BlendGAN: Implicitly GAN Blending for Arbitrary Stylized Face Generation

Anonymous Author(s)
Affiliation
Address
email

Abstract

Generative Adversarial Networks (GANs) have made a dramatic leap in high-fidelity image synthesis and stylized face generation. Recently, a layer-swapping mechanism has been developed to improve the stylization performance. However, this method is incapable of fitting arbitrary styles in a single model and requires hundreds of style-consistent training images for each style. To address the above issues, we propose BlendGAN for arbitrary stylized face generation by leveraging a flexible blending strategy and a generic artistic dataset. Specifically, we first train a self-supervised style encoder on the generic artistic dataset to extract the representations of arbitrary styles. In addition, a weighted blending module (WBM) is proposed to blend face and style representations implicitly and control the arbitrary stylization effect. By doing so, BlendGAN can gracefully fit arbitrary styles in a unified model while avoiding case-by-case preparation of style-consistent training images. To this end, we also present a novel large-scale artistic face dataset AAHQ. Extensive experiments demonstrate that BlendGAN outperforms state-of-the-art methods in terms of visual quality and style diversity for both latent-guided and reference-guided stylized face synthesis.

1 Introduction

Generative adversarial networks (GANs) [1] have achieved impressive performance in various tasks such as image generation [2, 3, 4, 5, 6, 7, 8, 9], image super-resolution [10, 11], and image translation [12, 13, 14, 15, 16, 17, 18, 19]. In recent years, GAN has also been widely used for face stylization such as portrait drawing [20], caricature [21, 22], manga [23], and anime [24]. With the deepening of GAN research, the community has further paid attention to the quality of generated images and the disentanglement ability of latent space.

To achieve high-quality generation and latent disentanglement, StyleGAN [8] introduces the adaptative instance normalization layer (AdaIN) [25] and proposes a new generator architecture, which achieves pretty good performance in generating face images. StyleGAN2 [9] explores the causes of droplet-like artifacts and further improves the quality of generated images by redesigning generator normalization. StyleGAN2-ada [26] proposes an adaptive discriminator augmentation mechanism which reduces the required number of images to several thousands. In the experiment of StyleGAN2-ada, the model is trained on MetFaces dataset [26] to generate artistic faces, however, the style of image is uncontrollable, and the corresponding original face cannot be obtained.

Recently, a layer-swapping mechanism [27] has been proposed to generate high-quality natural and stylized face image pairs, and equipped with an additional encoder for unsupervised image translation

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

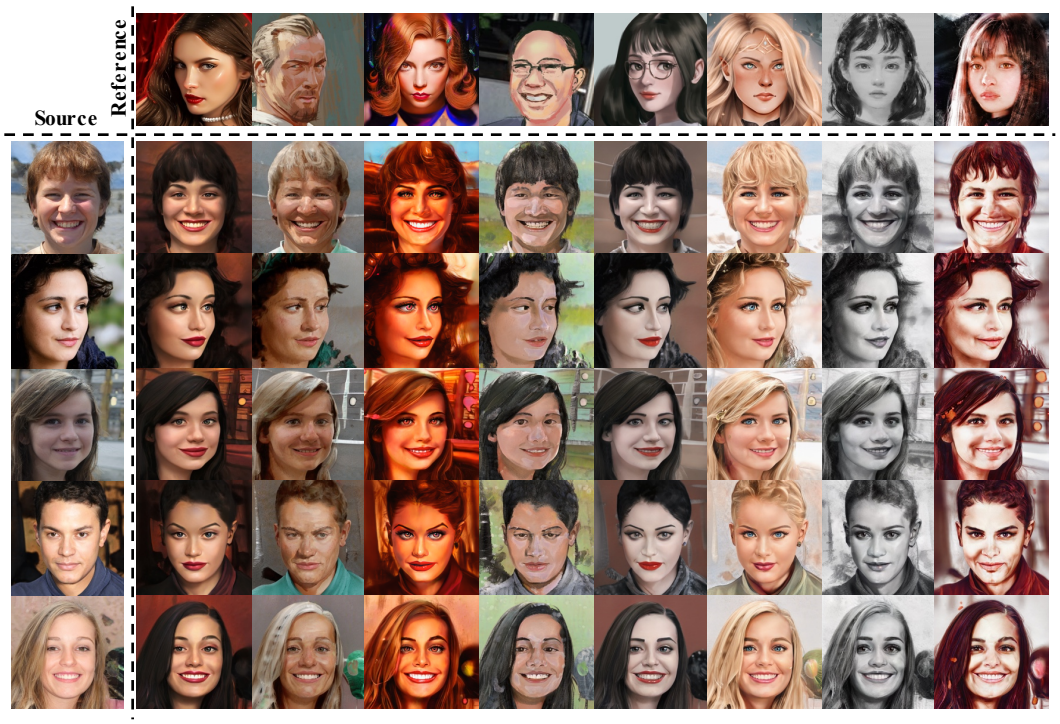

Figure 1: Illustration of some reference-guided synthesis results. Our framework can generate stylized face images with high quality.

