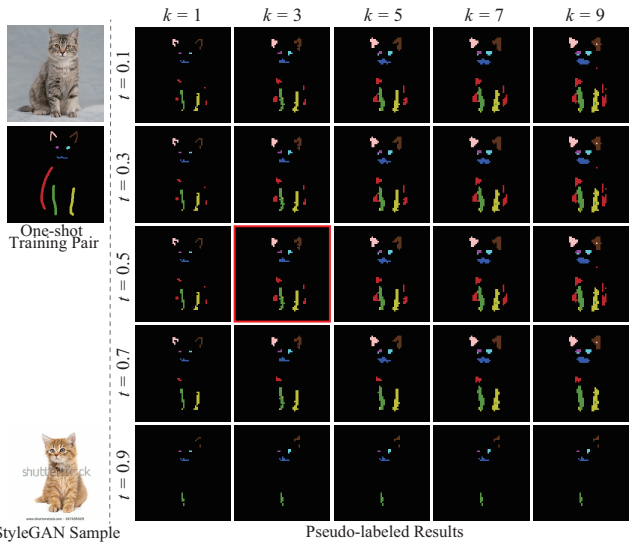

Fig. 5: Our pseudo-labeled results with different parameters. We set $k = 3$ and $t = 0.5$ (highlighted in red) for all results.

and N_c is the number of pixels annotated with class c in $\mathbf{x}'$). Meanwhile, we randomly sample images from the pre-trained StyleGAN model and extract their feature maps from the hidden layers. Next, we take the correspondence between each class-wise feature vector $\mathbf{f}_j^c$ and the pixels in these feature maps. For pseudo semantic scribbles, we want to retain spatial sparsity so that the pseudo semantic scribbles resemble genuine ones as much as possible. However, many annotated pixels in $\mathbf{x}'$ might match an identical pixel-wise vector in the feature maps (i.e., many-to-one mapping), which results in fewer samples in pseudo semantic scribbles (see Figure 4). Therefore, we calculate the top-k (i.e., k-nearest-neighbor) matching instead of one-nearest-neighbor to increase matchings. In the case of many-to-one mappings from different classes, we assign the class label of an annotation that has the largest cosine similarity. To avoid outliers, we discard the matchings if their cosine similarities are lower than a threshold t and assign the "unknown" label. Figure 5 shows examples of pseudo semantic scribbles with different parameters. We set $k = 3$ and $t = 0.5$ for all results in this paper.

3.2 Training procedure

Figure 2 and Algorithm 1 summarize the learning process of the StyleGAN encoder. After preparing a pre-trained StyleGAN and class-wise feature vectors as mentioned in Subsection 3.1, we iteratively train the StyleGAN encoder. Specifically, we sample random noise $\mathbf{z}$ from a normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$ and feed it to the StyleGAN's mapping network M to obtain latent

Algorithm 1 One-shot learning of StyleGAN encoder

Input: A labeled set $\mathcal{D}_l$ and unlabeled set $\mathcal{D}_u$
  Train StyleGAN using $\mathcal{D}_u$
  Compute class-wise feature vectors using $\mathcal{D}_l$
  for each training iteration **do**
    Sample latent codes according to $\mathcal{N}(\mathbf{0}, \mathbf{I})$
    Feed the latent codes to the generator
    Perform pseudo labeling using class-wise feature vectors
    Feed the pseudo semantic scribbles to the encoder
    Compute the loss $\mathcal{L}$ as in Eq. (1)
    Compute the gradient and optimize the encoder
  end for

codes $\mathbf{w}$. Next, we feed the latent codes to the pre-trained StyleGAN generator to synthesize images while extracting the intermediate layer's feature maps. Using these feature maps and the class-wise feature vectors, we create pseudo semantic scribbles, which are then fed to the encoder to extract latent codes $\{\hat{\mathbf{w}}_i\}_{i=1}^L$ (L is the number of StyleGAN layers to input latent codes). In the backward pass, we optimize the encoder using the following loss function:

$$\mathcal{L} = \mathbb{E}_{\mathbf{w} \sim M(\mathbf{z})} \Sigma_{i=1}^{L} \|\hat{\mathbf{w}}_i - \mathbf{w}\|_2^2. \quad (1)$$

This loss function indicates that our training is quite simple because backpropagation does not go through the pre-trained StyleGAN generator and does not require hyperparameter tuning.

3.3 Post-processing via optimization

In general, encoder-based GAN inversion is fast but not so accurate in reconstructing images [38] (see Fig-

Fig. 6: Post-processed results (a) without and (b) with latent code initialization via our encoder.

