Where is My Spot? Few-shot Image Generation via Latent Subspace Optimization – Supplementary Materials –

Chenxi Zheng [1†] Bangzhen Liu [1†] Huaidong Zhang [1‡] Xuemiao Xu[1,2,3,4‡] Shengfeng He [5]
[1]South China University of Technology [2]State Key Laboratory of Subtropical Building Science
[3]Ministry of Education Key Laboratory of Big Data and Intelligent Robot
[4]Guangdong Provincial Key Lab of Computational Intelligence and Cyberspace Information
[5]Singapore Management University
{cszcx, cs_liubz}@mail.scut.edu.cn, {huaidongz, xuemx}@scut.edu.cn, shengfenghe@smu.edu.sg

In the supplement, we provide additional details and results to support our main contributions.

1. Visualization of the Latent Space

The t-SNE [8] visualizations of latent space are shown in Fig. 1. Figure 1a illustrates the latent subspace with randomly initialized category centroids. With a randomly initialized unseen centroid c^u, the sampled unseen latent codes $w_i = G_{map}(z, c^u)$ can not be localized correctly around the target anchors $\varTheta$. Additionally, the randomly initialized trainable anchors $\varPhi_{opt}$ are unable to approximate the target anchors. Fig. 1b depicts the latent space after latent anchor localization. The trainable anchors successfully approximate the targets and correctly pull the entire subspace towards the neighborhood of target anchors, roughly estimating the distribution of unseen latent codes. For latent subspace refinement, as shown in Fig. 1c, the target anchors are disregarded because we use the perceptual loss instead of the approximation loss. Thus, the trainable anchors break away from the target anchors.

2. Implementation Details

In this section, we complement the implementation details, including inversion, conditional discriminator, and other training details.

StyleGAN Inversion. For StyleGAN inversion, we use the optimization loss proposed in II2S [10] together with a similarity loss [2] to optimize the latent codes in our implementation.

Similarity Loss in Eq. 7[*]. Following [2], we define the similarity loss by:

$$\mathcal{L}_{sim}(x, \hat{x}) = 1 - \langle C(x), C(\hat{x}) \rangle, \tag{1}$$

where C is a ResNet50 [3] trained with MOCOv2 [1].

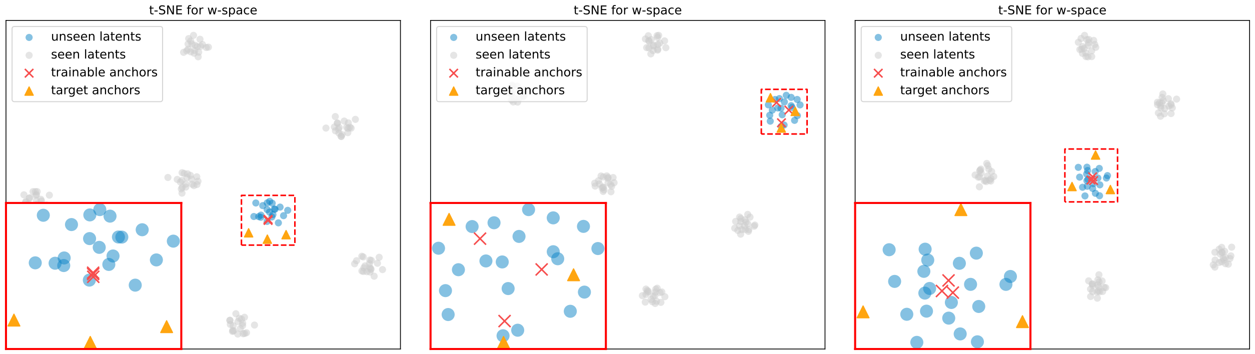
Conditional Discriminator. The conditional discriminator uses a projection head [6]. The features d of an image x are extracted by the convolution layers of the discriminator D_{conv}, i.e., $d = D_{conv}(x)$. The inner product of d and class embedded vector is calculated to produce a conditional real/fake estimation. To achieve the supervision for the extra unseen category, we extend the embedding parameter p from $\mathbb{R}^{n_s \times d_{hid}}$ to $\mathbb{R}^{(n_s+1) \times d_{hid}}$, where n_s and d_{hid} represents the number of seen categories and dimension of the hidden layer. For the regularization of the discriminator during latent subspace refinement, the output features of convolution layers are normalized before calculating the L_2 norm considering the magnitude of the loss.