[28]. In particular, these methods first finetune StyleGAN from a pretrained model with target-style faces and then swap some convolutional layers with the original model. Given randomly sampled latent codes, the original model generates natural face images, while the layer-swapped model outputs stylized face images correspondingly. Though effective at large, these methods have to train models in a case-by-case flavor, thus a single model can only generate images with a specific style. Besides, these methods still require hundreds of target-style images for finetuning, and one has to carefully select the number of training iterations, as well as the swapped layers.

In this paper, we present BlendGAN for arbitrary stylized face generation.[1] BlendGAN resorts to a flexible blending strategy and a generic artistic dataset to fit arbitrary styles without relying on style-consistent training images for each style. In particular, we analyze a stylized face image as composed of two latent parts: a face code (controlling the face attributes) and a style code (controlling the artistic appearance). Firstly, a self-supervised style encoder is trained via an instance discrimination objective to extract the style representation from artistic face images. Secondly, a *weighted blending module* (WBM) is proposed to blend the face and style latent codes into a final code which is then fed into a generator. By controlling the indicator in WBM, we are able to decide which parts of the face and style latent codes to be blended thus controlling the stylization effect. By combining the style encoder and the generator with WBM, our framework can generate natural and stylized face image pairs with either a randomly sampled style or a reference style (see Figure 1).

As for the generic artistic data, we present a novel large-scale dataset of high-quality artistic face images, Artstation-Artistic-face-HQ (AAHQ), which covers a wide variety of painting styles, color tones, and face attributes. We will release this dataset for the research community. Experiments show that compared to state-of-the-art methods, our framework can generate stylized face images with higher visual quality and style diversity for both latent-guided and reference-guided synthesis.

[1] Our framework can also cooperate with GAN inversion [29, 30, 31, 32, 33] or StyleGAN distillation [34] methods to enable end-to-end style transfer or image translation. We will show the results in the appendix.

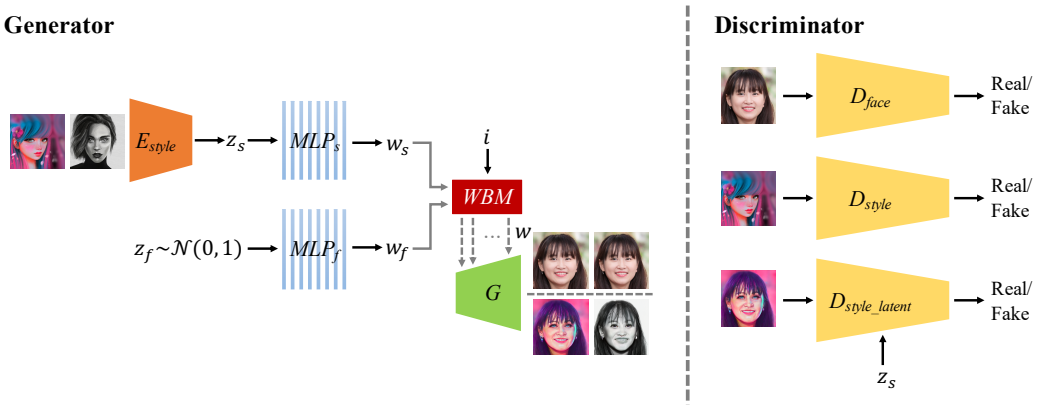

Figure 2: Overview of the proposed framework. The style encoder E_{style} extracts the style latent code $\mathbf{z}_s$ of a reference style image. The face latent code $\mathbf{z}_f$ is randomly sampled from the standard Gaussian distribution. Two MLPs transform face and style latent codes into their W spaces separately, then they are combined by the *weighted blending module (WBM)* and fed into generator G to synthesise natural and stylized face images. Three discriminators are used in our method. The face discriminator D_{face} distinguishes between real and fake natural-face images, the style discriminator D_{style} distinguishes between real and fake stylized-face images, and the style latent discriminator D_{style_latent} predicts whether the stylized-face image is consistent with the style latent code $\mathbf{z}_s$.

2 Related Work

Generative Adversarial Networks The recent years have witnessed rapid advances of generative adversarial networks (GANs) [1] for image generation. The key to the success of GANs is the adversarial training between the generator and discriminator. Various improvements to GANs have been proposed to stabilize GAN training and improve the image quality. WGAN [3] uses the Wasserstein distance as a new cost function to train the network. WGAN-GP [4] and SNGAN [5] improve the stability of GAN by introducing gradient penalty and spectral normalization separately to make the training satisfy the 1-Lipschitz constraint. SAGAN [35] enlarges the receptive field using the self-attention mechanism. BigGAN [6] shows the significant improvement of image quality when training GANs at large scales. StyleGAN [8, 9] redesigns the generator architecture with AdaIN layers, making it better disentangle the latent factors of variation. In this work, we employ GANs to generate high-quality stylized face images.

Image-to-Image Translation With the good performance of GANs, many GAN-based image-to-image translation techniques have been explored in recent years [12, 13, 14, 16, 17, 36, 18, 37, 19]. For image translation in two domains, Pix2pix [12] proposes the first unified image-to-image framework to translate images to another domain. CycleGAN [13] proposes a cycle-consistency loss, making the network train with unpair data. UNIT [14] maps images in source and target domains to a shared latent space to perform unsupervised image translation. UGATIT [24] proposes an adaptive layer-instance normalization (AdaLIN) layer to control shapes during translation. For image translation in multi-domains, MUNIT [16] extends UNIT to multi-modal contexts by decomposing images into content and style spaces. FUNIT [17] leverages a few-shot strategy to translate images using a few reference images from a target domain. DRIT++ [18] also proposes a multi-modal and multi-domain model using a discrete domain encoding. Training with both latent codes and reference images, StarGANv2 [19] could provide both latent-guided and reference-guided synthesis. Although these methods could translate images to multiple target domains, the diversity of target domains is still limited and these methods require hundreds of training images for each domain. Besides, the above methods could only operate at a resolution of at most 256x256 when applied to human faces.

