arXiv:2104.14754v1 [cs.CV] 30 Apr 2021

Exploiting Spatial Dimensions of Latent in GAN for Real-time Image Editing

Hyunsu Kim[1,2] Yunjey Choi[1] Junho Kim[1] Sungjoo Yoo[2] Youngjung Uh[3]

[1]NAVER AI Lab [2]Seoul National University [3]Yonsei University

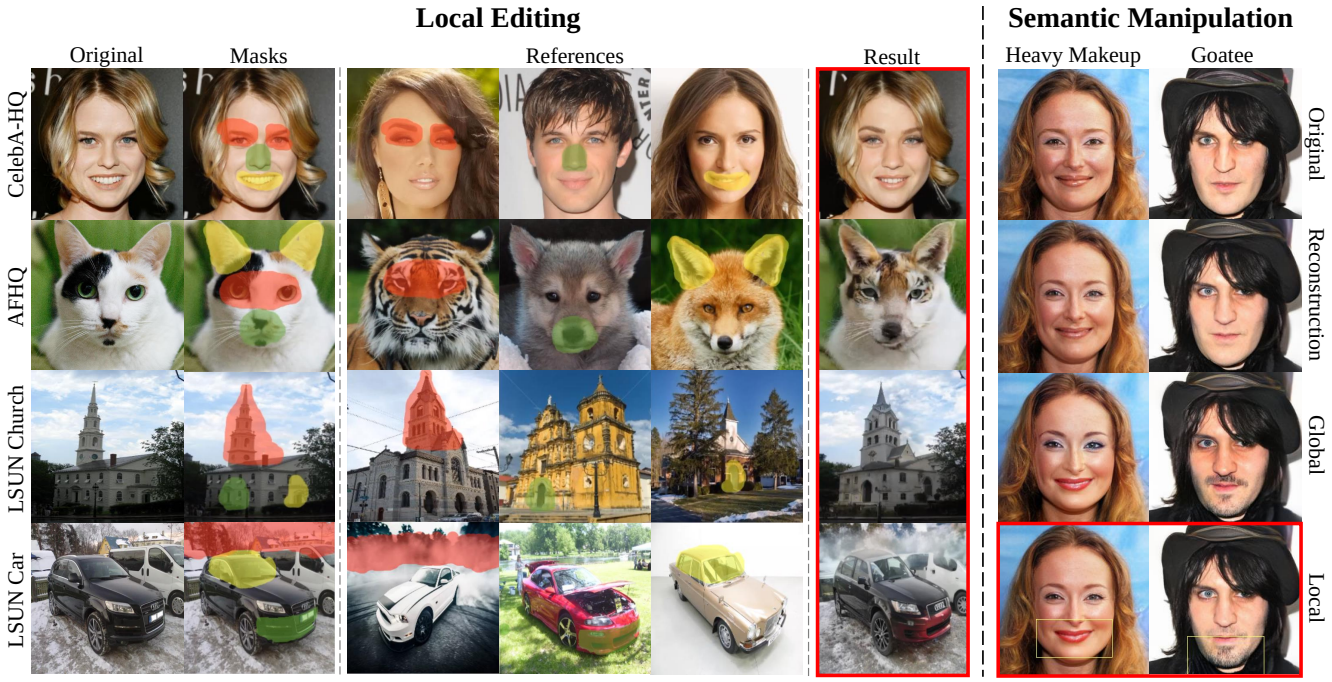


Figure 1: Various image editing results on multiple datasets. Local editing mixes multiple parts of reference images with the original image. Unlike other methods (*Global* case), ours can do semantic manipulation locally (Yellow box, *Local* case).

Abstract

Generative adversarial networks (GANs) synthesize realistic images from random latent vectors. Although manipulating the latent vectors controls the synthesized outputs, editing real images with GANs suffers from i) time-consuming optimization for projecting real images to the latent vectors, ii) or inaccurate embedding through an encoder. We propose StyleMapGAN: the intermediate latent space has spatial dimensions, and a spatially variant modulation replaces AdaIN. It makes the embedding through an encoder more accurate than existing optimization-based methods while maintaining the properties of GANs. Experimental results demonstrate that our method significantly outperforms state-of-the-art models in various image manipulation tasks such as local editing and image interpolation. Last but not least, conventional editing methods on GANs are still valid on our StyleMapGAN. Source code is available at https://github.com/naver-ai/StyleMapGAN.

1. Introduction

Generative adversarial networks (GANs) [16] have evolved dramatically in recent years, enabling high-fidelity image synthesis with models that are learned directly from data [6, 25, 26]. Recent studies have shown that GANs naturally learn to encode rich semantics within the latent space, thus changing the latent code leads to manipulating the corresponding attributes of the output images [22, 47, 17, 15, 48, 3, 57, 5]. However, it is still challenging to apply these manipulations to real images since the GAN lacks an inverse mapping from an image back to its corresponding latent code.

One promising approach for manipulating real images is

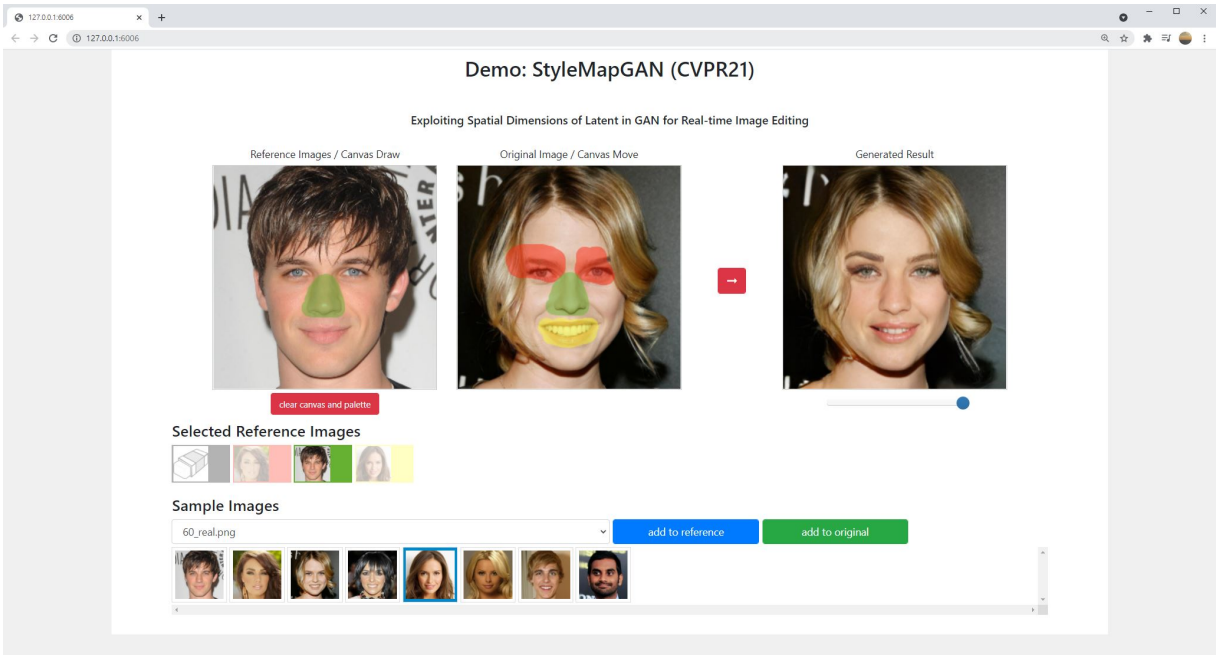


Figure 2: **StyleMapGAN Interactive Demo.** Please see our code for further details.

image-to-image translation [21, 64, 9, 27, 29], where the model learns to synthesize an output image given a user's input directly. However, these methods require pre-defined tasks and heavy supervision (*e.g.*, input-output pairs, class labels) for training and limit the user controllability at inference time. Another approach is to utilize pretrained GAN models by directly optimizing the latent code for an individual image [1, 2, 63, 37, 41]. However, even on high-end GPUs, it requires minutes of computation for each target image, and it does not guarantee that the optimized code would be placed in the original latent space of GAN.

A more practical approach is to train an extra encoder, which learns to project an image into its corresponding latent code [34, 62, 44, 36, 45]. Although this approach enables real-time projection in a single feed-forward manner, it suffers from the low fidelity of the projected image (*i.e.*, losing details of the target image). We attribute this limitation to the absence of spatial dimensions in the latent space. Without the spatial dimensions, an encoder compresses an image's local semantics into a vector in an entangled manner, making it difficult to reconstruct the image (*e.g.*, vector-based or low-resolution bottleneck layer is not capable of producing high-frequency details [33, 8]).

As a solution to such problems, we propose StyleMapGAN which exploits *stylemap*, a novel representation of the latent space. Our key idea is simple. Instead of learning a vector-based latent representation, we utilize a tensor with explicit spatial dimensions. Our proposed representation benefits from its spatial dimensions, enabling GANs to easily encode the local semantics of images into the latent space. This property allows an encoder to effectively project an image into the latent space, thus providing high-fidelity and real-time projection. Our method also offers a new capability to edit specific regions of an image by manipulating the matching positions of the stylemap. Figure 1 shows our local editing and local semantic manipulation results. Note that all editing is done in real-time. As shown in Figure 2, you can test our web demo to do interactive editing.

On multiple datasets, our stylemap indeed substantially enhances the projection quality compared to the traditional vector-based latent representation (§4.3). Furthermore, we show the advantage of our method over state-of-the-art methods on image projection, interpolation, and local editing (§4.4 & §4.5). Finally, we show that our method can transplant regions even when the regions are not aligned between one image and another (§4.6).

2. Related work

Optimization-based editing methods iteratively update the latent vector of pre-trained GANs to project a real image into the latent space [63, 7, 1, 62, 20, 4]. For example, Image2StyleGAN [1] reconstructs the image by optimizing intermediate representation for each layer of StyleGAN [25]. In-DomainGAN [62] focuses not only on reconstructing the image in pixel space, but also on landing the inverted code in the semantic domain of the original latent space. Neural Collage [53] and pix2latent [20] present a hybrid optimization strategy for projecting an image into the latent space of class-conditional GANs (*e.g.*, BigGAN [6]). On the other hand, we exploit an encoder, which makes editing two to

three orders of magnitude faster than optimization methods.