Fig. 7: Comparison with different post-processing methods. We first obtain the result (b) from the input image (a) via our encoder-based approach. (c) and (d) show the post-processed results of (b) using Equations (2) and (4).

ure 7(a)(b)). Therefore, in this section, we propose an optimization-based post-processing to further refine our encoder-based outputs. Inspired by the knowledge that StyleGAN features form semantic clusters using spherical k-means clustering [9], we formulate an optimization procedure such that feature vectors in the same class will become similar in the cosine distance. Specifically, we aim to find a latent code $\mathbf{w}^*$ in the $\mathcal{W}$ latent space[1], following the objective function of k-means clustering:

$$\mathbf{w}^* = \text{argmin}_{\hat{\mathbf{w}}} \mathcal{D}(f(\hat{\mathbf{w}}), \mathbf{C}\mathbf{X'}_{test}), \tag{2}$$

where the function f outputs a flattened StyleGAN feature map (i.e., $Z \times W'H'$ matrix) from the latent code $\hat{\mathbf{w}}$, $\mathbf{C} \in \mathbb{R}^{Z \times C}$ is a matrix consisting of C cluster centers in Z dimensions, $\mathbf{X'}_{test} \in \mathbb{R}^{C \times W'H'}$ is a matrix obtained by flattening the resized test semantic scribbles $\mathbf{x}'_{test} \in \{0,1\}^{C \times W' \times H'}$, and $\mathcal{D}$ is the mean of negative cosine similarities between corresponding columns of two input matrices. In the function f, we feed the latent code $\hat{\mathbf{w}}$ to be optimized to the first eight layers in StyleGAN because latent codes fed to the latter layers affect fine-scale attributes rather than layout. Meanwhile, we feed fixed latent codes (e.g., encoder outputs) to the latter layers and exclude them from gradient computation. For the cluster centers $\mathbf{C}$, we can define $\mathbf{C} = (\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_C)$, where $\mathbf{v}_c$ for class c is computed with the feature map $\mathbf{F}$ and the resized training semantic scribbles $\mathbf{x}'$ as follows:

$$\mathbf{v}_c = \frac{\sum_{x,y} \mathbf{F}^{(x,y)} \mathbb{1}\left[\mathbf{x}'^{(c,x,y)} = 1\right]}{\sum_{x,y} \mathbb{1}\left[\mathbf{x}'^{(c,x,y)} = 1\right]}, \tag{3}$$

where (x, y) denote pixel positions, and $\mathbb{1}[\cdot]$ is an indicator function that returns 1 if the argument is true and 0 otherwise. To avoid falling into undesirable local minima in Equation (2), we initialize $\hat{\mathbf{w}}$ with the encoder output $\hat{\mathbf{w}}_0$ before optimization.

However, if we naïvely compute Equation (2), the pixels in each cluster will have the same feature vectors, converging to an almost flat-color image, as shown in Figure 7(c). Inspired by GANSpace [12], we solve this problem by restricting the latent code exploration to certain principal directions. This restriction prevents the latent codes from deviating significantly from the actual data distribution. Specifically, we aim to find a parameter vector $\mathbf{g}^* \in \mathbb{R}^Z$ to obtain the latent code $\mathbf{w}^* = \mathbf{W}\mathbf{g}^*$ by using principal components in the latent space. Here, $\mathbf{W}$ is a $Z \times Z$ matrix consisting of basis vectors, sorted in the ascending order of their corresponding eigenvalues. We first initialize $\mathbf{g}^*$ with $\mathbf{W}^{-1}\hat{\mathbf{w}}_0$ ($\hat{\mathbf{w}}_0$ is the encoder output) and then optimize a part of $\mathbf{g}^*$ as follows:

$$\mathbf{g}^*_{s:t} = \text{argmin}_{\hat{\mathbf{g}}_{s:t}} \mathcal{D}(f(\mathbf{W}\hat{\mathbf{g}}), \mathbf{C}\mathbf{X'}_{test}), \tag{4}$$

where $\hat{\mathbf{g}}$ is a parameter vector initialized similar to $\mathbf{g}^*$. The subscript $s:t$ denotes a vector's elements from s-th to t-th, and we set $s = Z - 8$ and $t = Z$. Namely, we optimize the latent code only in the principle directions corresponding to the largest eight eigenvalues. Note that the "unknown" region is excluded from the

[1] We can also optimize latent codes in the $\mathcal{W}+$ latent space [1], but we chose the $\mathcal{W}$ latent space because it can obtain stable results.