Neural Style Transfer The arbitrary stylized face generation task could also be regarded as a kind of the neural style transfer [38, 39, 40, 25]. Gatys *et al.* [38], for the first time, proposes a

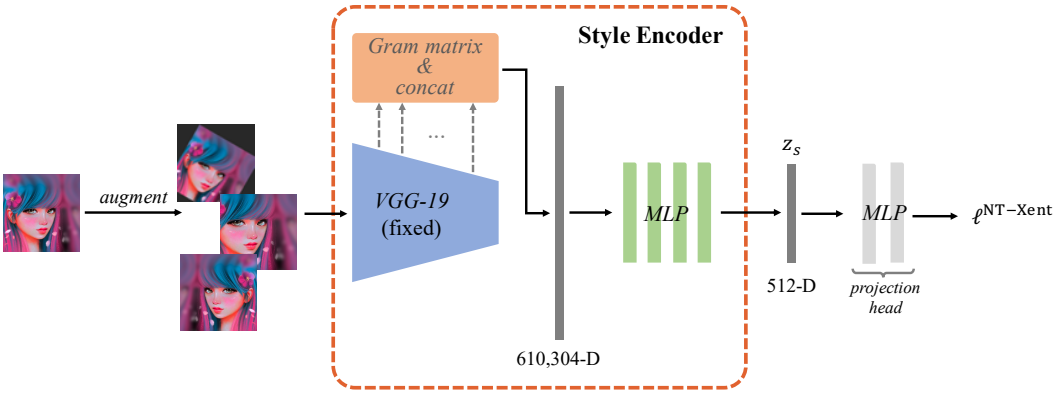

Figure 3: Architecture of the proposed style encoder, which consists of a pretrained *VGG-19* [44] network, a *Gram matrix and concat* module and an MLP predictor. More detailed notations are described in the contexts of Sec. 3.1.

neural style transfer method based on optimization strategy. Johnson *et al.* [39] trains a feed-forward network to avoid the time consumption of optimization. Luan *et al.* [41] proposes a photorealism regularization term to preserve the structure of source images. The above methods could only transfer source images to a single style with a model, many works extend the style flexibility and achieved multi-style or arbitrary style transfer such as MSG-Net [40], AdaIN [25], WCT [42] and SANet [43]. Although arbitrary style transfer methods could transfer images to arbitrary styles, they only transfer the reference style to source images globally without considering local semantic styles, which leads to global texture artifacts in the outputs.

3 Method

Our goal is to train a generator

$$\hat{x}_f, \hat{x}_s = G(\mathbf{z}_f, \mathbf{z}_s, i), \tag{1}$$

that can generate a image pair (natural face image $\hat{x}_f$ and its artistic counterpart $\hat{x}_s$) with a pair of given latent code $(\mathbf{z}_f, \mathbf{z}_s)$ and a blending indicator i. The face latent code $\mathbf{z}_f$ controls the face identity, and the style latent code $\mathbf{z}_s$ controls the style of $\hat{x}_s$. The blending indicator i decides which parts of $\mathbf{z}_f$ and $\mathbf{z}_s$ are blended to generate plausible images. Figure 2 illustrates an overview of our framework.

3.1 Self-supervised Style Encoder

The style encoder E_{style} is independently trained to extract the style representation from an artistic face image while discarding the content information such as face identity. Many style transfer methods [38, 39, 40] obtain the style representation by calculating the Gram matrices of feature maps extracted by a pretrained VGG [44] network. However, if simply concatenating the Gram matrices of all the selected feature maps, the resulting style latent code ('style embedding') would have a very high dimension. For example, if we choose layers $relu1_2$, $relu2_2$, $relu3_4$, $relu4_4$ and $relu5_4$, the style embedding would be a 610,304-dimensional vector ($64^2 + 128^2 + 256^2 + 512^2 * 2$). Higher latent code dimension leads to sparser distribution, which makes the generator harder to train and perform worse in style disentanglement. Hence, following the self-supervised method SimCLR [45, 46], we train the style encoder to reduce the dimension of style embeddings.

Figure 3 illustrates the architecture of our style encoder. The style image is augmented and then sent to a pretrained VGG [44] network. We calculate the Gram matrices of selected feature maps, flatten and concatenate them into a long vector (610,304-D in our case). Then the vector is passed into a 4-layer MLP module (predictor) to reduce the dimension and generate the style embedding $\mathbf{z}_s$ of the input style image. Following $\mathbf{z}_s$, another 2-layer MLP performs as a projection head for

contrastive learning. The augmentations in the original SimCLR include affine transformation and color transformation, following the assumption that affine and color transformations do not change the classes of objects. However, for style encoding, the image style is strongly related to color, so we only use affine transformations in the augmentation step. During training the style encoder, only parameters of two MLP modules are optimized while *VGG-19* is fixed to perform as a feature extractor. The network is trained using NT-Xent loss [45]:

$$\ell_{i,j}^{\mathrm{NT-Xent}} = -\log \frac{\exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)}, \tag{2}$$

where (i, j) is a pair of positive examples (augmented from the same image), $\boldsymbol{z}$ is the projection result, $\mathrm{sim}(\cdot, \cdot)$ is cosine similarity between two vectors, and τ is a temperature scalar.