Other training details. We use the term $kimgs$ to denote the number of training steps as [4]. $kimgs = n$ means the network is optimized with total times of $n * 1000$ using unseen images only. For latent subspace localization, we optimize for $1kimgs$ times. For latent subspace refinement, we optimize for $5kimgs$ time. The experiments are done on an NVIDIA GeForce RTX 3090.

3. Analysis of Noise Intensity Factor

We introduce a noise intensity factor t as an extra hyperparameter to control image generation. The intensity factor controls the magnitude of the noise, which is crucial to reveal the overall generation capability of the optimized generator. We further analyze the influence of the number

[*]Equation from the main body.
[†]Equal Contributions.
[‡]Corresponding authors.



(a) randomly initialized (b) anchor-localized (c) subspace-optimized

Figure 1. Visualization of the latent space of randomly initialized, anchor localized, and subspace optimized. The dashed area is enlarged to the lower left corner. The unseen and seen latent codes are sampled with a Gaussian noise z and the corresponding category centroid c. Trainable anchors Φ_{opt} are parameters that are optimized to approximate the inverted target anchors Θ during latent subspace optimization.

of optimization steps (Sec. 3.2) and the number of samples (Sec. 3.3) under different noise intensities.

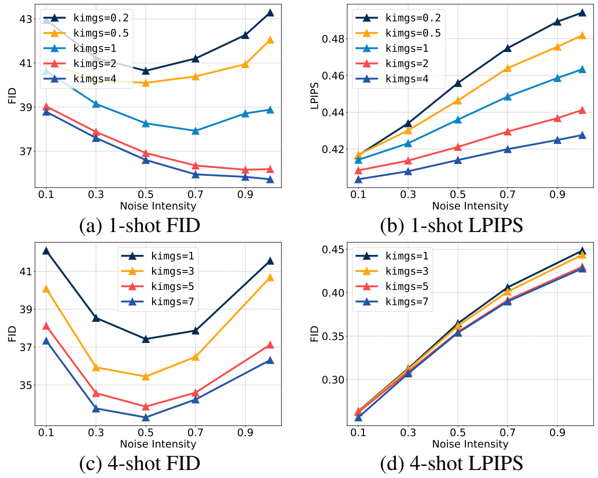
3.1. Noise Intensity

The intensity factor $t \in [0, 1]$ achieves control by $\hat{x} = G^u(z * t, c^u)$, where z is a randomly sampled noise. We carry out experiments across varying levels of noise intensity and show the comprehensive generation capability by presenting FID and LPIPS curves.

As is shown in Fig. 2 and Fig. 3, the majority of FID curves exhibit a hyperbolic shape. This can be explained from the perspective of the latent space. As the intensity factor lies around zero, all the produced images are visually similar, leading to high FID. When the noise intensity goes up and lies in the optimal interval, the sampling area is limited to the high-confidence region of the subspace. This results in the generation of reliable unseen images. Conversely, the high-intensity noise greatly increases the sampling range of the latent space, so the latent codes at the edge of the subspace will produce images with a semantic shift. Consequently, the distance between the generated distribution and the unseen distribution widens.

Note that the FID curve under the 1-shot setting shows a monotonous decreasing tendency when the training step is larger than $2kimgs$. This indicates that even if we set the factor $t = 1$, the marginal area of the subspace can still produce unseen features. However, the generator is essentially overfitting to the unseen categories. Since the subspace trained with the full intensity is based on 100 samples, it is unreasonable to expect that a complete subspace with $t = 1$ can be solely determined through k unseen samples.