Learning-based editing methods train an extra encoder to directly infer the latent code given a target image [34, 13, 12, 14, 45]. For example, ALI [14] and BiGAN [12] introduce a fully adversarial framework to jointly learn the generator and the inverse mapping. Several work [34, 51, 55] has been made towards combining the variational autoencoder [32] with GANs for latent projection. ALAE [45] builds an encoder to predict the intermediate latent space of StyleGAN. However, all the above methods provide limited reconstruction quality due to the lack of spatial dimensions of latent space. Swap Autoencoder [43] learns to encode an image into two components, structure code and texture code, and generate a realistic image given any swapped combination. Although it can reconstruct images fast and precisely thanks to such representation, texture code is still a vector, which makes structured texture transfer challenging. Our editing method successfully reflects not only the color and texture but also the shape of a reference image.

Local editing methods tackle editing specific parts [11, 3, 65, 60, 49] (*e.g.*, nose, background) as opposed to the most GAN-based image editing methods modifying global appearance [47, 57, 43]. For example, Editing in Style [11] tries to identify each channel's contribution of the per-layer style vectors to specific parts. Structured Noise [3] replaces the learned constant from StyleGAN with an input tensor, which is a combination of local and global codes. However, these methods [11, 3, 5] do not target real image, which performances are degraded significantly in the real image. SEAN [65] facilitates editing real images by encoding images into the per-region style codes and manipulating them, but it requires pairs of images and segmentation masks for training. Besides, the style code is still a vector, so it has the same problem as Swap Autoencoder [43].

3. StyleMapGAN

Our goal is to project images to a latent space accurately with an encoder in real-time and locally manipulate images on the latent space. We propose StyleMapGAN which adopts *stylemap*, an intermediate latent space with spatial dimensions, and a spatially variant modulation based on the stylemap (§3.1). Note that the *style* denotes not only textures (fine style) but also shapes (coarse style) following [25]. Now an encoder can embed an image to the stylemap which reconstructs the image more accurately than optimization-based methods, and partial change in the stylemap leads to local editing on the image (§3.3).

3.1. Stylemap-based generator

Figure 3 describes the proposed stylemap-based generator. While a traditional mapping network produces style vectors to control feature maps, we create a stylemap with

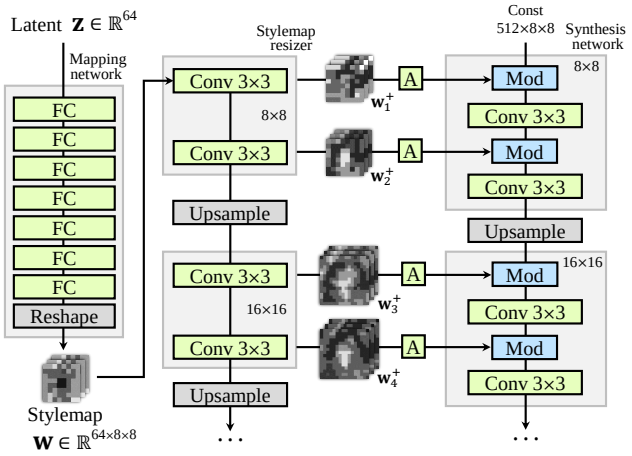


Figure 3: **StyleMapGAN Generator.** The stylemap $\mathbf{w}$ is resized to $\mathbf{w}^+$ through convolutional layers to match the spatial resolution of each feature in the synthesis network. Here "A" stands for a learned affine transform, which produces spatial modulation parameters (γ and β in Equation 1). "Mod" indicates modulation consisting of element-wise multiplication and addition. Note that the synthesis network starts from a learned constant input, and the output image's *style* is adjusted by resized stylemaps.

spatial dimensions, which not only makes the projection of a real image much more effective at inference but also enables local editing. The mapping network has a reshape layer at the end to produce the stylemap which forms the input to the spatially varying affine parameters. Since the feature maps in the synthesis network grow larger as getting closer to the output image, we introduce a stylemap resizer, which consists of convolutions and upsampling, to match the resolutions of stylemaps with the feature maps. The stylemap resizer resizes and transforms the stylemap with learned convolutions to convey more detailed and structured styles.

Then, the affine transform produces parameters for the modulation regarding the resized stylemaps. The modulation operation of the i-th layer in the synthesis network is as follows:

$$h_{i+1} = \left(\gamma_i \otimes \frac{h_i - \mu_i}{\sigma_i}\right) \oplus \beta_i \tag{1}$$

where $\mu_i, \sigma_i \in \mathbb{R}$ are the mean and standard deviation of activations $h_i \in \mathbb{R}^{C_i \times H_i \times W_i}$ of the i-th layer, respectively. $\gamma_i, \beta_i \in \mathbb{R}^{C_i \times H_i \times W_i}$ are modulation parameters. $\otimes$ and $\oplus$ are element-wise multiplication and addition, respectively.

We remove per-pixel noise which was an extra source of spatially varying inputs in StyleGAN, because our stylemap already provides spatially varying inputs and the single input makes the projection and editing simpler. Please see the supplementary material (§C) for other details about the net-

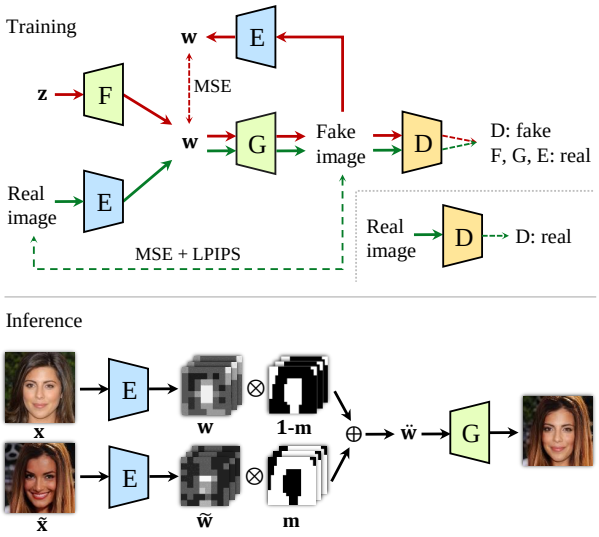


Figure 4: The upper figure contains an overall training scheme. Green and red arrows refer to flows associated with the real image and the generated image from Gaussian distribution, respectively. Dashed lines indicate loss functions. The lower figure shows our local editing method on the stylemap.

work and relationship with the autoencoder approach [19].

3.2. Training procedure and losses

In Figure 4, we use F, G, E, and D to indicate the mapping network, synthesis network with stylemap resizer, encoder, and discriminator, respectively, for brevity. D is the same as StyleGAN2, and the architecture of E is similar to D except without minibatch discrimination [46]. All networks are jointly trained using multiple losses as shown in Table 1. G and E are trained to reconstruct real images in terms of both pixel-level and perceptual-level [61]. Not only the image but E tries to reconstruct the stylemap with mean squared error (MSE) when G(F) synthesizes an image from z. D attempts to classify the real images and the fake images generated from Gaussian distribution. Lastly, we exploit domain-guided loss for the in-domain property [62]. E tries to reconstruct more realistic images by competing with D, making projected stylemap more suitable for image editing. If we remove any of the loss functions, generation and editing performance are degraded. Refer to the supplementary material for the effect of each loss function (§D) and joint learning (§B). Further training details (§C) are also involved.

3.3. Local editing

As shown at the bottom of Figure 4, the goal of local editing is to transplant some parts of a reference image to an original image with respect to a mask, which indicates the region to be modified. Note that the mask can be in any shape allowing interactive editing or label-based editing with semantic segmentation methods.

We project the original image and the reference image through the encoder to obtain stylemaps $\mathbf{w}$ and $\widetilde{\mathbf{w}}$, respectively. The edited stylemap $\ddot{\mathbf{w}}$ is an alpha blending of $\mathbf{w}$ and $\widetilde{\mathbf{w}}$:

$$\ddot{\mathbf{w}} = \mathbf{m} \otimes \widetilde{\mathbf{w}} \oplus (1 - \mathbf{m}) \otimes \mathbf{w} \tag{2}$$

where the mask $\mathbf{m}$ is shrunk by max pooling, and $\otimes$ and $\oplus$ are the same as Equation 1. In general, the mask is finer than 8×8, so we blend the stylemaps on the $\mathbf{w}^+$ space to achieve detailed manipulation. But for simplicity, we explain blending on the $\mathbf{w}$ space; the $\mathbf{w}^+$ space blending method is in the supplementary material (§A). Unless otherwise stated, local editing figures are blends on the $\mathbf{w}^+$ space.

Contrarily to SPADE [42] or SEAN [65], even rough masks as coarse as 8×8 produces plausible images so that the burden for user to provide detailed masks is lifted. This operation can be further revised for unidentical masks of the two images (§4.6).

Loss	G	D	E
Adversarial loss [16]	✓	✓	
R_1 regularization [40]		✓	
Latent reconstruction			✓
Image reconstruction	✓		✓
Perceptual loss [61]	✓		✓
Domain-guided loss [62]	✓	✓	✓

Table 1: Losses for training each network. Non-saturating loss [16] is used as the adversarial loss. R1-regularization is applied every 16 steps [26] for D to stabilize training. Latent reconstruction loss is mean squared error (MSE) in the $\mathbf{w}$ space. Image reconstruction is MSE in image pixel-level space. We use learned perceptual image patch similarity (LPIPS) as perceptual loss for calculating perceptual differences between original and reconstructed images. Domain-guided loss is related to adversarial training that reconstructed images from the encoder tries to be classified as real by the discriminator.

4. Experiments

Our proposed method efficiently projects images into the style space in real-time and effectively manipulates specific regions of real images. We first describe our experimental setup (§4.1) and evaluation metrics (§4.2) and show how the proposed spatial dimensions of stylemap affect the image projection and generation quality (§4.3). We then compare our method with the state-of-the-art methods on real image projection (§4.4) and local editing (§4.5). We finally show a

more flexible editing scenario and the usefulness of our proposed method (§4.6). Please see the supplementary material for high-resolution experiments (§B) and additional results (§E) such as random generation, style mixing, semantic manipulation, and failure cases.