3.2 Generator

The generator G is similar to StyleGAN2 [9], but its input is a combination of two latent codes - face latent code $\mathbf{z}_f$ and style latent code $\mathbf{z}_s$. In particular, $\mathbf{z}_f$ controls the face identity of the generated images, and is randomly sampled from the standard Gaussian distribution $\mathcal{N}(0, 1)$. $\mathbf{z}_s$ controls the style of the generated stylized image, it could be either randomly sampled from $\mathcal{N}(0, 1)$ or encoded by the style encoder E_{style} from a reference artistic image. Two MLPs transform $\mathbf{z}_f$ and $\mathbf{z}_s$ into their W spaces as $\mathbf{w}_f$ and $\mathbf{w}_s$ separately. For 1024×1024 output, the dimension of $\mathbf{w}_f$ and $\mathbf{w}_s$ are all 18×512. Controlled by the blending indicator i, the *weighted blending module (WBM)* combines $\mathbf{w}_s$ and $\mathbf{w}_f$ into $\mathbf{w}$, which is then fed into the generator as the final code.

WBM Different resolution layers in the StyleGAN model are responsible for different features in the generated image [8] (e.g. low resolution layers control face shape and hair style, high resolution layers control color and lighting). Therefore, the blending weights of $\mathbf{w}_s$ and $\mathbf{w}_f$ should not be consistent at different layers. We propose *WBM* to blend the two codes, which can be described as:

$$\mathbf{w} = \mathbf{w}_s \odot \hat{\alpha} + \mathbf{w}_f \odot (\mathbb{1} - \hat{\alpha}), \tag{3}$$

$$\hat{\alpha} = \alpha \odot \mathbf{m}(i), \tag{4}$$

$$\mathbf{m}(i; \theta) = [m_0, m_1, m_2, \ldots, m_n]\,,\ m_j = \begin{cases} 0 & j < i \\ \theta & j = i \\ 1 & j > i \end{cases},\ \ \theta \in (0, 1), \tag{5}$$

where $\odot$ denotes the broadcasting element-wise product, $\mathbf{w}_s, \mathbf{w}_f \in \mathbb{R}^{18 \times 512}$ are the input codes, $\alpha \in \mathbb{R}^{18}$ is a learnable vector to balance the blending weights for $\mathbf{w}_s$ and $\mathbf{w}_f$ in different layers. $\mathbf{m}(i; \theta)$ has the same dimension as α, it controls which layer should be blended. If $i = 0$, $\mathbf{w} = \mathbf{w}_s \odot \alpha + \mathbf{w}_f \odot (\mathbb{1} - \alpha)$, the generator outputs stylized face image $\hat{x}_s$; if $i = 18$, $\mathbf{w} = \mathbf{w}_f$, the generator outputs natural face image $\hat{x}_f$. When $0 < i < 18$, some low resolution layers are not influenced by style codes, which ensures the $\hat{x}_s$ to keep the same face identity as $\hat{x}_f$ (see Figure 2). θ is used for finer adjustment.

3.3 Discriminator

There are three discriminators in our framework(see the right column in Figure 2). The face discriminator D_{face} and the style discriminator D_{style} have the same architecture as in StyleGAN2 [9]. D_{face} distinguishes between real and fake natural-face images, it only receives images sampled from the natural-face dataset (FFHQ) or generated by G when $i = 0$. In the contrast, D_{style} distinguishes between real and fake stylized-face images and it only receives images sampled from the artistic face dataset (AAHQ) or generated by G when $i = 18$. The style latent discriminator D_{style_latent} has two inputs - a generated stylized-face image $\hat{x}_s$ and a style latent code $\mathbf{z}_s$, it predicts whether $\hat{x}_s$ is consistent with $\mathbf{z}_s$. During training the network, we use an embedding queue to storage the style latent codes of the previously sampled style images embedded by E_{style}, and randomly sample one as the fake $\mathbf{z}_s$.

3.4 Training objectives

Given a natural-face image $x_f \in \mathcal{X}_{natural}$, a stylized-face image $x_s \in \mathcal{X}_{style}$ and a randomly sampled face latent code $\mathbf{z}_f \sim \mathcal{N}(0, 1)$, we train our framework using the following adversarial objectives.

For the face discriminator, the blending indicator $i = 0$. G only takes $\mathbf{z}_f$ as the input and learns to generate a natural-face image $\hat{x}_f = G_{i=0}(\mathbf{z}_f)$ via an adversarial loss

$$\mathcal{L}_{face} = \mathbb{E}_{x_f}[\log D_{face}(x_f)] + \mathbb{E}_{\mathbf{z}_f}[\log(1 - D_{face}(G_{i=0}(\mathbf{z}_f)], \tag{6}$$

where $D_{face}(\cdot)$ denotes the output of the face discriminator.