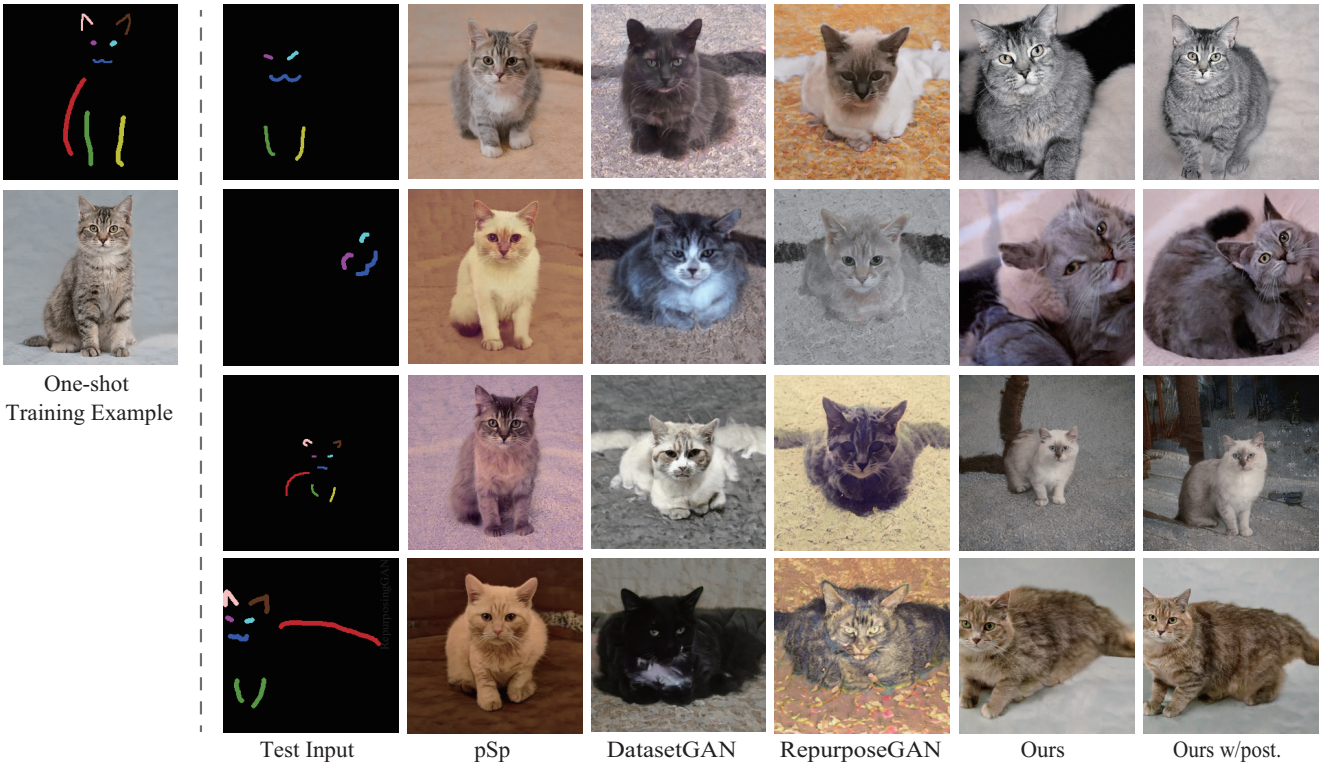

Fig. 8: Comparison of the cat images generated using the semantic scribbles in one-shot setting.

loss calculation. Figure 7(d) shows the result obtained by decoding the latent code $\mathbf{W}\mathbf{g}^*$.

Note that, because Equation (4) has local optimal solutions, the result depends on the initial value. Depending on the initial value of $\hat{\mathbf{g}}$, converged results may deviate from the layout of the input mask, as shown in Figure 6(a). Therefore, we can see that optimizing Equation (4) without appropriate initialization is insufficient and the latent code initialization by our encoder is essential, as shown in Figure 6(b).

4 Experiments

We conducted experiments to validate the effectiveness of our method. We first explain implementation details of our method and then show results and discuss them.

4.1 Implementation details

We implemented our method with PyTorch and ran our code on PCs equipped with GeForce GTX 1080 Ti. We used public StyleGAN2 models [25,19] pre-trained with unlabeled images. We trained the encoder using the Ranger optimizer [26] with a learning rate of 0.0001. The batch size (i.e., the number of pseudo-labeled images per iteration) was set to 2. We performed 100,000 iterations, which took a day at most. For post-processing, we applied PCA to 1,024 randomly-sampled latent codes to obtain bases $\mathbf{W}$. We used the Adam optimizer with a learning rate of 0.05 and performed 200 iterations for the cat images and 50 iterations for the anime portrait images. Testing one image took about 0.2 seconds for the encoder-based approach and around 10 seconds for post-processing. To increase interactivity, the users first edited images via only the encoder-based approach and then used post-processing as necessary.