4.1. Experimental setup

Baselines. We compare our model with recent generative models, including StyleGAN2 [26], Image2StyleGAN [1], In-DomainGAN [62], Structured Noise [3], Editing in Style [11], and SEAN [65]. We train all the baselines from scratch until they converge using the official implementations provided by the authors. For optimization-based methods [26, 1, 62, 3, 11], we use all the hyperparameters specified in their papers. We also compare our method with ALAE [45] qualitatively in the supplementary material (§E.2). Note that we do not compare our method against Image2StyleGAN++ [2] and Swap Autoencoder [43], since the authors have not published their code yet.

Datasets. We evaluate our model on CelebA-HQ [24], AFHQ [10], and LSUN Car & Church [59]. We adopt CelebA-HQ instead of FFHQ [25], since CelebA-HQ includes segmentation masks so that we can train the SEAN baseline and exploit the masks to evaluate local editing accurately in a semantic level. The AFHQ dataset includes wider variation than the human face dataset, which is suitable for showing the generality of our model. The optimization methods take an extremely long time, we limited the test and validation set to 500 images the same as In-DomainGAN [62]. The numbers of training images for CelebA-HQ, AFHQ, and LSUN Car & Church are 29K, 15K, 5.5M, and 126K, respectively. We trained all models at 256×256 resolution for comparison in a reasonable time, but we also provide 1024×1024 FFHQ results in the supplementary material (§B).

4.2. Evaluation metrics

Fréchet inception distance (FID). To evaluate the performance of image generation, we calculate FID [18] between images generated from Gaussian distribution and training set. We set the number of generated samples equal to that of training samples. We use the ImageNet-pretrained Inception-V3 [54] for feature extraction.

FID_{lerp}. To evaluate the global manipulation performance, we calculate FID between interpolated samples and training samples (FID_{lerp}). To generate interpolated samples, we first project 500 test images into the latent space and randomly choose pairs of latent vectors. We then generate an image using a linearly interpolated latent vector whose interpolation coefficient is randomly chosen between 0 and 1. We set the number of interpolated samples equal to that of training samples. Low FID_{lerp} indicates that the model provides high-fidelity and diverse interpolated samples.

MSE & LPIPS. To evaluate the projection quality, we estimate pixel-level and perceptual-level differences between target images and reconstructed images, which are mean square error (MSE) and learned perceptual image patch similarity (LPIPS) [61], respectively.

Average precision (AP). To evaluate the quality of locally edited images, we measure the average precision with the binary classifier trained on real and fake images [58], following the convention of the previous work [43]. We use the Blur+JPEG(0.5) model and full images for evaluation. The lower AP indicates that manipulated images are more indistinguishable from real images.

MSE_{src} & MSE_{ref}. In order to mix specific semantic, we make merged masks by combining target semantic masks of original and reference images. MSE_{src} and MSE_{ref} measure mean square error from the original image outside the mask and from the reference image inside the mask, respectively. To naturally combine them, images are paired by target semantic mask similarity. For local editing comparison on CelebA-HQ, 250 sets of test images are paired in each semantic (*e.g.*, background, hair) [35], which produces a total of 2500 images. For local editing on AFHQ, 250 sets of test images are paired randomly, and masks are chosen between the horizontal and vertical half-and-half mask, which produces 250 images.

4.3. Effects of stylemap resolution

To manipulate an image using a generative model, we first need to project the image into its latent space accurately. In Table 2, we vary the spatial resolution of stylemap and compare the performance of reconstruction and generation. For a fair comparison, we train our encoder model after training the StyleGAN2 generator. As the spatial resolution increases, the reconstruction accuracy improves significantly. It demonstrates that our stylemap with spatial dimensions is highly effective for image projection. FID varies differently across datasets, possibly due to different contextual relationships between locations for a generation. Note that our method with spatial resolution accurately preserves small details, *e.g.*, the eyes are not blurred.

Next, we evaluate the effect of the stylemap's resolution in editing scenarios, mixing specific parts of one image and another. Figure 5 shows that the 8×8 stylemap synthesizes the most plausible images in terms of seamlessness and preserving the identities of an original and reference image. We see that when the spatial resolution is higher than 8×8, the edited parts are easily detected.

Furthermore, we estimate FID_{lerp} in different resolution models in CelebA-HQ. The 8×8 model shows the best FID_{lerp} value (9.97) than other resolution models; 10.72, 11.05, and 12.10 for 4×4, 16×16, and 32×32, respectively. We suppose that the larger resolution of stylemap, the more likely projected latent from the encoder gets out of the latent

Method	Style resolution	Runtime (s)	CelebA-HQ MSE	CelebA-HQ LPIPS	CelebA-HQ FID	AFHQ MSE	AFHQ LPIPS	AFHQ FID
StyleGAN2	1×1	0.030	0.089	0.428	4.97	0.139	0.539	8.59
StyleMapGAN	4×4	0.085	0.062	0.351	**4.03**	0.070	0.394	14.82
StyleMapGAN	8×8	0.082	0.023	0.237	4.72	0.037	0.304	11.10
StyleMapGAN	16×16	0.078	0.010	0.146	4.71	0.016	0.183	**6.71**
StyleMapGAN	32×32	0.074	**0.004**	**0.076**	7.18	**0.006**	**0.090**	7.87

Table 2: Comparison of reconstruction and generation quality across different resolutions of the stylemap. The higher resolution helps accurate reconstruction, validating the effectiveness of stylemap. We observe that 8×8 stylemap already provides accurate enough reconstruction and accuracy gain, and afterward, improvements get visually negligible. Although FID varies differently across datasets, possibly due to the different contextual relationships between locations for generation, the stylemap does not seriously harm the images' quality; rather, it is even better in some configurations. Using our encoder and generator, total inference time is less than 0.1s with almost perfectly reconstructed images. Although StyleGAN2 with our encoder is faster than StyleMapGAN, but it suffers from poorly reconstructed images (second column).

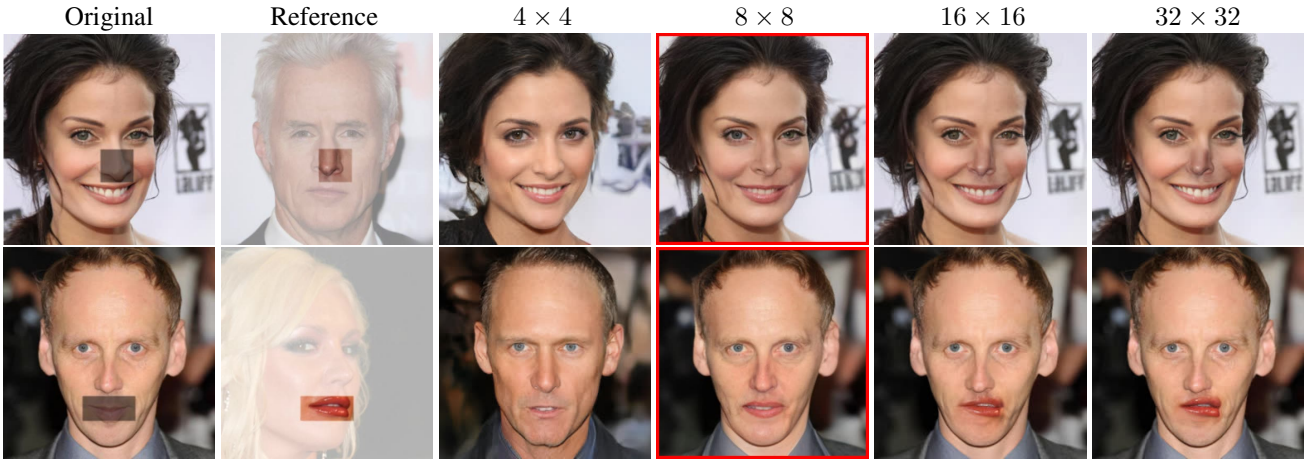
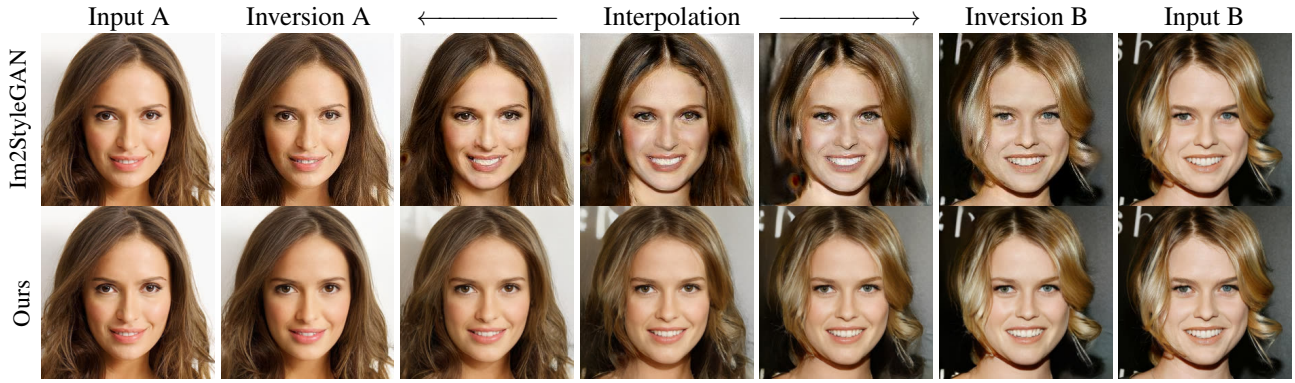


Figure 5: Local editing comparison across different resolutions of the stylemap. Regions to be discarded are faded on the original and the reference images. 4×4 suffers from the poor reconstruction. Resolutions greater than or equal to 16×16 result in too heterogeneous images. 8×8 resolution shows acceptable reconstruction and natural integration. Note that our method works well even in the case that the mask locates improperly as shown in the reference image of the first row.