For the style discriminator, the blending indicator $i = 18$. G takes $\mathbf{z}_f$ and x_s as inputs, and generate a stylized-face image $\hat{x}_s$ which has the same style as x_s. The loss is described as

$$\mathcal{L}_{style} = \mathbb{E}_{x_s}[\log D_{style}(x_s)] + \mathbb{E}_{\mathbf{z}_f, x_s}[\log(1 - D_{style}(G_{i=18}(\mathbf{z}_f, E_{style}(x_s)))], \tag{7}$$

where the style encoder E_{style} extracts the style latent code of x_s, and $D_{style}(\cdot)$ denotes the output of the style discriminator.

For the style latent discriminator, we denote $\mathbf{z}_s = E_{style}(x_s)$ as the style latent code of x_s, and randomly sample another style latent code $\mathbf{z}_s^-$ from the embedding queue as a negative sample, then the loss could be described as

$$\mathcal{L}_{style_latent} = \mathbb{E}_{\mathbf{z}_f, x_s}[\log D_{style_latent}(x_s, \mathbf{z}_s)] + \mathbb{E}_{\mathbf{z}_f, x_s}[\log(1 - D_{style_latent}(\hat{x}_s, \mathbf{z}_s^-)]. \tag{8}$$

Consequently, we combine all the above loss functions as our full objective as follows:

$$\mathcal{L}_G = \mathcal{L}_{face} + \mathcal{L}_{style} + \mathcal{L}_{style_latent}, \tag{9}$$

$$\mathcal{L}_D = -\mathcal{L}_G. \tag{10}$$

4 Experiments

Our proposed method is able to generate arbitrary stylized face images with high quality and diversity. In this section, we describe evaluation setups and test our method both qualitatively and quantitatively on a large amount of images spanning a large range of face and style varieties.

Datasets As described in Sec. 3, for arbitrary stylized face generation, we need a natural-face dataset and an artistic face dataset to train the networks. We use FFHQ [8] as the natural-face dataset, which includes 70,000 high-quality face images[2]. In addition, we build a new dataset of artistic-face images, Artstation-Artistic-face-HQ (AAHQ), consisting of 33,245 high-quality artistic faces at 1024^2 resolution (Figure 4). The dataset covers a wide variety in terms of painting styles, color tones, and face attributes. The artistic images are crawled from Artstation[3] (thus inheriting all the biases of that website) and automatically aligned and cropped as FFHQ. Finally, we manually remove images without faces or with low quality. More details of this dataset can be found in the appendix.

Baselines We compare our model with several leading baselines on diverse image synthesis including AdaIN [25], MUNIT [16], FUNIT [17], DRIT++ [18], and StarGANv2 [19]. All of these methods could synthesise images in different modalities through the control of latent codes or reference images. All the baselines are trained on FFHQ and AAHQ using the open-source implementations provided by the authors, and our BlendGAN code is based on an unofficial PyTorch implementation of StyleGAN2[4].

[2]The FFHQ dataset is under Creative Commons BY-NC-SA 4.0 license.

[3]https://www.artstation.com

[4]https://github.com/rosinality/stylegan2-pytorch

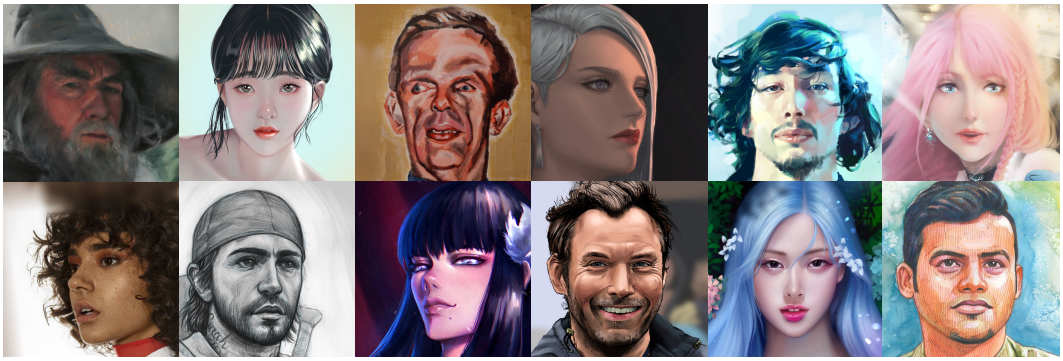

Figure 4: The AAHQ dataset offers a lot of variety in terms of painting styles, color tones and face attributes.

Table 1: FID and LPIPS comparison with different blending indicators.

Indicator i		0	2	4	6	8	10	12	14	16	18
latent	FID ↓	**8.97**	12.45	12.75	23.17	42.45	57.34	68.31	76.91	78.25	76.73
	LPIPS ↑	**0.581**	0.571	0.568	0.515	0.459	0.367	0.304	0.207	0.145	0.159
reference	FID ↓	**3.79**	6.39	6.82	15.08	34.33	51.49	63.73	76.00	77.11	76.97
	LPIPS ↑	**0.661**	0.651	0.650	0.599	0.540	0.450	0.377	0.237	0.191	0.160

Evaluation metrics To evaluate the quality of our results, we use Frechet inception distance (FID) metric [47] to measure the discrepancy between the generated images and AAHQ dataset. A lower FID score indicates that the distribution of generated images are more similar to that of AAHQ dataset and the generated images are more plausible as real stylized-face images. In addition, we adopt the learned perceptual image patch similarity (LPIPS) metric [48] to measure the style diversity of generated images. Following [19], as for a face latent code (corresponding to a natural-face image), we randomly sample 10 style latent codes or reference style images to generate 10 outputs and evaluate the LPIPS scores between every 2 outputs. We randomly sample 1000 face latent codes, repeat the above process and average all of the scores as the final LPIPS score. A higher LPIPS score indicates that the model could generate images with larger style diversity.