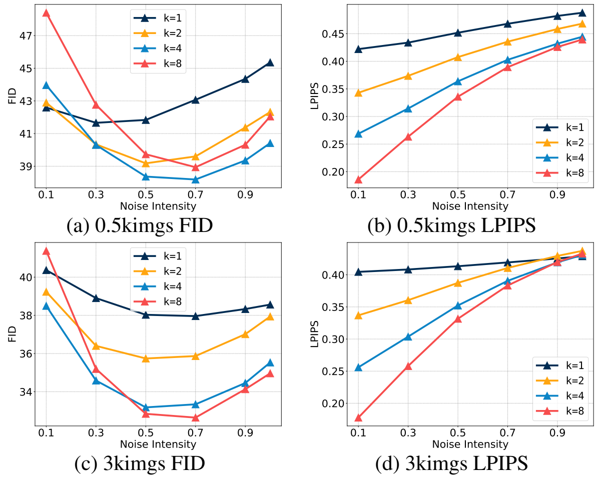
Regarding LPIPS, all LPIPS curves demonstrate a monotonically increasing trend, suggesting a strong correlation between the range of latent code sampling and generation diversity.



(a) 1-shot FID (b) 1-shot LPIPS

(c) 4-shot FID (d) 4-shot LPIPS

Figure 2. Statistical analysis of the optimization steps.



(a) 0.5kimgs FID (b) 0.5kimgs LPIPS

(c) 3kimgs FID (d) 3kimgs LPIPS

Figure 3. Statistical analysis of the number of shots k.

3.2. Number of steps

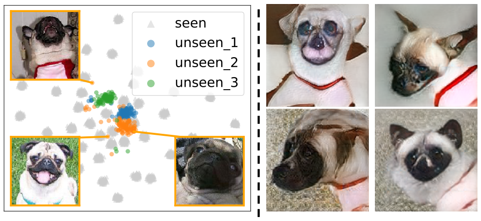
In contrast to the previous emphasis on curve shapes, we will concentrate on the relationship among curves in the following analysis. We visualize the FID and LPIPS curves under different optimization steps in Fig. 2. As the number of training iterations increases, the FID and LPIPS curves for both the 1-shot and 4-shot settings exhibit a similar pattern. It is noticeable that, at the same level of the noise intensity, both FID and LPIPS gradually decrease, implying that the distribution is gradually aligned as the optimization progresses. However, a certain degree of diversity is unavoidably lost in the process.

3.3. Number of shots

Fig. 3 shows the experimental results conducted on the different number of shots. With the increase of k, FID no longer shows a monotonous decreasing trend. For instance, when $k = 8$, FID is only superior to that of $k = 1$ during the initial stages (Fig. 3a) of optimization. However, after sufficient training (Fig. 3c), $k = 8$ exhibits a considerable advantage in the interval $[0.5, 1]$. This indicates that a larger number of inputs requires more optimization to capture unseen features.

In contrast, In contrast, LPIPS exhibits a consistent trend where, under the same noise, larger values of k result in smaller LPIPS scores. However, it is apparent that the disparity between the curves gradually diminishes as the factor t increases. This is because our optimization incorporates a magnitude regularization term (Eq. 3*) that encourages the anchors Φ_{opt} to cluster around the centroid c^u. Consequently, the image generated by a zero noise and a centroid, i.e., $\hat{x}_{cen} = G^u(z * \mathbf{0}, c^u)$, can be viewed as an average estimate of the referenced unseen images. Since a larger k increases the likelihood of image duplication among different few-shot tasks, the averaged images $\hat{x}_{cen}$ result in greater similarity, leading to the decrease of LPIPS.

4. Limitations



(a) Latent space visualization (b) Failure cases

Figure 4. Visualization of unseen classes. The right part shows failure cases produced by some outliers (2/3 of images on the left).

The proposed method has two main limitations: the influence of outliers and the optimization speed.

Outliers can lead to poor content quality of generated images. We visualize three unseen classes, each of them with 100 randomly selected samples in Fig. 4a. We can see most samples of the unseen class form a coarse subspace, demonstrating that our assumption is valid for most samples (also proved in all of our quantitative results). On the other hand, this can also show the failure case if we select two outliers to form a 3-shot input. As shown in Fig. 4b, as outliers may not be properly inverted into the latent space, using them produces images with corrupted semantics (e.g. monkey face, cat head, horse-like neck).

Another limitation is generation speed. Compared with other few-shot generation works, our method has no advantage in time, as we require additional time to adapt to each unseen task. We will delve into ways to expedite the process of adapting to new categories.