4.2 Results

Qualitative results. Figures 8 and 9 show cat and anime portrait images generated from semantic scribbles. In these comparisons, pSp [26], which was trained only with a one-shot example without our pseudo semantic scribbles and randomly-sampled images, ignored the input layouts due to over-fitting to the single training examples and did not reflect the input layouts in the results. We obtained the other results via our framework with different pseudo-labeling approaches. The compared methods are DatasetGAN [43] and RepurposeGAN [35], which use pre-trained StyleGAN mod-

Fig. 9: Comparison of the anime portrait images generated using the cross lines in one-shot setting.

Table 1: Layout faithfulness scores in user study. Best score and second best score in each row are marked in red and blue. OAvg. means overall average for all cases.

Cat				
ID	DatasetGAN	RepurposeGAN	Ours	Ours w/ post.
1	3.27	1.67	**3.33**	**4.13**
2	1.60	1.33	**3.87**	**4.73**
3	**2.93**	1.87	**3.93**	**3.93**
4	2.07	1.20	**4.53**	**4.60**
Avg.	2.47	1.52	**3.92**	**4.35**
Anime portrait				
ID	DatasetGAN	RepurposeGAN	Ours	Ours w/ post.
5	3.47	**4.13**	3.60	**3.73**
6	**4.27**	3.87	4.00	**4.33**
7	1.93	1.80	**2.93**	**4.53**
8	2.27	3.13	**4.40**	**4.47**
Avg.	2.98	3.23	**3.73**	**4.27**
OAvg.	2.73	2.38	**3.83**	**4.31**

els similarly to ours[2]. For the cat results, the compared methods did not sufficiently consider the given layouts. In contrast, our results overall reflected the given layouts. For the anime portrait results, the compared methods worked relatively well. Still, some results did not reflect the given layouts (e.g., face orientation in the third row in Figure 9). Meanwhile, our method worked well as seen in the overall results. Furthermore, post-processing (Ours w/post.) improved their layouts, especially in the case of the cat results. We also demonstrate our interactive demo in the supplemental video. Additionally, Figure 14 shows results with LSUN car and ukiyo-e datasets (see Appendix A for details).

User study. We conducted a user study for quantitative evaluation to compare our method with the previous work regarding the layout faithfulness of generated images. We asked 15 participants to score the eight sets of results in Figures 8 and 9 in a range of 1 to 5, where 1 means that the generated images did not match the input layouts at all, and 5 means that the generated images completely matched input layouts. We compared the results of four methods (i.e., DatasetGAN, RepurposeGAN, Ours, and Ours w/post), and thus the number of collected evaluations is 256.

Table 1 shows the average scores for each case, the average scores for each category, and the overall average scores for all cases. We assigned an index to each case in the order from top to bottom rows in Figures 8 and 9. For the cat results, our method obtained the best score, and we can also confirm the effectiveness of the post-processing stage. For the anime portrait re-

[2] We used the public codes with the default settings that can be downloaded from the DatasetGAN and RepurposeGAN project pages.

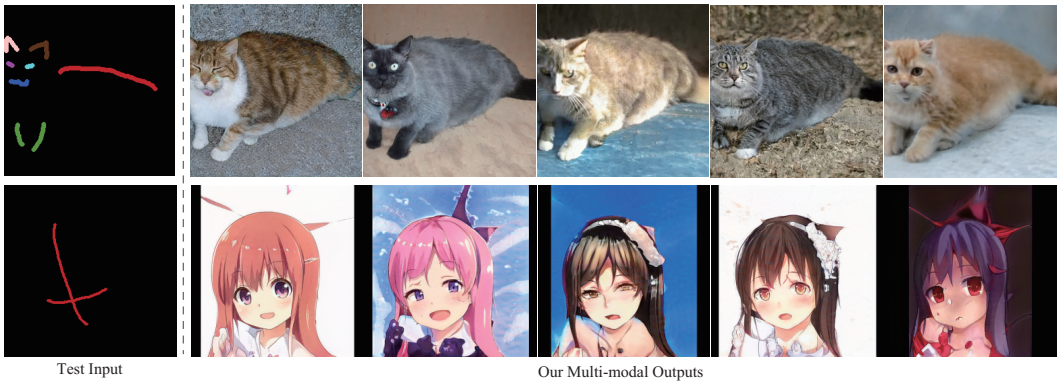

Fig. 10: Multi-modal results of our method.