4.1 Blending indicator

As described in Sec. 3.2, the indicator i in WBM controls in which layers the weights of $\mathbf{w}_s$ and $\mathbf{w}_f$ should be blended. In StyleGAN generator, low-resolution layers are responsible for face shape and high-resolution layers control color and lighting. Hence, as i gets larger, less low-resolution layers are influenced by style codes, and the face shapes of stylized outputs are more similar to the natural-face images. As shown in Figure 5, when $i = 0$, the generated images have the strongest style as well as the most different face identities from the corresponding natural-face images. When $i = 18$, the generated images completely lose the style information and become natural-face images. Table 1 also shows that the style quality and diversity of the generated images decrease with the increase of i. We notice that when $i = 6$, the generated images have a good balance between stylization strength and face shapes consistency with the natural-face images. To this end, we set $i = 6$ in the qualitative experiments, while we use both $i = 0$ and $i = 6$ in the quantitative experiments.

4.2 Comparison on arbitrary stylized face generation

Our framework can generate stylized-face images with two mechanisms: latent-guided generation and reference-guided generation.

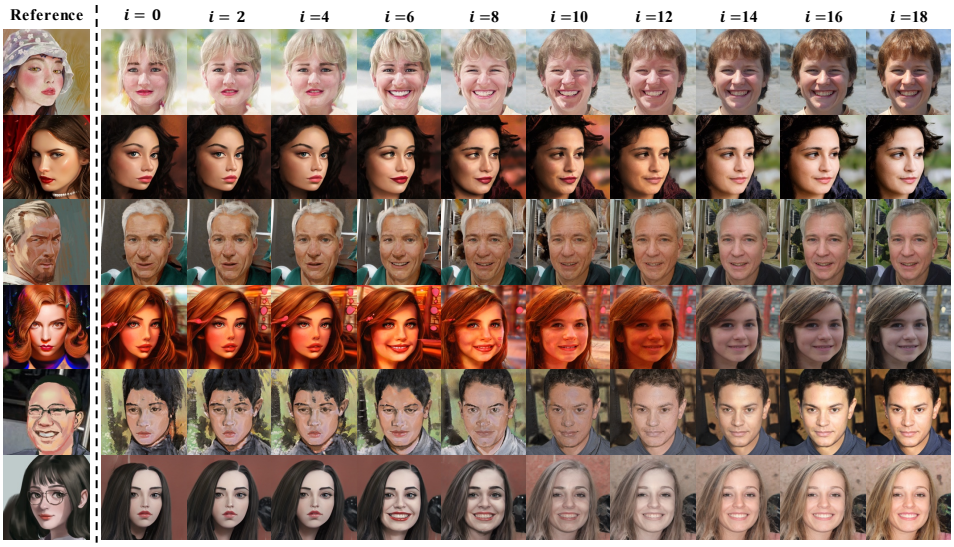

Figure 5: Reference-guided stylized-face images with different blending indicators.

Table 2: Quantitative comparison on latent-guided stylized-face generation.

Method	FID ↓	LPIPS ↑
MUNIT	29.69	0.394
FUNIT	176.96	0.500
DRIT++	22.53	0.448
StarGANv2	50.20	0.312
BlendGAN ($i = 6$)	23.17	0.515
BlendGAN ($i = 0$)	**8.97**	**0.581**

Table 3: Quantitative comparison on reference-guided stylized-face generation.

Method	FID ↓	LPIPS ↑
AdaIN	37.23	0.345
MUNIT	103.61	0.192
FUNIT	87.71	0.327
DRIT++	31.71	0.241
StarGANv2	50.03	0.307
BlendGAN ($i = 6$)	15.08	0.599
BlendGAN ($i = 0$)	**3.79**	**0.661**

Latent-guided generation. For latent-guided generation, the style latent code $\mathbf{z}_s$ is randomly sampled from the standard Gaussian distribution. Figure 6 shows a qualitative comparison of the competing methods. The results of AdaIN [25] are not shown because it is not able to take latent codes as the style inputs. Since FUNIT is designed for few-shot image-to-image translation, it could not generate face images when inputting randomly sampled style latent codes. The results of MUNIT and DRIT++ have severe visible artifacts and they cannot reserve face identities in some images. StarGANv2 has the best performance among the baselines but the generated images still have some subtle artifacts and are not as natural as our results. We observe that the generated stylized-face images of our method have the highest visual quality, and well preserve the face identities of the source images. Quantitative comparison is shown in Table 2. Although when $i = 6$, the FID score of our method is slightly higher than DRIT++, the qualitative comparison in Figure 6 shows that our stylized images have less artifacts and higher visual quality. When $i = 0$, our method has the lowest FID score and the highest LPIPS, indicating that our model could generate stylized images with the best visual quality and style diversity.