5. Comparison with Image Translation

Few-shot image generation and few-shot image-to-image translation [7] are closely related. We compared our approach with COCO-FUNIT [7] in Fig. 5. While COCO-FUNIT can successfully transfer some low-level details of an unseen input image to a target image when given content information, it fails to provide correct category semantics, such as a triangular mouth for the dog. The results reveal the main difference between the two areas that few-shot image translation focuses on transferring the style, while we focus on generating new samples for an unseen category.

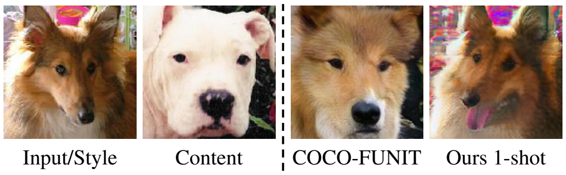


Figure 5. Comparison with COCO-FUNIT.

6. Visualization

The results obtained under 1-shot and 3-shot settings are exhibited in Fig. 7. Results of 1-shot image generations indicate that latent subspace optimization is capable of performing abundant class-irrelevant transformations (e.g., number, shape, orientation) on the single unseen image. For the 3-shot setting, the produced images demonstrate that latent subspace optimization achieves accurate extraction of the category-relevant information of unseen images.

7. Qualitative Ablation

In our ablation study, we present various visualizations of the generated images using different variations of our method. Fig. 6 demonstrates the qualitative results of different components of the four introduced variants of our method. Latent anchor localization benefits the optimizing procedure by reducing the burden of the network to capture general knowledge. Without latent anchor localization, the generator cannot focus on learning the detailed features of the unseen category, but to learn structural information simultaneously during the later refinement. Latent subspace

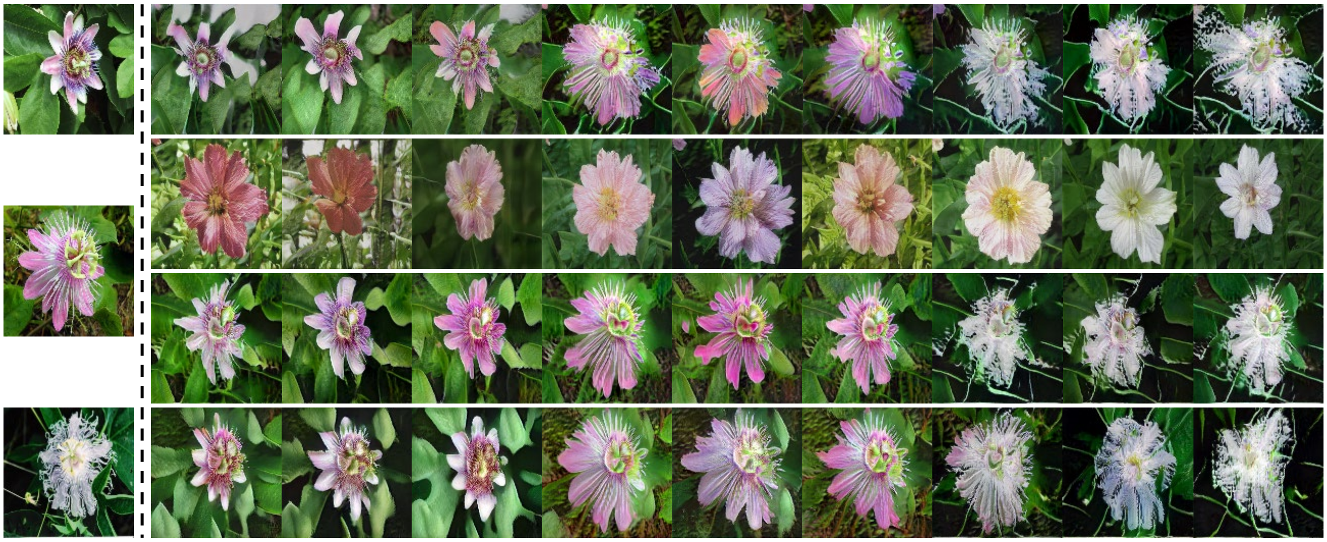


Figure 6. Visualization of ablation experiments. Three unseen images on the left constitute a 3-shot image generation task. Results of w/o localization, w/o refinement, w/o regularization and full are provided from the top to the bottom on the right side.

refinement plays a crucial role in incorporating unseen features. Without refinement, the generator struggles to accurately model the unseen distribution, resulting in distortion. The regularization term is crucial in maintaining diversity. In the absence of the regularization term, the generated flowers share common structural features, such as similar shapes.