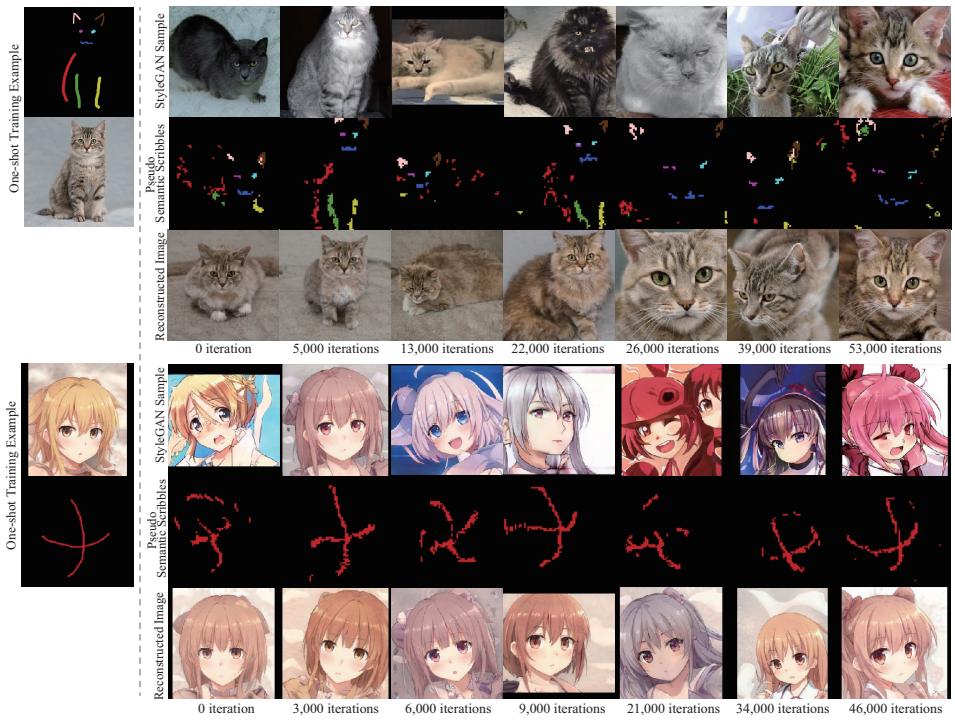

Fig. 11: Intermediate training outputs.

sults, while the compared methods sometimes obtained the best score, our method obtained the best or second best scores for all cases and obtained the best score on average. Overall, Ours and Ours w/ post. outperformed the compared methods on average in the user study.

Multi-modal generation. Figure 10 demonstrates that our method can generate multi-modal results. To obtain multi-modal outputs in test time, we followed the same approach as pSp [26]; we fed latent codes encoded from an input layout to the first l layers of the generator and random latent codes to the other layers. We set $l = 5$ for the results in Figure 10, and $l = 8$ for the other results.

Analysing training procedure. Figure 11 shows the intermediate outputs during training iterations. For each set of results, we fed random latent codes to the pre-trained StyleGAN generator to obtain synthetic images (top row) and feature vectors, from which we calculated pseudo semantic scribbles (middle row). We then used the pseudo semantic scribbles to train the encoder to generate latent codes for reconstructing images (bottom row). It can be seen that the layouts of the bottom-row images reconstructed from the middle-row pseudo semantic scribbles gradually become close to those of the top-row StyleGAN samples as the training iterations increase.

Fig. 12: Comparison of pseudo labeling. Given StyleGAN2's outputs, we obtained pseudo-labeled results using our method, RepurposeGAN [35], and SemanticGAN [20]. Given StyleGAN's images, we also obtained pseudo-labeled results using DatasetGAN [43].

Analyzing pseudo labeling. We analyzed the fidelity[3] of pseudo labeling. Figure 12 shows pseudo-labeled images generated by the compared methods using one-shot pairs in Figures 8 and 9. Using our method and RepurposeGAN [35], we assigned class labels to StyleGAN2's outputs generated from the same latent codes. Meanwhile, we could not evaluate DatasetGAN [43] with the same inputs because the official code uses not StyleGAN2 but StyleGAN. We, therefore, analyzed the labeling performance for other inputs qualitatively. As can be seen in the results of RepurposeGAN and DatasetGAN, some noise appeared, and labels were missing. These existing approaches, which train segmentation models, are suitable for "few-shot" and "dense" semantic segmentation, as their papers demonstrated. However, their performance may decrease due to overfitting for "one-shot" and "sparse" semantic scribbles consisting of imbalanced class labels. In addition, we compared our method with SemanticGAN [20]. This approach does not use pre-trained StyleGAN models and trains GANs from scratch. Therefore, we trained SemanticGANs using the official code with the LSUN cat [40] and Danbooru2019 portrait [4] datasets, and our one-shot annotations. However, the labeling results looked not like sparse scribbles but rather like inaccurate dense masks. In addition, the generated masks looked similar even if the input images were different, probably because mode collapse occurred due to the lack of annotation data. In contrast to these existing methods, our pseudo labeling is simple yet effective for sparse semantic scribbles.