Method	Runtime (s)	CelebA-HQ			AFHQ		
		MSE	LPIPS	FID_{lerp}	MSE	LPIPS	FID_{lerp}
StyleGAN2 [26]	80.4	0.079	0.247	30.30	0.091	0.288	**13.87**
Image2StyleGAN [1]	192.5	**0.009**	**0.203**	23.68	**0.018**	**0.282**	40.80
Structured Noise [3]	64.4	0.097	0.256	27.96	0.144	0.332	34.99
In-DomainGAN [62]	6.8	0.052	0.340	**22.05**	0.077	0.414	17.54
SEAN [65]	0.146	0.064	0.334	30.29	N/A	N/A	N/A
StyleMapGAN (Ours, 8×8)	**0.082**	**0.024**	**0.242**	**9.97**	**0.037**	**0.304**	**12.42**

Table 3: Comparison with the baselines for real image projection. Runtime covers the end-to-end interval of projection and generation in seconds. FID_{lerp} measures the quality of the images interpolated on the style space as a proxy for the potential quality of the manipulated images. Our method allows real-time manipulation of real images while achieving the best reconstruction accuracy and the best quality of the interpolated images. Although Image2StyleGAN produces the smallest reconstruction error, it suffers from minutes of runtime and poor interpolation quality, which are not suitable for practical editing. Its flaws can be found in the figure: rugged details in overall images, especially in teeth. SEAN is not applicable to AFHQ because it requires segmentation masks for training which are not available. The horizontal line between methods separates optimization-based methods and encoder-based methods.

space, which comes from a standard Gaussian distribution. Considering the editing quality and FID_{lerp}, we choose the 8×8 resolution as our best model and use it consistently for all subsequent experiments.

4.4. Real image projection

In Table 3, we compare our approach with the state-of-the-art methods for real image projection. For both datasets, StyleMapGAN achieves better reconstruction quality (MSE & LPIPS) than all competitors except Image2StyleGAN. However, Image2StyleGAN fails to meet requirements for editing in that it produces spurious artifacts in latent interpolation (FID_{lerp} and figures) and suffers from minutes of runtime. Our method also achieves the best FID_{lerp}, which implicitly shows that our manipulation on the style space leads to the most realistic images. Importantly, our method runs at least $100\times$ faster than the optimization-based baselines since a single feed-forward pass provides accurate projection thanks to the stylemap, which is measured in a single V100 GPU. SEAN also runs with a single feed-forward pass, but it requires ground-truth segmentation masks for both training and testing, which is a severe drawback for practical uses.

4.5. Local editing

We evaluate local editing performance regarding three aspects: detectability, faithfulness to the reference image in the mask, and preservation of the original image outside the mask. Figures 6 and 7 visually demonstrate that our method seamlessly composes the two images while others struggle. Since there is no metric for evaluating the last two aspects, we propose two quantitative metrics: MSE_{src} and MSE_{ref}. Table 4 shows that the results from our method are the hardest for the classifier to detect fakes, and both original and reference images are best reflected. Note that MSEs are not the sole measures, but AP should be considered together for the realness of the image.

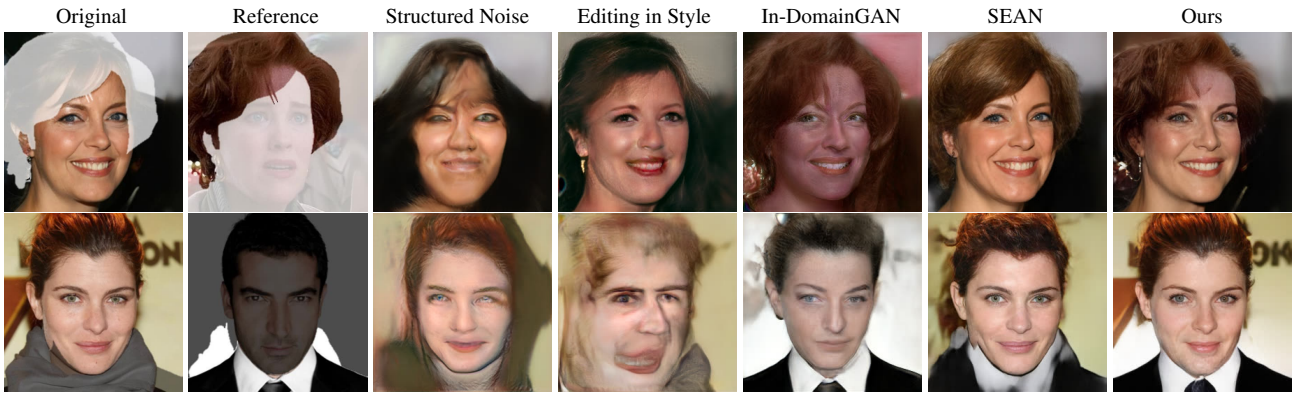


Figure 6: Local editing comparison on CelebA-HQ. The first two baselines [3, 11] even fail to preserve the untouched region. In-DomainGAN loses a lot of the original image's identity and poorly blends the two images, leaking colors to faces, hair, or background, respectively. SEAN locally transfers coarse structure and color but significantly loses details. Ours seamlessly transplants the target region from the reference to the original.

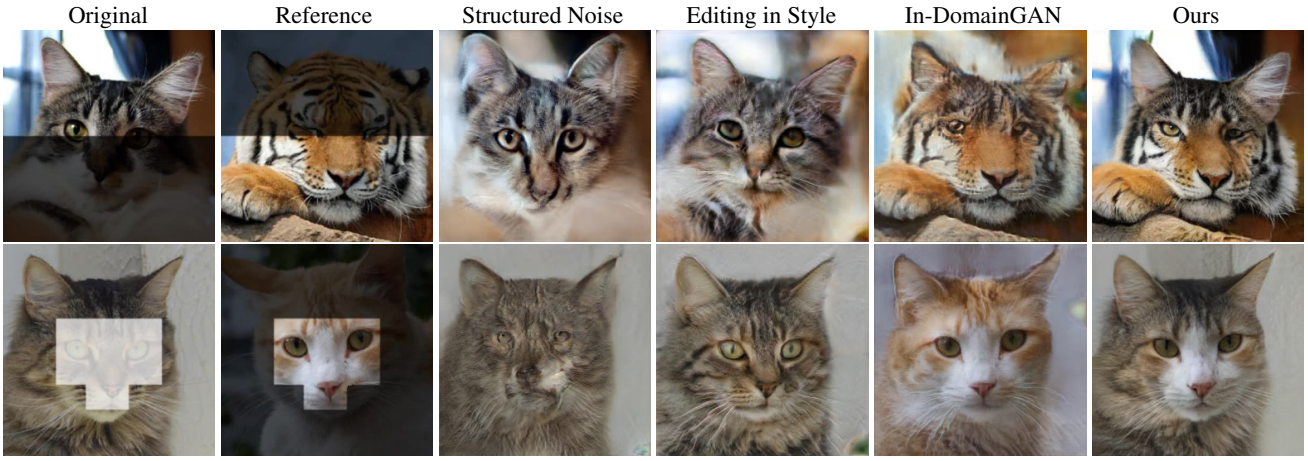


Figure 7: Local editing comparison on AFHQ. Each row blends the two images with horizontal and custom masks, respectively. Our method seamlessly composes two species with well-preserved details resulting in novel creatures, while others tend to lean towards one species.

Method	**Runtime (s)**	**CelebA-HQ**			**AFHQ**		
		AP	MSE_{src}	MSE_{ref}	AP	MSE_{src}	MSE_{ref}
Structured Noise [3]	64.4	99.16	0.105	0.395	99.88	0.137	0.444
Editing in Style [11]	55.6	98.34	0.094	0.321	99.52	0.130	0.417
In-DomainGAN [62]	6.8	98.72	0.164	**0.015**	99.59	0.172	**0.028**
SEAN [65]	0.155	90.41	0.067	0.141	N/A	N/A	N/A
StyleMapGAN (Ours, 8×8)	**0.099**	**83.60**	**0.039**	**0.105**	**98.66**	**0.050**	**0.050**

Table 4: Comparison with the baselines for local image editing. Average precision (AP) is measured with the binary classifier trained on real and fake images [58]. Low AP shows our edited images are more indistinguishable from real images than other baselines. Low MSE_{src} and MSE_{ref} imply that our model preserves the identity of the original image and brings the characteristics of the reference image well, respectively. Our method outperforms in all metrics except MSE_{ref} in In-DomainGAN. In-DomainGAN uses masked optimization, which only optimizes the target mask so that the identity of the original image has a great loss as shown in Figures 6 and 7.

Figure 8: Examples of unaligned transplantation. StyleMapGAN allows composing arbitrary number of any regions. The size and pose of the tower, bumper and wheels are automatically adjusted regarding the surroundings. The masks are specified on 8×8 grid and the stylemaps are blended on $\mathbf{w}$ space. The first row shows an example of copying one area of the reference image into multiple areas of the original images. The second row shows another example of copying two areas of the reference image. Our method can transplant the arbitrary number and size of areas of reference images.

4.6. Unaligned transplantation

Here, we demonstrate a more flexible use case, unaligned transplantation (image blending), showing that our local editing does not require the masks on the original and the reference images to be aligned. We project the images to the stylemaps and replace the designated region of the original stylemap with the crop of the reference stylemap even though they are in different locations. Users can specify what to replace. Figure 8 shows examples of LSUN Car & Church.

5. Discussion and Conclusion

Invertibility of GANs has been essential for editing real images with unconditional GAN models at a practical time, and it has not been properly answered yet. To achieve this goal, we propose StyleMapGAN, which introduces explicit spatial dimensions to the latent space, called a stylemap. We show that our method based on the stylemap has a number of advantages over prior approaches through an extensive evaluation. It can accurately project real images in real-time into the latent space and synthesize high-quality output images by both interpolation and local editing. We believe that improving fidelity by applying our latent representation to other methods such as conditional GANs (*e.g.*, BigGAN [6]) or variational autoencoders [32] would be exciting future work.

Acknowledgements. The authors thank NAVER AI Lab researchers for constructive discussion. All experiments were conducted on NAVER Smart Machine Learning (NSML) platform [28, 52].