Reference-guided generation. For reference-guided generation, the style latent code $\mathbf{z}_s$ is embedded by the style encoder E_{style} from a reference artistic face image. Figure 7 illustrates qualitative comparison results. The results show that the style transfer method AdaIN can not consider semantic style information and introduces severe visible artifacts in their generated stylized faces. FUNIT and DRIT++ also lead to visible artifacts in some images, as shown in the first and fourth rows. Although MUNIT allows to input a reference image at the testing stage, it only transfers the average color of the reference images, and the results do not have similar textures to the references. StarGANv2 has the

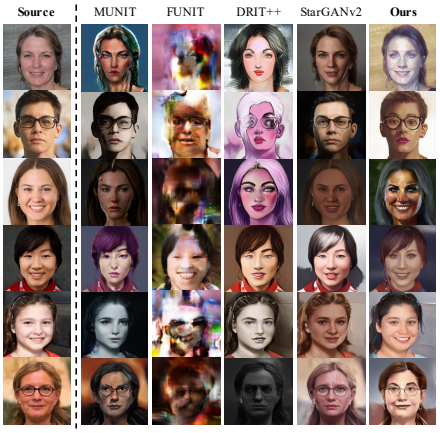

Figure 6: Comparison of latent-guided generation. The source images are generated by our model with indicator $i = 0$, and the style latents are randomly sampled from $\mathcal{N}(0, 1)$. From left to right: the source image, the results of MUNIT [16], FUNIT [17], DRIT++ [18], StarGANv2 [19] and our BlendGAN. Digital zoom-in recommended.

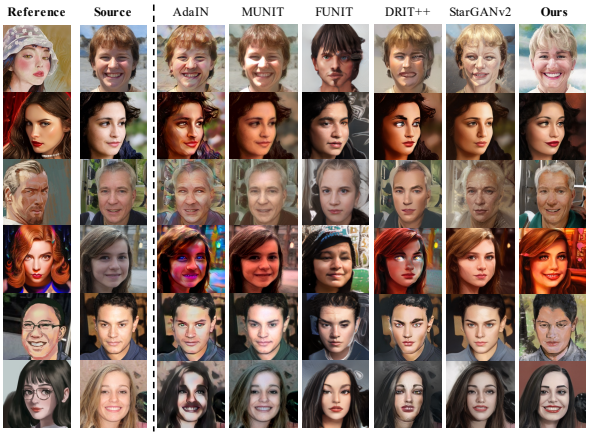

Figure 7: Comparison of reference-guided generation. The source images are generated by our model with indicator $i = 0$, and the reference images are sampled from the AAHQ dataset. From left to right: the reference artistic-face image, the source natural-face image, the results of AdaIN [25], MUNIT [16], FUNIT [17], DRIT++ [18], StarGANv2 [19] and our BlendGAN. Digital zoom-in recommended.

best performance among the baselines, however, the style of some generate images is not consistent with their references, as shown in the fourth and fifth rows. As a comparison, our BlendGAN generates stylized-face images with the highest quality and the style of generated images is the most consistent with the references thanks to our well-designed training strategy. Quantitative comparison is shown in Table 3. For $i = 0$ and $i = 6$, our method achieves FID of 3.79 and 15.08, which outperform all the previous leading methods by a large margin. This implies that images generated by our model are the most similar to real stylized-face images. The LPIPS of our method is also the highest, indicating that our model can produce faces with the most diverse style considering the reference images. Besides, it is worth noting that our method with $i = 0$ performs better than $i = 6$ because low-resolution layers are not affected by indicator i.

5 Conclusions

In this paper, we propose a novel framework for arbitrary stylized face generation. Training with FFHQ and a new large-scale artistic face dataset (AAHQ), our framework can generate infinite arbitrary stylized faces as well as their unstylized counterparts, and the generation could be either latent-guided or reference-guided. Specifically, we propose a self-supervised style encoder to extract the style representation from reference images, as well as a weighted blending module to implicitly combine the face and style latent codes for controllable generation. The experimental results show that our proposed model could generate images with high quality and style diversity, and the generated images have good style consistency with reference images, remarkably outperforming the previous leading methods. Our experiments are currently only performed on face datasets, future work includes extending this framework to other domains, such as stylized landscape generation.

Broader Impact

This work would be of great interest to the general community. It shows the great potential of applying GAN models to stylized image generation, which could be further used for other tasks, such as style transfer, style manipulation, and new style creation. On the other hand, as our method is trained using FFHQ[8] and AAHQ datasets, it inherits all the biases of these datasets.