[3] Note that achieving perfect pixel alignment between pseudo semantic scribbles and output images is out of our scope and might be too difficult in our one-shot scenario.

5 Conclusion and Future Work

In this paper, we proposed a simple yet effective method for controlling StyleGAN's outputs using se-

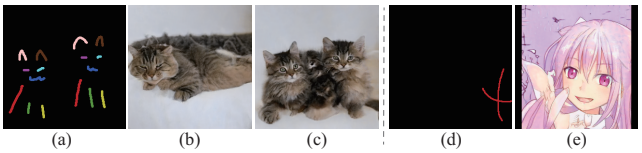

Fig. 13: Limitations of our method. (a)(b) Our encoder-based method cannot consider complex layouts. (c) Post-processing can improve results somewhat. In addition, (d)(e) we cannot handle layouts that pre-trained StyleGAN models cannot reproduce.

mantic scribbles in a one-shot scenario for the first time. To compensate for the lack of pixel-wise annotation data, we generate pseudo semantic scribbles via k-nearest-neighbor mapping between the feature vectors of a pre-trained StyleGAN generator and each semantic class in one-shot labeled pairs. In each training iteration, we can generate a pseudo label from random noise to train an encoder [26] for controlling the pre-trained StyleGAN generator using a simple L2 loss. In addition, we proposed a post-processing method that optimizes latent codes and refines generated images according to principal directions in the latent space. Experiments demonstrated that our method can synthesize high-quality images with more accurate spatial control than competitive methods.

Limitations. Because our pseudo labeling is not completely accurate, its performance may also affect the results of the encoder-based approach. As shown in Figure 13(a)(b), the input layout contains two cats, but the result contains only one cat. Post-processing can alleviate this problem somewhat, as shown in (c). In addition, our method, which depends on pre-trained StyleGAN models, cannot handle layouts that the StyleGAN models cannot reproduce. For example, as shown in Figure 13(d)(e), even if we draw the cross lines on the right side of the image, it is difficult to generate images that reflect those positions because the StyleGAN model is trained with aligned faces. The GAN inversion method [16] and StyleGAN3 [17], which can handle geometric translations, might solve this problem in the future. Finally, although the scope of this paper is to control StyleGAN using sparse semantic scribbles, we would also like to extend our method to handle dense inputs, such as semantic segmentation masks.

References

1. Abdal, R., et al.: Image2StyleGAN: How to embed images into the StyleGAN latent space? In: ICCV 2019, pp. 4431–4440 (2019)
2. Abdal, R., et al.: Image2StyleGAN++: How to edit the embedded images? In: CVPR 2020, pp. 8293–8302 (2020)
3. Alaluf, Y., Patashnik, O., Cohen-Or, D.: ReStyle: A residual-based StyleGAN encoder via iterative refinement. In: ICCV 2021, pp. 6691–6700 (2021)
4. Branwen, G., Anonymous, Community, D.: Danbooru2019 portraits: A large-scale anime head illustration dataset (2019). URL https://www.gwern.net/Crops#danbooru2019-portraits
5. Brock, A., Donahue, J., Simonyan, K.: Large scale GAN training for high fidelity natural image synthesis. In: ICLR 2019 (2019)
6. Chen, Q., Koltun, V.: Photographic image synthesis with cascaded refinement networks. In: ICCV 2017, pp. 1520–1529 (2017)
7. Chen, W., Hays, J.: SketchyGAN: Towards diverse and realistic sketch to image synthesis. In: CVPR 2018, pp. 9416–9425 (2018)
8. Chiu, C., Koyama, Y., Lai, Y., Igarashi, T., Yue, Y.: Human-in-the-loop differential subspace search in high-dimensional latent space. ACM Trans. Graph. **39**(4), 85 (2020)
9. Collins, E., et al.: Editing in style: Uncovering the local semantics of gans. In: CVPR 2020, pp. 5770–5779 (2020)
10. Endo, Y., Kanamori, Y.: Diversifying semantic image synthesis and editing via class- and layer-wise vaes. Comput. Graph. Forum **39**(7), 519–530 (2020)
11. Gao, C., Liu, Q., Xu, Q., Wang, L., Liu, J., Zou, C.: SketchyCOCO: Image generation from freehand scene sketches. In: CVPR 2020, pp. 5173–5182 (2020)
12. Härkönen, E., et al.: Ganspace: Discovering interpretable GAN controls. In: NeurIPS 2019 (2020)
13. Isola, P., et al.: Image-to-image translation with conditional adversarial networks. In: CVPR 2017, pp. 5967–5976 (2017)
14. Jahanian, A., Chai, L., Isola, P.: On the "steerability" of generative adversarial networks. In: ICLR 2020 (2020)
15. Jo, Y., Park, J.: SC-FEGAN: face editing generative adversarial network with user's sketch and color. In: ICCV 2019, pp. 1745–1753
16. Kang, K., Kim, S., Cho, S.: GAN inversion for out-of-range images with geometric transformations. In: ICCV 2021, pp. 13,941–13,949 (2021)
17. Karras, T., Aittala, M., Laine, S., Härkönen, E., Hellsten, J., Lehtinen, J., Aila, T.: Alias-free generative adversarial networks. In: NeurIPS 2021 (2021)
18. Karras, T., et al.: A style-based generator architecture for generative adversarial networks. In: CVPR 2019, pp. 4401–4410 (2019)
19. Karras, T., et al.: Analyzing and improving the image quality of stylegan. In: CVPR 2020, pp. 8107–8116 (2020)
20. Li, D., Yang, J., Kreis, K., Torralba, A., Fidler, S.: Semantic segmentation with generative models: Semi-supervised learning and strong out-of-domain generalization. In: CVPR 2021, pp. 8300–8311 (2021)
21. Li, K., et al.: Diverse image synthesis from semantic layouts via conditional IMLE. In: CVPR 2019, pp. 4219–4228 (2019)
22. Li, Y., Cheng, Y., Gan, Z., Yu, L., Wang, L., Liu, J.: Bachgan: High-resolution image synthesis from salient object layout. In: CVPR 2020, pp. 8362–8371 (2020)
23. Liu, X., et al.: Learning to predict layout-to-image conditional convolutions for semantic image synthesis. In: NeurIPS 2019, pp. 568–578 (2019)
24. Park, T., et al.: Semantic image synthesis with spatially-adaptive normalization. In: CVPR 2019, pp. 2337–2346 (2019)