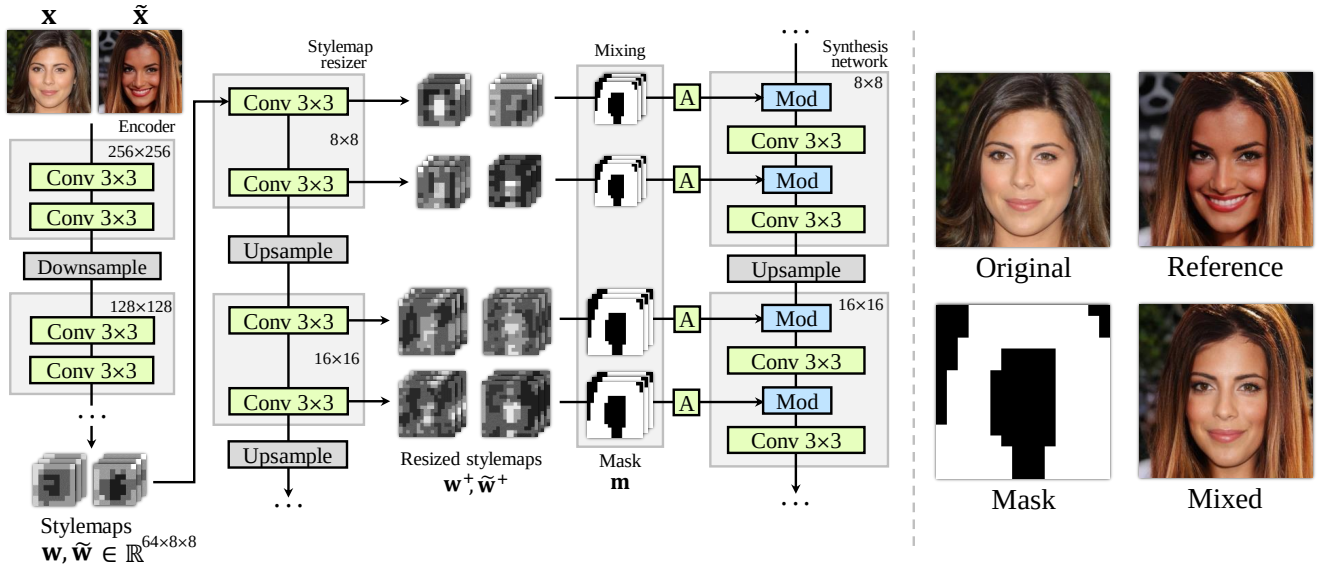


Figure 9: Our local editing starts with a learned encoder for fast image-to-stylemap projection. We estimate the stylemaps $\mathbf{w}$ and $\widetilde{\mathbf{w}}$ of the original $\mathbf{x}$ and the reference $\widetilde{\mathbf{x}}$ and transform them into multiple resolutions through the learned stylemap resizer. For each resolution, we calculate the alpha blending of the two stylemaps using the user-defined binary mask $\mathbf{m}$. Finally, the learned generator produces the output using the spatially-mixed stylemaps. The right one shows an example generated using our method.

A. Local editing in $\mathbf{w}^+$ space

This section illustrates how we perform local editing using StyleMapGAN. Although we already described the local editing method in Section 3.3 of the paper, it is impossible to edit in detail due to the coarse mask resolution (8×8). Contrary to the previous method, we propose a local editing method in $\mathbf{w}^+$ space. Regardless of the resolution of stylemap ($\mathbf{w}$), we can exploit detailed masks with resized stylemaps ($\mathbf{w}^+$) in high resolutions.

Figure 9 shows the overview of blending on the $\mathbf{w}^+$ space. The edited i-th resized stylemap $\ddot{\mathbf{w}}^+$ is an alpha blending of $\mathbf{w}^+$ and $\widetilde{\mathbf{w}}^+$:

$$\ddot{\mathbf{w}}_i^+ = \mathbf{m}_i \otimes \widetilde{\mathbf{w}}_i^+ \oplus (1 - \mathbf{m}_i) \otimes \mathbf{w}_i{}^+ \tag{3}$$

where i-th resized mask $\mathbf{m}_i$ is shrunk by max pooling. Although the mask's shape does not align with the 8×8 stylemap, we can precisely blend the two images on the $\mathbf{w}^+$ space.

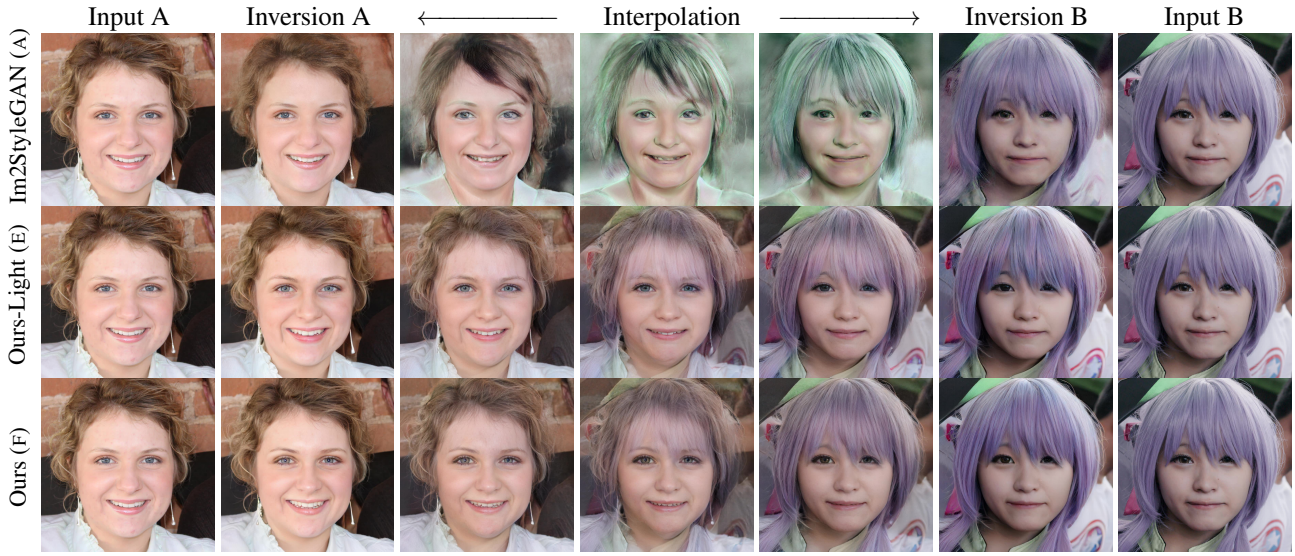
B. Experiments in the high-resolution dataset

We evaluate our model on FFHQ at 1024×1024 resolution. Baseline is StyleGAN2, and we also test Image2StyleGAN (A). StyleGAN2 official pretrained network is used for a fair comparison. StyleMapGAN adopts 32×32 stylemap for the high-resolution dataset, compared to 8×8 stylemap for 256×256 image. StyleMapGAN-Light (E) is a light version of StyleMapGAN; it reduces the number of parameters of the generator. Another training setting (D) is sequential learning, which trains the generator first and then trains the encoder. In Table 5, we used the same protocol as the paper to calculate MSE, LPIPS, and FID_{lerp}. The number of training images for FFHQ is 69K, and we limited the test and validation set to 500 images.

Comparison with baselines. As shown in Table 5, Image2StyleGAN reconstructs the image well, but it struggles with poor interpolation quality. Low FID_{lerp}, rugged interpolation results and lengthy runtime shows Image2StyleGAN is not suitable for image editing tasks. StyleMapGAN outperforms baselines in all metrics, and even StyleMapGAN-Light shows astonishing results.

StyleMapGAN-Light is $2.5\times$ smaller than the original version. Stylemap resizer accounts for a large portion of the network's size, so we reduce the number of channels of feature maps in the stylemap resizer. The reconstruction image lacks some detail, but StyleMapGAN-Light still outperforms baselines, and FID_{lerp} is even better than the original version. Please see our code to refer to the number of channels.

Joint learning is important when training StyleMapGAN. It makes training stable and network performance



Network	Projection	Joint learning	G param(M)	runtime(s)	G GPU(GB)	MSE	LPIPS	FID_{lerp}
A StyleGAN2	Image2StyleGAN	✗	30.4	454.7	2.1	0.021	0.468	38.00
B StyleGAN2	StyleGAN2	✗	30.4	142.2	2.1	0.093	0.467	34.65
D StyleMapGAN-Light	Encoder	✗	18.6	0.253	3.0	0.071	0.546	201.18
E StyleMapGAN-Light	Encoder	✓	18.6	0.253	3.0	0.017	0.347	**13.52**
F StyleMapGAN	Encoder	✓	46.4	0.249	3.1	**0.016**	**0.344**	13.68

Table 5: Comparison with StyleGAN2 on 1024×1024 FFHQ. We also explored the effect of other components such as generator size and joint learning. StyleMapGAN-Light has reduced the number of channels of the stylemap resizer. 32×32 stylemap is used for the large size of the image. "G" denotes the generator.

better. Training the encoder after training the generator fails to reconstruct images. We speculate the reason why joint learning is better than sequential learning as follows. In joint learning, the generator and the encoder affect each other. The generator generates an image that would be easy to reconstruct by the encoder. The structure of the encoder is a stack of convolutional layers, which makes the projected stylemap is prone to have local correspondence: Partial change in the stylemap leads to local editing on the image. Through joint learning, the mapping network in the generator also learns to make the stylemap from Gaussian distribution have the local correspondence.

C. Implementation details

Architecture. We follow StyleGAN2 [26] regarding the discriminator architecture and the feature map counts in the convolutional layers of the synthesis network. Our mapping network is an MLP with eight fully connected layers followed by a reshape layer. The channel sizes are 64, except the last being 4,096. Our encoder adopts the discriminator architecture until the 8×8 layer and without minibatch discrimination [46].

Training. We jointly train the generator, the encoder, and the discriminator. It is simpler and leads to more stable training and higher performance than separately training the adversarial networks and the encoder, as described in §B. For the rest, we mostly follow the settings of StyleGAN2, *e.g.*, the discriminator architecture, R1 regularization [40] in the discriminator using $\gamma = 10$, Adam [30] optimizer with 0.002 learning rate, $\beta_1 = 0.0$ and $\beta_2 = 0.99$, an exponential moving average of the generator and the encoder, leaky ReLU [38], equalized learning rate [24] for all layers, random horizontal flip for augmentation, and reducing the learning rate by two orders [25] of magnitude for the mapping network. Our code is based on unofficial PyTorch implementation of StyleGAN2[1]. All StyleMapGAN variants at 256×256 are trained for two weeks on 5M images with 2 Tesla V100 GPUs using a minibatch size of 16. In §B, 1024×1024 models are trained for one week on 2.5M images with 8 Tesla V100 GPUs using a minibatch size of 16. We note that most cases keep slowly improving until 10M images. Our code is publicly available online for reproducibility[2].