8. Image Editing

Image editing is a common downstream application of generative networks. To demonstrate the robustness of our work, we extensively conduct experiments on unseen image editing. We initially apply latent subspace optimization for the unseen image. The unseen image $\hat{x}$ can be reconstructed by $\hat{x} = G^u(a, c^u)$, where G^u, c^u and a represents the refined generator, the unseen centroid and the optimized anchor respectively. Following [9], we manipulate the unseen images with attribute-relevant channels in the style space $\mathcal{S}$. Fig. 8 shows the editing of four attributes from top to bottom. We view the unseen samples as 1-shot tasks and perform latent subspace optimization on each sample independently. Even though the generators are optimized separately, they still share the same style space, enabling the use of the same channels to achieve the same editing effects. The success of editing illustrates that the pretrained semantic-meaningful directions in the latent space are well preserved during latent subspace optimization. As a result, these directions can be applied to unseen image editing.

9. High-quality Image Generation

We present additional results for high-resolution few-shot image generation using the CelebA-HQ [5] portrait dataset. Fig. 9 shows the results of 4-shot image generation, where each task involves generating four 512×512 samples from the same unseen category. Our experimental results indicate that even at high resolutions, the generator does not degrade during the optimization process, but retains the generative ability.

10. Few-shot Incremental Image Generation

The latent subspace optimization framework supports incremental schemes. Qualitative results have been displayed in Fig. 10. In the incremental experiment, 4-shot tasks of different categories are fed to the generator sequentially. The results in Fig. 10 show that the generator can adapt well to novel unseen categories while maintaining its generative capability of the previously trained categories. This success in incremental generation confirms the effectiveness of our method, as it indicates that different unseen categories can be accurately located and refined in the latent space.

References

[1] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020. 1

[2] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, pages 9729–9738, 2020. 1

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 1

[4] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. *NeurIPS*, 33:12104–12114, 2020. 1

[5] Cheng-Han Lee, Ziwei Liu, Lingyun Wu, and Ping Luo. Maskgan: Towards diverse and interactive facial image manipulation. In *CVPR*, 2020. 4

[6] Takeru Miyato and Masanori Koyama. cgans with projection discriminator. *arXiv preprint arXiv:1802.05637*, 2018. 1

[7] Kuniaki Saito, Kate Saenko, and Ming-Yu Liu. Coco-funit: Few-shot unsupervised image translation with a content conditioned style encoder. In *ECCV*, pages 382–398. Springer, 2020. 3

[8] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *JMLR*, 9(11), 2008. 1

[9] Zongze Wu, Dani Lischinski, and Eli Shechtman. Stylespace analysis: Disentangled controls for stylegan image generation. In *CVPR*, pages 12863–12872, 2021. 4

[10] Peihao Zhu, Rameen Abdal, Yipeng Qin, John Femiani, and Peter Wonka. Improved stylegan embedding: Where are the good latents? *arXiv preprint arXiv:2012.09036*, 2020. 1