References

[1] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014. 1, 3

[2] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014. 1

[3] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017. 1, 3

[4] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 5769–5779, 2017. 1, 3

[5] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018. 1, 3

[6] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2018. 1, 3

[7] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018. 1

[8] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019. 1, 3, 5, 6, 9

[9] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8110–8119, 2020. 1, 3, 5

[10] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017. 1

[11] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy. Esrgan: Enhanced super-resolution generative adversarial networks. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018. 1

[12] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017. 1, 3

[13] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017. 1, 3

[14] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 700–708, 2017. 1, 3

[15] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2337–2346, 2019. 1

[16] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 172–189, 2018. 1, 3, 6, 9

[17] Ming-Yu Liu, Xun Huang, Arun Mallya, Tero Karras, Timo Aila, Jaakko Lehtinen, and Jan Kautz. Few-shot unsupervised image-to-image translation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10551–10560, 2019. 1, 3, 6, 9

[18] Hsin-Ying Lee, Hung-Yu Tseng, Qi Mao, Jia-Bin Huang, Yu-Ding Lu, Maneesh Kumar Singh, and Ming-Hsuan Yang. Drit++: Diverse image-to-image translation viadisentangled representations. *International Journal of Computer Vision*, pages 1–16, 2020. 1, 3, 6, 9

[19] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 1, 3, 6, 7, 9

[20] Ran Yi, Yong-Jin Liu, Yu-Kun Lai, and Paul L Rosin. Apdrawinggan: Generating artistic portrait drawings from face photos with hierarchical gans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10743–10752, 2019. 1

[21] Kaidi Cao, Jing Liao, and Lu Yuan. Carigans: unpaired photo-to-caricature translation. *ACM Transactions on Graphics (TOG)*, 37(6):1–14, 2018. 1

[22] Yichun Shi, Debayan Deb, and Anil K Jain. Warpgan: Automatic caricature generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10762–10771, 2019. 1

[23] Hao Su, Jianwei Niu, Xuefeng Liu, Qingfeng Li, Jiahe Cui, and Ji Wan. Unpaired photo-to-manga translation based on the methodology of manga drawing. *arXiv preprint arXiv:2004.10634*, 2020. 1

[24] Junho Kim, Minjae Kim, Hyeonwoo Kang, and Kwanghee Lee. U-gat-it: unsupervised generative attentional networks with adaptive layer-instance normalization for image-to-image translation. *arXiv preprint arXiv:1907.10830*, 2019. 1, 3

[25] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1501–1510, 2017. 1, 3, 4, 6, 8, 9

[26] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. In *Proc. NeurIPS*, 2020. 1

[27] Justin NM Pinkney and Doron Adler. Resolution dependant gan interpolation for controllable image synthesis between domains. *arXiv preprint arXiv:2010.05334*, 2020. 1

[28] Sam Kwong, Jialu Huang, and Jing Liao. Unsupervised image-to-image translation via pre-trained stylegan2 network. *IEEE Transactions on Multimedia*, 2021. 2

[29] Weihao Xia, Yulun Zhang, Yujiu Yang, Jing-Hao Xue, Bolei Zhou, and Ming-Hsuan Yang. Gan inversion: A survey. *arXiv preprint arXiv:2101.05278*, 2021. 2

[30] Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2stylegan: How to embed images into the stylegan latent space? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4432–4441, 2019. 2

[31] Yujun Shen, Jinjin Gu, Xiaoou Tang, and Bolei Zhou. Interpreting the latent space of gans for semantic face editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9243–9252, 2020. 2

[32] Stanislav Pidhorskyi, Donald A Adjeroh, and Gianfranco Doretto. Adversarial latent autoencoders. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14104–14113, 2020. 2

[33] Elad Richardson, Yuval Alaluf, Or Patashnik, Yotam Nitzan, Yaniv Azar, Stav Shapiro, and Daniel Cohen-Or. Encoding in style: a stylegan encoder for image-to-image translation. *arXiv preprint arXiv:2008.00951*, 2020. 2

[34] Yuri Viazovetskyi, Vladimir Ivashkin, and Evgeny Kashin. Stylegan2 distillation for feed-forward image manipulation. *arXiv preprint arXiv:2003.03581*, 2020. 2

[35] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *International conference on machine learning*, pages 7354–7363. PMLR, 2019. 3

[36] Hsin-Ying Lee, Hung-Yu Tseng, Jia-Bin Huang, Maneesh Kumar Singh, and Ming-Hsuan Yang. Diverse image-to-image translation via disentangled representations. In *European Conference on Computer Vision*, 2018. 3

[37] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 3

[38] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423, 2016. 3, 4

[39] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, 2016. 3, 4

[40] Hang Zhang and Kristin Dana. Multi-style generative network for real-time transfer. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018. 3, 4

[41] Fujun Luan, Sylvain Paris, Eli Shechtman, and Kavita Bala. Deep photo style transfer. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4990–4998, 2017. 4

[42] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming Hsuan Yang. Universal style transfer via feature transforms. *Advances in Neural Information Processing Systems*, 2017:386–396, 2017. 4

[43] Dae Young Park and Kwang Hee Lee. Arbitrary style transfer with style-attentional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5880–5888, 2019. 4

[44] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015. 4

[45] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020. 4, 5

[46] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big self-supervised models are strong semi-supervised learners. *arXiv preprint arXiv:2006.10029*, 2020. 4

[47] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *arXiv preprint arXiv:1706.08500*, 2017. 7

[48] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 7

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See Abstract and Section 1.
   (b) Did you describe the limitations of your work? [Yes] See the last sentence of Section 5.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Broader Impact.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] The code and the data are proprietary, but we will release them after confirmation by our company.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 4 and the supplemental material.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] Error bars are not reported because it would be too computationally expensive.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] They are included in the supplemental material.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes] We mentioned the license of the FFHQ dataset in 4.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No] The assets contain proprietary information, but we will release them after confirmation by our company.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] See Section 4.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] See Section 4.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]