[1] https://github.com/rosinality/stylegan2-pytorch
[2] https://github.com/naver-ai/StyleMapGAN

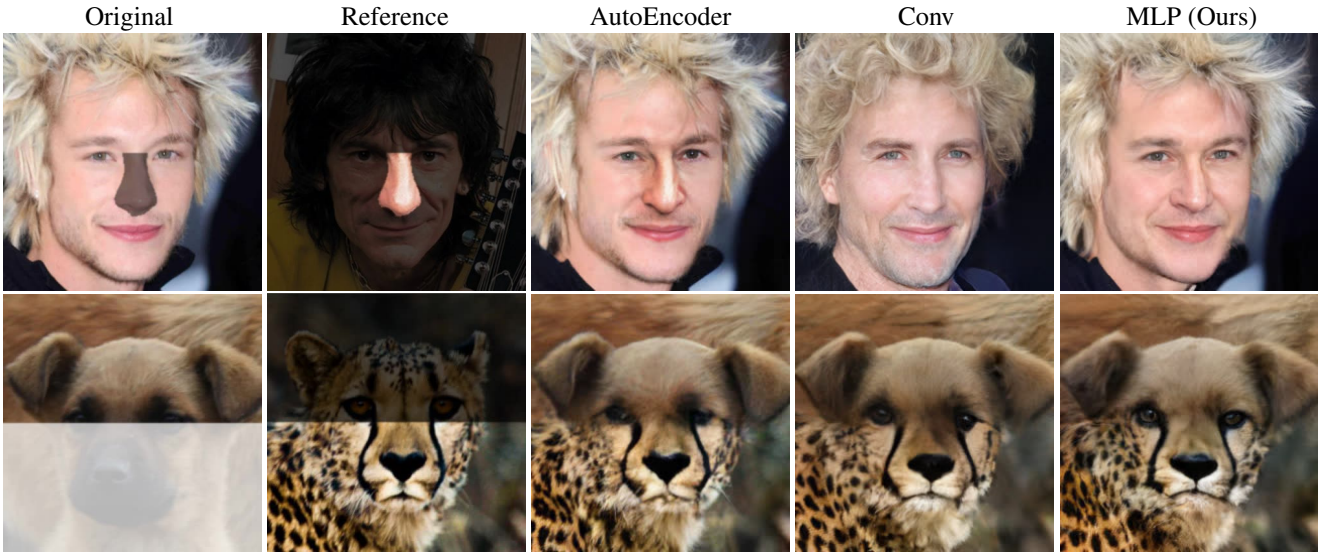


Figure 10: Local editing comparison across different mapping network architectures of StyleMapGAN. MLP-based architecture provides more natural images compared to autoencoder-based and convolution-based architecture.

Mapping network design for stylemap. There are several choices when designing a mapping network. We can remove the mapping network so that our method does not generate images from the standard Gaussian distribution and uses only real images for training like autoencoder [19]. As shown in Figure 10, autoencoder fails to produce realistic images using the projected stylemap. It seems to copy and paste between two images on RGB space. The autoencoder uses only images as input, which is a discrete variable. On the contrary, our method uses not only images but also the latent from Gaussian distribution, which is a continuous space. If we mix two latent codes for editing the image, training with continuous latent space can cover more latent values than discrete latent space.

Alternatively, we can easily think of convolutional layers due to the spatial dimensions of the stylemap. But, the mapping network with convolutional layers struggles in reconstruction so that the edited results images are quite different from the original images. We assume that there is such a limit because the convolutional layer's mapping is bounded to the local area. On the other hand, each weight and input in MLP are fully-connected so that it can make a more flexible latent space.

D. Loss details

In Section 3.2 of the main paper, we briefly introduced six losses. In this section, we provide details of the losses and their responsibilities. Some losses degrade reconstruction quality (MSE, LPIPS [61]), but we need every loss for the best editing quality (FID_{lerp}). We can obtain the best FID_{lerp} by training with all losses. Table 6 shows the quantitative results of the ablation study. The coefficients of all loss terms are set to 1.

Figure 11: This figure shows the interpolation results of different networks. Leftmost images are results of a network trained using whole losses. Second column images are generated by a network trained without random Gaussian noise, which is similar to AutoEncoder. A network trained without domain-guided loss generates rightmost column images.

Adversarial loss. The discriminator tries to classify fake images as fake, which are generated randomly from Gaussian distribution or reconstruction of input images. On the contrary, the generator fools the discriminator by producing more realistic images. Generation from the continuous

Removed loss	MSE	LPIPS	FID	FID_{lerp}
Adversarial loss	**0.009**	**0.137**	278.87	12.99
Domain-guided loss	0.013	0.193	5.11	16.84
Latent reconstruction loss	0.021	0.220	**4.43**	10.08
Image reconstruction loss	0.029	0.254	5.01	10.29
Perceptual loss	0.033	0.304	5.34	13.33
R1 regularization	0.097	0.403	31.82	14.56
Train with all losses	0.023	0.237	4.72	**9.97**

Table 6: Loss ablation study removing one loss at a time. We used CelebA-HQ, 256×256 image, and 8×8 stylemap.

space increases generation power in terms of smooth interpolation. Without adversarial loss related to the mapping network, we can not obtain a smooth manifold of latent space as mentioned in §C. Figure 11 also shows unnatural interpolation results and checkerboard artifacts if we don't use adversarial loss. We use the non-saturating loss [16] as our adversarial loss.

Domain-guided loss. Domain-guided loss is introduced by In-DomainGAN [62]. We use an adversarial training manner on fake images generated from real images via the encoder and the generator. The discriminator tries to classify generated images as fake while the encoder and the generator attempt to fool the discriminator. The loss pushes the projected latent code to remain in the original latent space of GAN, which facilitates smooth real-image editing by exploiting GAN's properties (*e.g.*, smooth interpolation). Without domain-guided loss, interpolation results are blurry as shown in Figure 11.

Latent reconstruction loss. The goal of the encoder is to find the latent code which generates the target image. When we generate a fake image from Gaussian distribution, we know the pair of the latent code and the generated image. Using that supervision, we train the encoder like other approaches [56, 51, 45, 62]. The encoder tries to project images in the semantic domain of the original latent space and alleviates strong bias against pixel-level reconstruction.

Image reconstruction loss. To make the output image visually identical to the input image, we minimize differences between them at pixel-level. If we do not use this loss function, visual reconstruction fails as ALAE [45] does.

Perceptual loss. Image reconstruction loss often makes the encoder overfit and output blurry images. Several approaches [1, 2, 62] adopt perceptual loss [23], which exploits the features extracted by VGG [50], for perceptual-level reconstruction. We use LPIPS [61] for perceptual loss, which has better feature representation.

R1 regularization. R1 regularization [40] makes training stable. We find that lazy regularization [26] is enough and apply it every 16 steps for the discriminator. Without this loss function, performance degrades in all metrics.

E. Additional results

In this section, we show extensive qualitative results. §E.1 illustrates randomly generated images to show that the generation capability of our method does not degenerate compared to the baseline. §E.2 and §E.3 provide expanded comparison on reconstruction and local editing, respectively. §E.4 shows additional unaligned transplantation examples. Our method is applicable to other latent-based editing methods as shown in §E.5 and §E.6. Lastly, we discuss the limitations (§E.7) of our method.

E.1. Random generation

The primary objective of GANs is generating high-fidelity images from random Gaussian noise. We show random generation results for each dataset: CelebA-HQ [24], AFHQ [10], and LSUN Car & Church [59]. We use 8×8 resolution of stylemap except for AFHQ, in which case 16×16 resolution provides much better generation quality as shown in Table 2 of the main paper. To generate high-quality images, we use the truncation trick [6, 31, 39] with $\psi = 0.7$. Figure 12 shows uncurated images and FID values. In CelebA-HQ and AFHQ, we use the same FID protocol as in the main experiments; the number of generated samples equal to that of training samples. On the other hand, LSUN consists of a lot of training images so that we use 50k images randomly chosen from the training set; the number of generated samples is 50k, too. Low FIDs reveal that our method has satisfactory generation capability.

E.2. Image projection & Interpolation

Although encoder-based methods project images into latent space in real time, their projection quality falls short of expectations. Figure 13 shows the projection quality comparison between our method and other encoder-based baselines (ALAE [45], In-DomainGAN [62], and SEAN [65]).

Figure 14 shows projection and interpolation results of our method and Image2StyleGAN [1]. Although Image2StyleGAN reconstructs the input images in high-fidelity, it struggles in latent interpolation because its projection on $\mathbf{w}^+$ drifts from the learned latent space $\mathbf{w}$ of the generator.

E.3. Local editing

Figure 15 shows local editing comparison with competitors. We eject two competitors (Structured Noise [3] and Editing in Style [11]) due to their poor results, as shown in

Figure 4 of the main paper. It is because they do not target editing real images but target fake images.

E.4. Unaligned transplantation

Figure 16 and 17 show unaligned transplantation results in LSUN Car & Church [59]. Our method can transplant the arbitrary number and location of areas in reference images to the original images. Note that our method adjusts the color tone and structure of the same reference regarding the original images.

E.5. Semantic manipulation

We exploit InterFaceGAN [47] to find the semantic boundary in the latent space. Our method can change the semantic attribute using a certain direction derived from the boundary. We apply the direction on stylemap ($\mathbf{w}$ space). Figure 18 shows two versions of semantic manipulation. The global version is a typical way to manipulate attributes. The local version is only available in our method due to the spatial dimensions of the stylemap. We apply the semantic direction on the specified location in the $\mathbf{w}$ space. It allows us not to change the undesired area of the original image regardless of attribute correlation. For example, "Rosy Cheeks" makes lips red and "Goatee" changes the color of noses in the global version but not in the local version as shown in Figure 18. Furthermore, we can change part of attributes such as lip makeup from "Heavy Makeup" and beard from "Goatee". It alleviates the hard labor of highly granular labeling. Swap Autoencoder [43] shows region editing that the structure code also can be manipulated locally. However, it can not apply region editing on some attributes (*e.g.*, "Pale Skin") which are related to color and textures due to the absence of spatial dimensions in the texture code.

E.6. Style mixing

StyleGAN [25] proposed the style mixing method, which copies a specified subset of styles from the reference image. We operate style mixing in resized stylemaps ($\mathbf{w}^+$). Unlike StyleGAN, our generator has color and texture information in the resized stylemaps of low resolution (8×8). On the other hand, it generates overall structure through other resolutions ($16^2 - 256^2$). If we want to bring the color and texture styles from the reference image, we replace 8×8 resized stylemaps by reference. Figure 19 shows the examples.