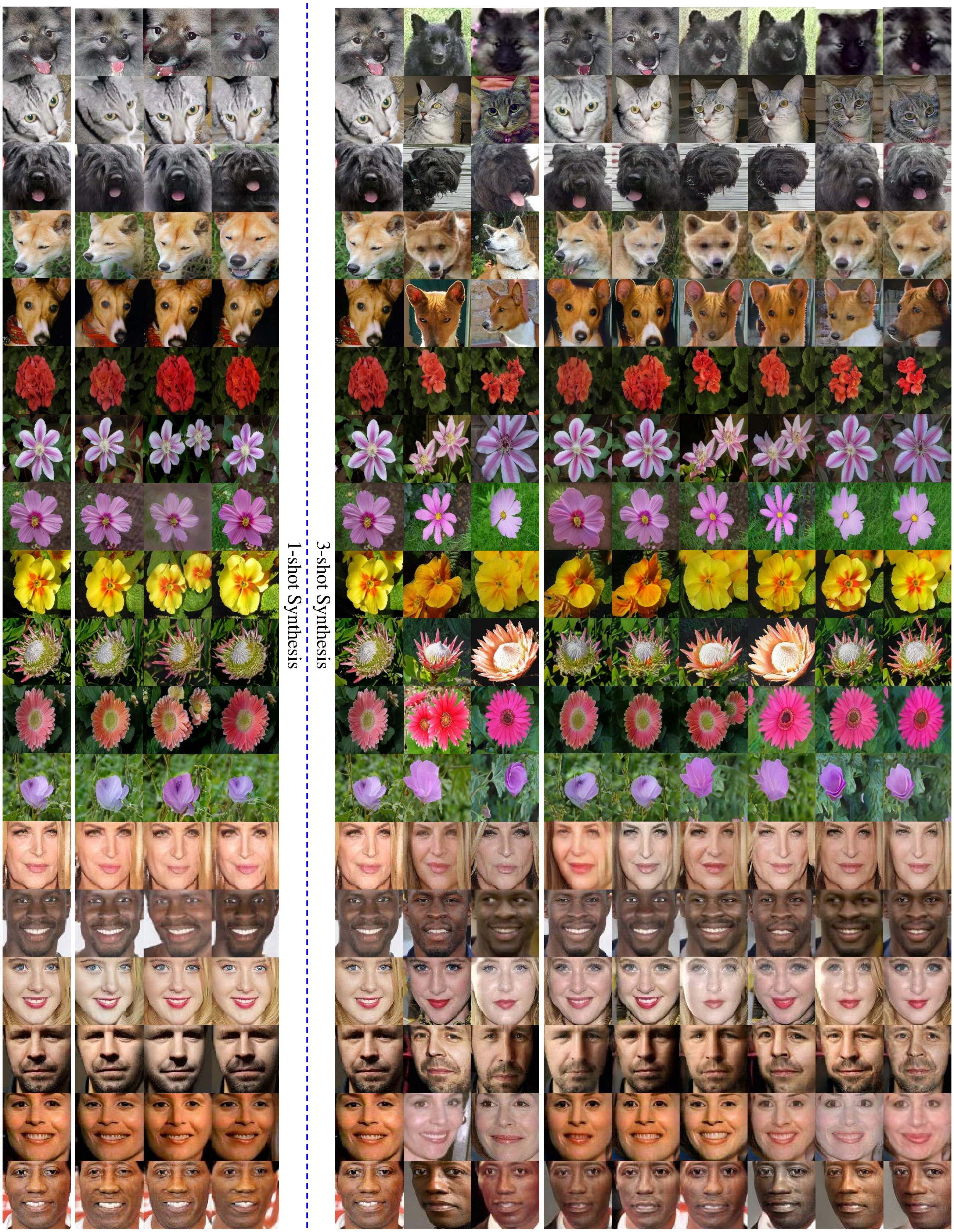


Figure 7. Results under 1-shot and 3-shot settings. The left most column of each part shows the input few-shot images.

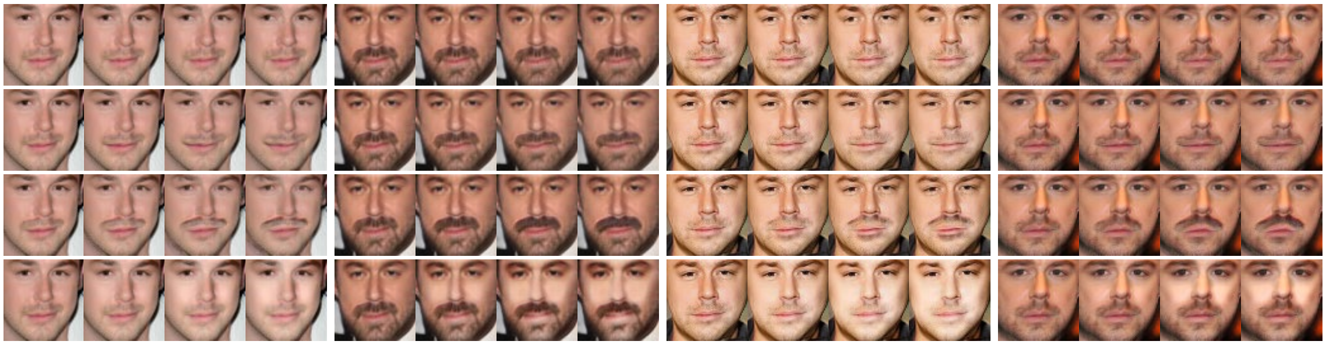


Figure 8. Results of image editing on unseen images. The edited attributes are eye pose, smile, mustache, and illumination from top to bottom.