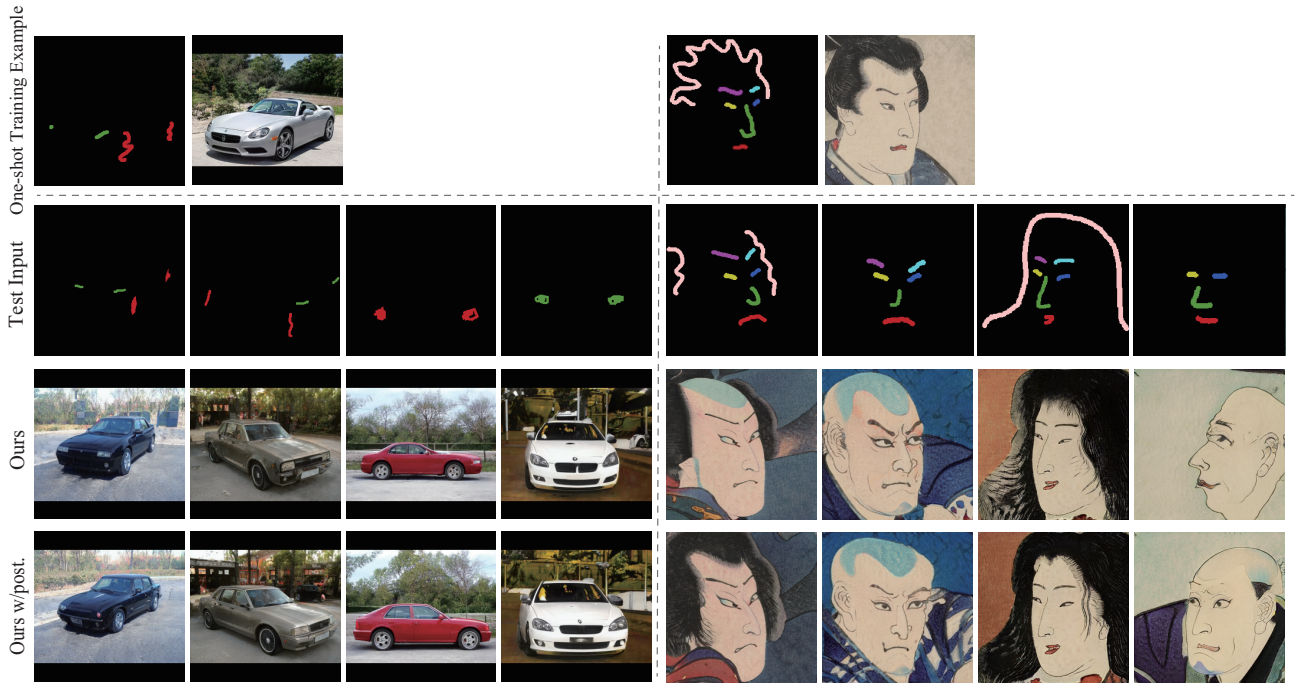

Fig. 14: Additional qualitative results for other pre-trained StyleGAN models.