Using style mixing and unaligned transplantation, we can transfer local structure only as shown in Figure 20. We use the original image on the first resized stylemap and the reference image for the remaining resolutions in the target region.

E.7. Failure cases

Our method has a limitation when original and reference images have different poses and target semantic sizes. Figure 21 shows failure cases on different poses. Especially, hair is not interpolated smoothly. Figure 22 shows failure cases on different target semantic sizes. The sizes and poses of the bumper vary, and our method can not transplant it naturally. This limitation gets worse when the resolution of the stylemap increases. Resolving this problem would be interesting future work.

References

[1] Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2stylegan: How to embed images into the stylegan latent space? In *CVPR*, 2019. 2, 5, 7, 13

[2] Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2stylegan++: How to edit the embedded images? In *CVPR*, 2020. 2, 5, 13

[3] Yazeed Alharbi and Peter Wonka. Disentangled image generation through structured noise injection. In *CVPR*, 2020. 1, 3, 5, 7, 8, 13

[4] David Bau, Hendrik Strobelt, William Peebles, Jonas Wulff, Bolei Zhou, Jun-Yan Zhu, and Antonio Torralba. Semantic photo manipulation with a generative image prior. *ACM Transactions on Graphics*, 2019. 2

[5] David Bau, Jun-Yan Zhu, Hendrik Strobelt, Bolei Zhou, Joshua B Tenenbaum, William T Freeman, and Antonio Torralba. Gan dissection: Visualizing and understanding generative adversarial networks. *ICLR*, 2019. 1, 3

[6] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. In *ICLR*, 2019. 1, 2, 9, 13

[7] Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. Neural photo editing with introspective adversarial networks. *arXiv preprint arXiv:1609.07093*, 2016. 2

[8] Huiwen Chang, Jingwan Lu, Fisher Yu, and Adam Finkelstein. Pairedcyclegan: Asymmetric style transfer for applying and removing makeup. In *CVPR*, 2018. 2

[9] 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 *CVPR*, 2018. 2

[10] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. In *CVPR*, 2020. 5, 13

[11] Edo Collins, Raja Bala, Bob Price, and Sabine Susstrunk. Editing in style: Uncovering the local semantics of gans. In *CVPR*, 2020. 3, 5, 8, 13

[12] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, 2016. 3

[13] Jeff Donahue and Karen Simonyan. Large scale adversarial representation learning. In *NeurIPS*, 2019. 3

[14] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron

Courville. Adversarially learned inference. *arXiv preprint arXiv:1606.00704*, 2016. 3

[15] Lore Goetschalckx, Alex Andonian, Aude Oliva, and Phillip Isola. Ganalyze: Toward visual definitions of cognitive image properties. In *ICCV*, 2019. 1

[16] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. In *NeurIPS*, 2014. 1, 4, 13

[17] Erik Härkönen, Aaron Hertzmann, Jaakko Lehtinen, and Sylvain Paris. Ganspace: Discovering interpretable gan controls. *arXiv preprint*, 2020. 1

[18] 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. In *NeurIPS*, 2017. 5

[19] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 2006. 4, 12

[20] Minyoung Huh, Richard Zhang, Jun-Yan Zhu, Sylvain Paris, and Aaron Hertzmann. Transforming and projecting images into class-conditional generative networks. *arXiv preprint arXiv:2005.01703*, 2020. 2

[21] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial nets. In *CVPR*, 2017. 2

[22] Ali Jahanian, Lucy Chai, and Phillip Isola. On the "steerability" of generative adversarial networks. In *ICLR*, 2020. 1

[23] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016. 13

[24] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *ICLR*, 2018. 5, 11, 13

[25] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019. 1, 2, 3, 5, 11, 14

[26] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *CVPR*, 2020. 1, 4, 5, 7, 11, 13

[27] Hyunsu Kim, Ho Young Jhoo, Eunhyeok Park, and Sungjoo Yoo. Tag2pix: Line art colorization using text tag with secat and changing loss. In *ICCV*, 2019. 2

[28] Hanjoo Kim, Minkyu Kim, Dongjoo Seo, Jinwoong Kim, Heungseok Park, Soeun Park, Hyunwoo Jo, KyungHyun Kim, Youngil Yang, Youngkwan Kim, et al. Nsml: Meet the mlaas platform with a real-world case study. *arXiv preprint arXiv:1810.09957*, 2018. 9

[29] Junho Kim, Minjae Kim, Hyeonwoo Kang, and Kwang Hee Lee. U-gat-it: Unsupervised generative attentional networks with adaptive layer-instance normalization for image-to-image translation. In *ICLR*, 2020. 2

[30] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 11

[31] Diederik P. Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *NeurIPS*, 2018. 13

[32] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint*, 2013. 3, 9

[33] Guillaume Lample, Neil Zeghidour, Nicolas Usunier, Antoine Bordes, Ludovic Denoyer, and Marc'Aurelio Ranzato. Fader networks: Manipulating images by sliding attributes. In *NeurIPS*, 2017. 2

[34] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In *ICML*, 2016. 2, 3

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

[36] Junyu Luo, Yong Xu, Chenwei Tang, and Jiancheng Lv. Learning inverse mapping by autoencoder based generative adversarial nets. In *ICNIP*, 2017. 2

[37] Fangchang Ma, Ulas Ayaz, and Sertac Karaman. Invertibility of convolutional generative networks from partial measurements. In *NeurIPS*, 2018. 2

[38] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *ICML*, 2013. 11

[39] M. Marchesi. Megapixel size image creation using generative adversarial networks. *ArXiv*, abs/1706.00082, 2017. 13

[40] Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. Which training methods for gans do actually converge? In *ICML*, 2018. 4, 11, 13

[41] Atsuhiro Noguchi and Tatsuya Harada. Image generation from small datasets via batch statistics adaptation. In *ICCV*, 2019. 2

[42] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *CVPR*, 2019. 4

[43] Taesung Park, Jun-Yan Zhu, Oliver Wang, Jingwan Lu, Eli Shechtman, Alexei A. Efros, and Richard Zhang. Swapping autoencoder for deep image manipulation. In *NeurIPS*, 2020. 3, 5, 14

[44] Guim Perarnau, Joost Van De Weijer, Bogdan Raducanu, and Jose M Álvarez. Invertible conditional gans for image editing. *arXiv preprint*, 2016. 2

[45] Stanislav Pidhorskyi, Donald Adjeroh, and Gianfranco Doretto. Adversarial latent autoencoders. In *CVPR*, 2020. 2, 3, 5, 13, 18

[46] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *NeurIPS*, 2016. 4, 11

[47] Yujun Shen, Jinjin Gu, Xiaoou Tang, and Bolei Zhou. Interpreting the latent space of gans for semantic face editing. In *CVPR*, 2020. 1, 3, 14

[48] Yujun Shen and Bolei Zhou. Closed-form factorization of latent semantics in gans. *arXiv preprint*, 2020. 1

[49] Assaf Shocher, Yossi Gandelsman, Inbar Mosseri, Michal Yarom, Michal Irani, William T Freeman, and Tali Dekel. Semantic pyramid for image generation. In *CVPR*, 2020. 3

[50] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 13

[51] Akash Srivastava, Lazar Valkov, Chris Russell, Michael U. Gutmann, and Charles Sutton. Veegan: Reducing mode collapse in gans using implicit variational learning. In *NeurIPS*, 2017. 3, 13

[52] Nako Sung, Minkyu Kim, Hyunwoo Jo, Youngil Yang, Jingwoong Kim, Leonard Lausen, Youngkwan Kim, Gayoung Lee, Donghyun Kwak, Jung-Woo Ha, et al. Nsml: A machine learning platform that enables you to focus on your models. *arXiv preprint arXiv:1712.05902*, 2017. 9

[53] Ryohei Suzuki, Masanori Koyama, Takeru Miyato, Taizan Yonetsuji, and Huachun Zhu. Spatially controllable image synthesis with internal representation collaging. *arXiv preprint arXiv:1811.10153*, 2018. 2

[54] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016. 5

[55] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. It takes (only) two: Adversarial generator-encoder networks. *arXiv preprint arXiv:1704.02304*, 2017. 3

[56] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. It takes (only) two: Adversarial generator-encoder networks. In *AAAI*, 2017. 13

[57] Andrey Voynov and Artem Babenko. Unsupervised discovery of interpretable directions in the gan latent space. In *ICML*, 2020. 1, 3

[58] Sheng-Yu Wang, Oliver Wang, Richard Zhang, Andrew Owens, and Alexei A Efros. Cnn-generated images are surprisingly easy to spot... for now. In *CVPR*, 2020. 5, 8

[59] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015. 5, 13, 14

[60] Fangneng Zhan, Hongyuan Zhu, and Shijian Lu. Spatial fusion gan for image synthesis. In *CVPR*, 2019. 3

[61] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 4, 5, 12, 13

[62] Jiapeng Zhu, Yujun Shen, Deli Zhao, and Bolei Zhou. In-domain gan inversion for real image editing. In *ECCV*, 2020. 2, 4, 5, 7, 8, 13, 18, 20

[63] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros. Generative visual manipulation on the natural image manifold. In *ECCV*, 2016. 2

[64] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017. 2

[65] Peihao Zhu, Rameen Abdal, Yipeng Qin, and Peter Wonka. Sean: Image synthesis with semantic region-adaptive normalization. In *CVPR*, 2020. 3, 4, 5, 7, 8, 13, 18, 20

CelebA-HQ, FID: **4.92**

AFHQ, FID: **6.71**

LSUN Car, FID: **4.15**

LSUN Church, FID: **2.95**

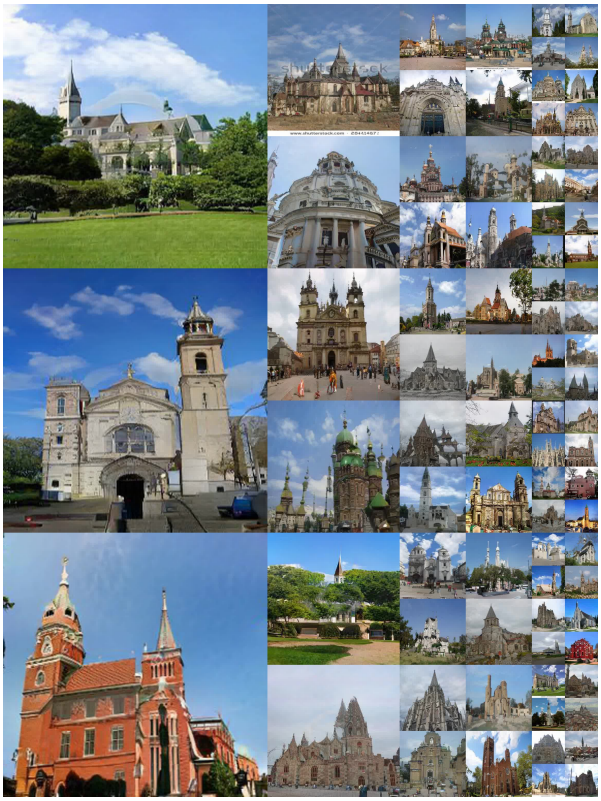


Figure 12: Uncurated random generation results in four datasets.