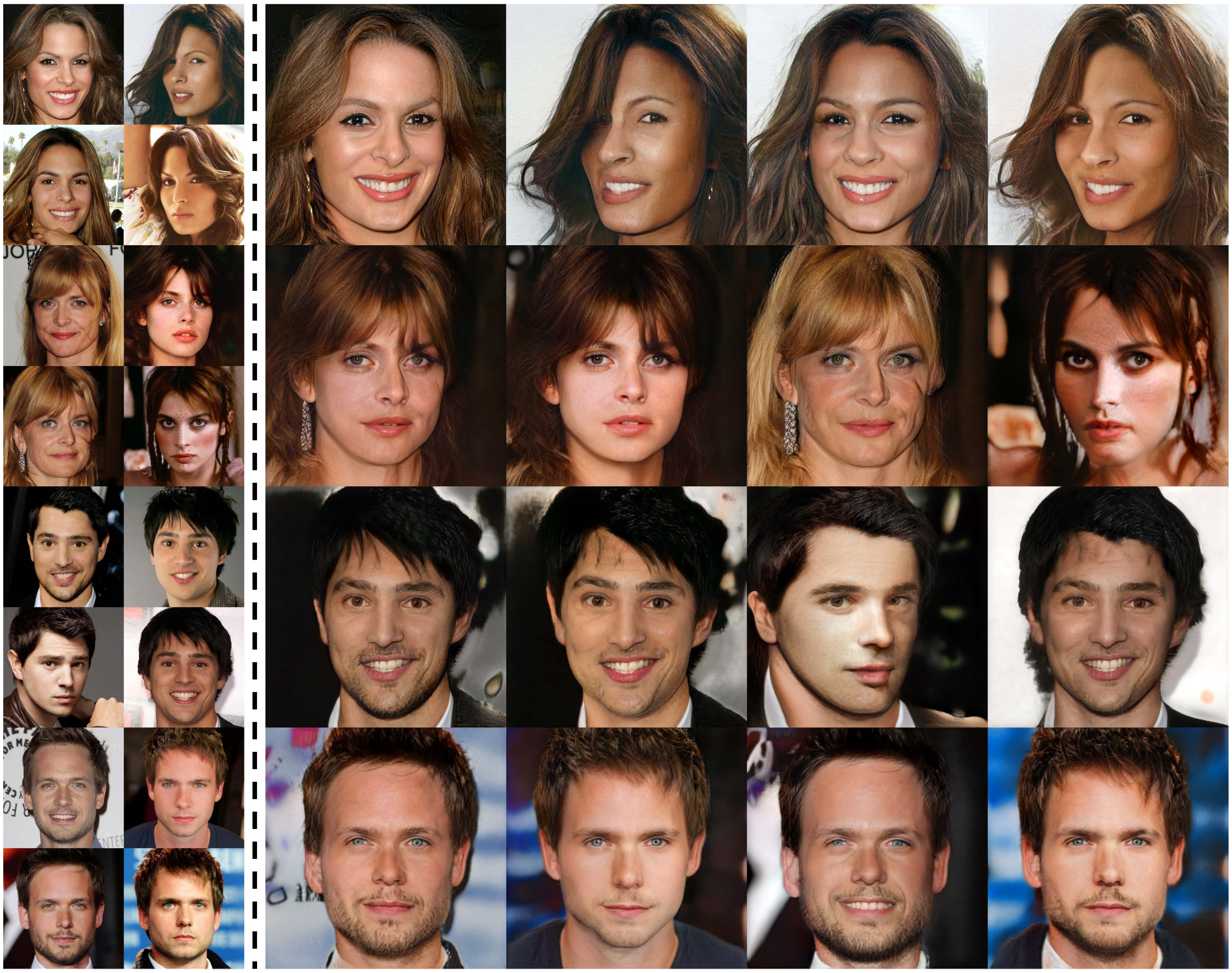


Figure 9. Results of high-quality image generation. The images on the left depict 4-shot image generation tasks, whereas the resulting images are displayed on the right-hand side.