25. Pinkney, J.: Awesome Pretrained StyleGAN2 (2020). https://github.com/justinpinkney/awesome-pretrained-stylegan2
26. Richardson, E., Alaluf, Y., Patashnik, O., Nitzan, Y., Azar, Y., Shapiro, S., Cohen-Or, D.: Encoding in style: A StyleGAN encoder for image-to-image translation. In: CVPR 2021, pp. 2287–2296 (2021)
27. Roich, D., Mokady, R., Bermano, A.H., Cohen-Or, D.: Pivotal tuning for latent-based editing of real images. CoRR **abs/2106.05744** (2021)
28. Sangkloy, P., Lu, J., Fang, C., Yu, F., Hays, J.: Scribbler: Controlling deep image synthesis with sketch and color. In: CVPR 2017, pp. 6836–6845 (2017)
29. Shen, Y., Gu, J., Tang, X., Zhou, B.: Interpreting the latent space of gans for semantic face editing. In: CVPR 2020, pp. 9240–9249 (2020)
30. Shen, Y., Zhou, B.: Closed-form factorization of latent semantics in gans. In: CVPR 2021, pp. 1532–1540 (2021)
31. Sun, W., Wu, T.: Image synthesis from reconfigurable layout and style. In: ICCV 2019, pp. 10,530–10,539 (2019)
32. Tang, H., et al.: Dual attention gans for semantic image synthesis. In: ACM Multimedia 2020, pp. 1994–2002 (2020)
33. Tang, H., et al.: Local class-specific and global image-level generative adversarial networks for semantic-guided scene generation. In: CVPR 2020, pp. 7867–7876 (2020)
34. Tov, O., Alaluf, Y., Nitzan, Y., Patashnik, O., Cohen-Or, D.: Designing an encoder for StyleGAN image manipulation. ACM Trans. Graph. **40**(4), 133:1–133:14 (2021)
35. Tritrong, N., Rewatbowornwong, P., Suwajanakorn, S.: Repurposing GANs for one-shot semantic part segmentation. In: CVPR 2021, pp. 4475–4485 (2021)
36. Wang, S.Y., Bau, D., Zhu, J.Y.: Sketch your own GAN. In: ICCV 2021, pp. 14,050–14,060 (2021)
37. Wu, Z., Lischinski, D., Shechtman, E.: StyleSpace analysis: Disentangled controls for StyleGAN image generation. In: CVPR 2021, pp. 12,863–12,872 (2021)
38. Xia, W., et al.: GAN Inversion: A Survey. CoRR **abs/2101.05278** (2021)
39. Xue, Y., Guo, Y., Zhang, H., Xu, T., Zhang, S., Huang, X.: Deep image synthesis from intuitive user input: A review and perspectives. Comput. Vis. Media **8**(1), 3–31 (2022)
40. Yu, F., Zhang, Y., Song, S., Seff, A., Xiao, J.: LSUN: construction of a large-scale image dataset using deep learning with humans in the loop. CoRR **abs/1506.03365** (2015)
41. Zhang, P., Zhang, B., Chen, D., Yuan, L., Wen, F.: Cross-domain correspondence learning for exemplar-based image translation. In: CVPR 2020, pp. 5142–5152 (2020)
42. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: CVPR 2018, pp. 586–595 (2018)
43. Zhang, Y., Ling, H., Gao, J., Yin, K., Lafleche, J., Barriuso, A., Torralba, A., Fidler, S.: DatasetGAN: Efficient labeled data factory with minimal human effort. In: CVPR 2021, pp. 10,145–10,155 (2021)
44. Zhao, B., Meng, L., Yin, W., Sigal, L.: Image generation from layout. In: CVPR 2019, pp. 8584–8593 (2019)
45. Zhu, Z., et al.: Semantically multi-modal image synthesis. In: CVPR 2020, pp. 5466–5475 (2020)

A Additional Qualitative Results

As shown in Figure 14, we also tried applying our method to other StyleGAN models pre-trained with LSUN car or ukiyo-e datasets. We can see that our method can generate photorealistic images according to the given layouts even if the one-shot training examples and test inputs were specified roughly and sparsely. We can also confirm that our post-processing made the generated images more faithful to the given layouts.