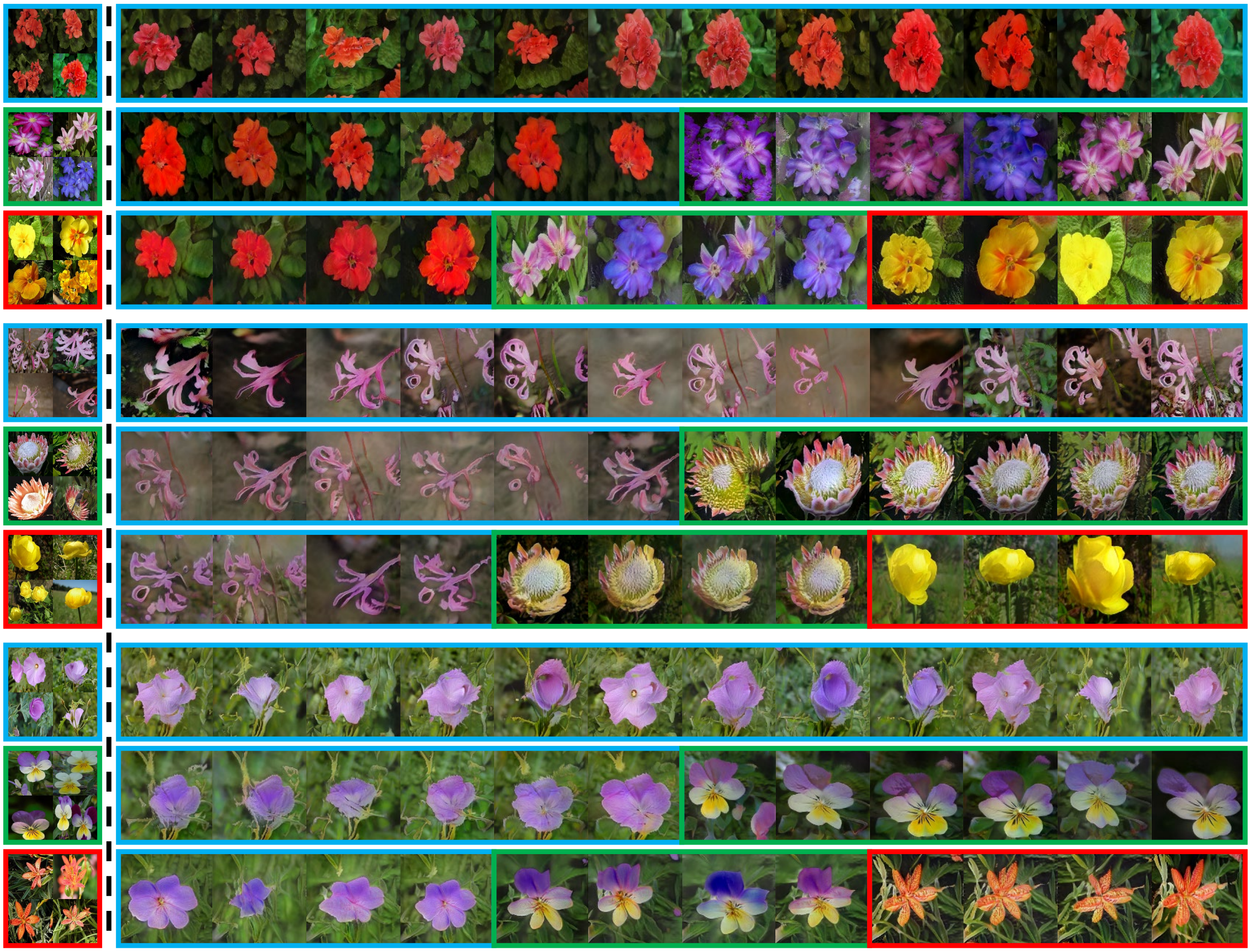


Figure 10. Results of few-shot incremental generation. The leftmost column lists the incremental tasks. We exhibit three incremental procedures with each procedure containing three few-shot tasks. Three sequential tasks with the blue, green, and red bounding boxes are fed to the generator orderly. The incremental generation results are listed on the right.