Figure 13: Reconstruction results in encoder-based methods. ALAE [45] does not preserve identities in the original images. In-DomainGAN [62] has better reconstruction results than ALAE, but it sometimes fails to generate human-like images as shown in the second-last row. Note that In-DomainGAN requires additional optimization steps which take seconds. SEAN [65] fails to preserve the shape of original images (*e.g.*, background, hair curl, and cloth). Our method reconstructs images well not only the color and texture but also the shape.

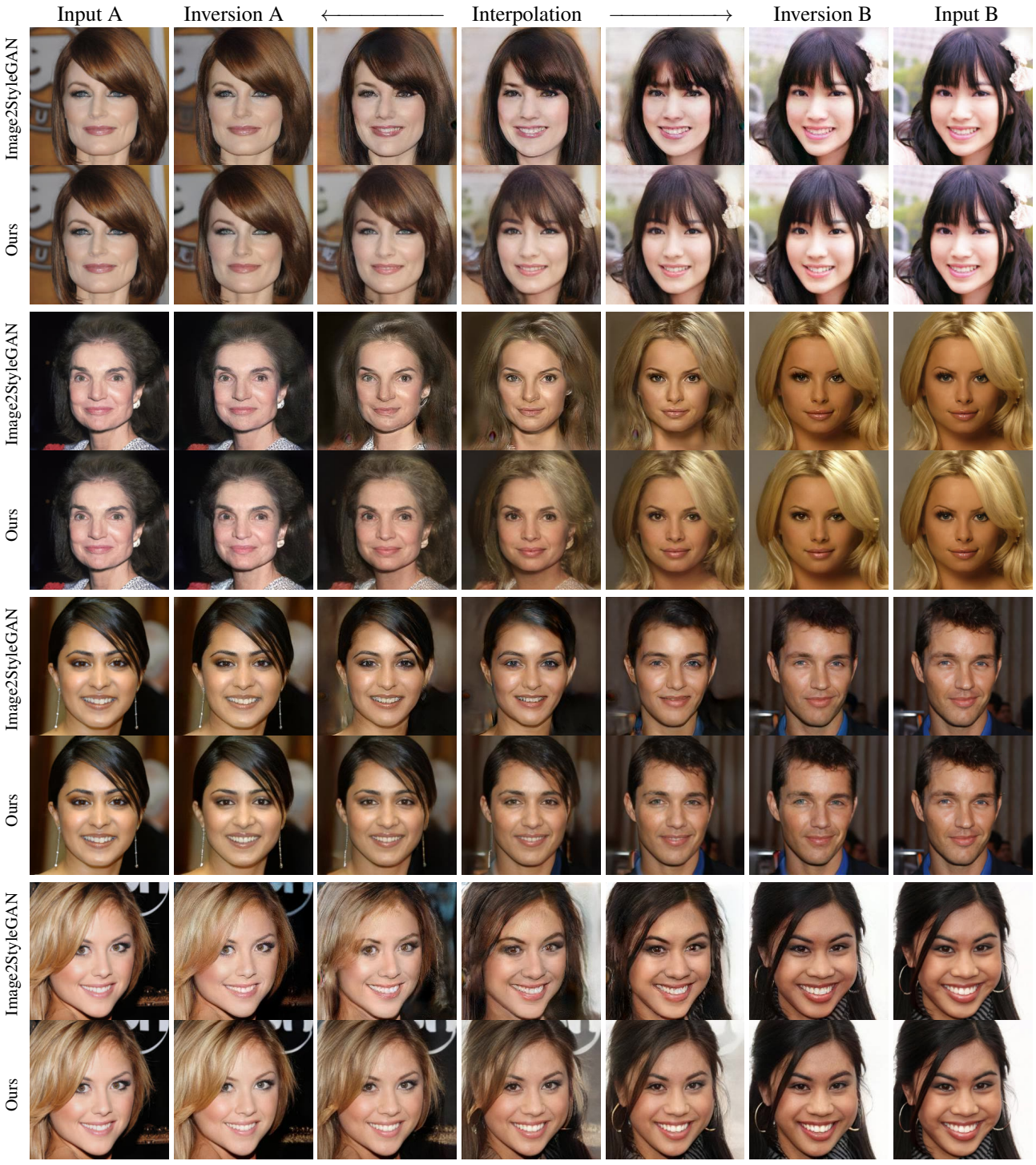


Figure 14: Comparison with Image2StyleGAN for interpolation quality. Image2StyleGAN shows rugged and discontinuous interpolations, even though the reconstruction quality is visually good. Our method produces clearer and smoother interpolations, which reveal our superiority in both pixel-level and semantic-level (*i.e.*, the semantics the original latent space) reconstruction. Note that the reconstruction speed of our method is $2000\times$ faster than Image2StyleGAN.

Figure 15: Local editing comparison in CelebA-HQ. Each row edits hair, eyes, nose, cloth, and background. In-DomainGAN [62] only optimizes region in the target mask and it changes identities of the original images. SEAN [65] tends to bring only the color and the texture of reference images, not the shape (especially on hair lines). Our method reflects the shape of the reference image as well and preserves the identity of the original image.

Figure 16: Transplantation results of our method in LSUN Car. We transplant the cabin, wheel, and bumper.

Figure 17: Transplanting tower, gate and windows in LSUN Church by our method. Our method can transplant the arbitrary number and location of areas in reference images to the original images. Note that our method adjusts color tone and structure of the same references regarding the original images.

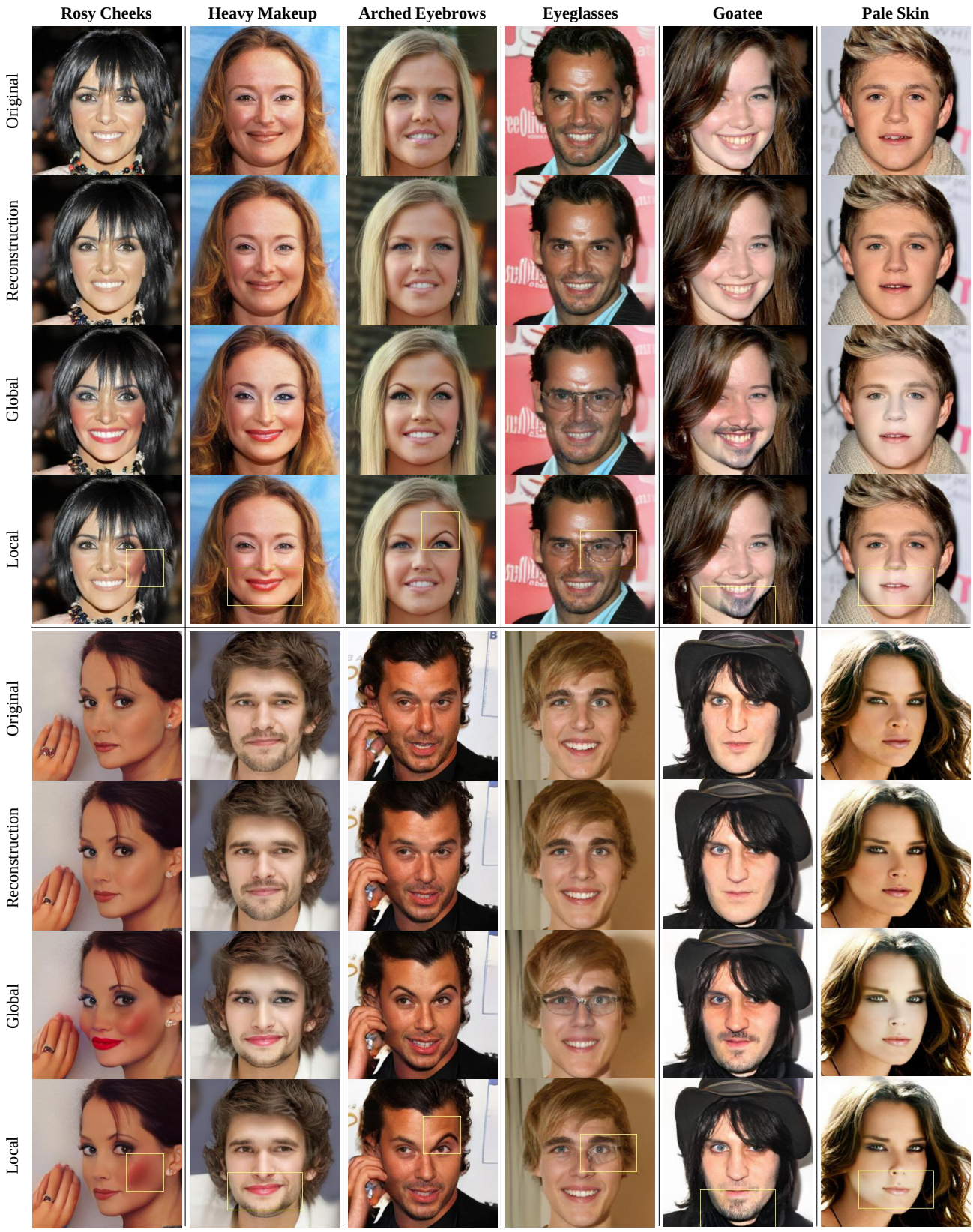


Figure 18: The results of the global and local version of semantic manipulation in our method. In local manipulation, we apply the semantic direction in the yellow box area.

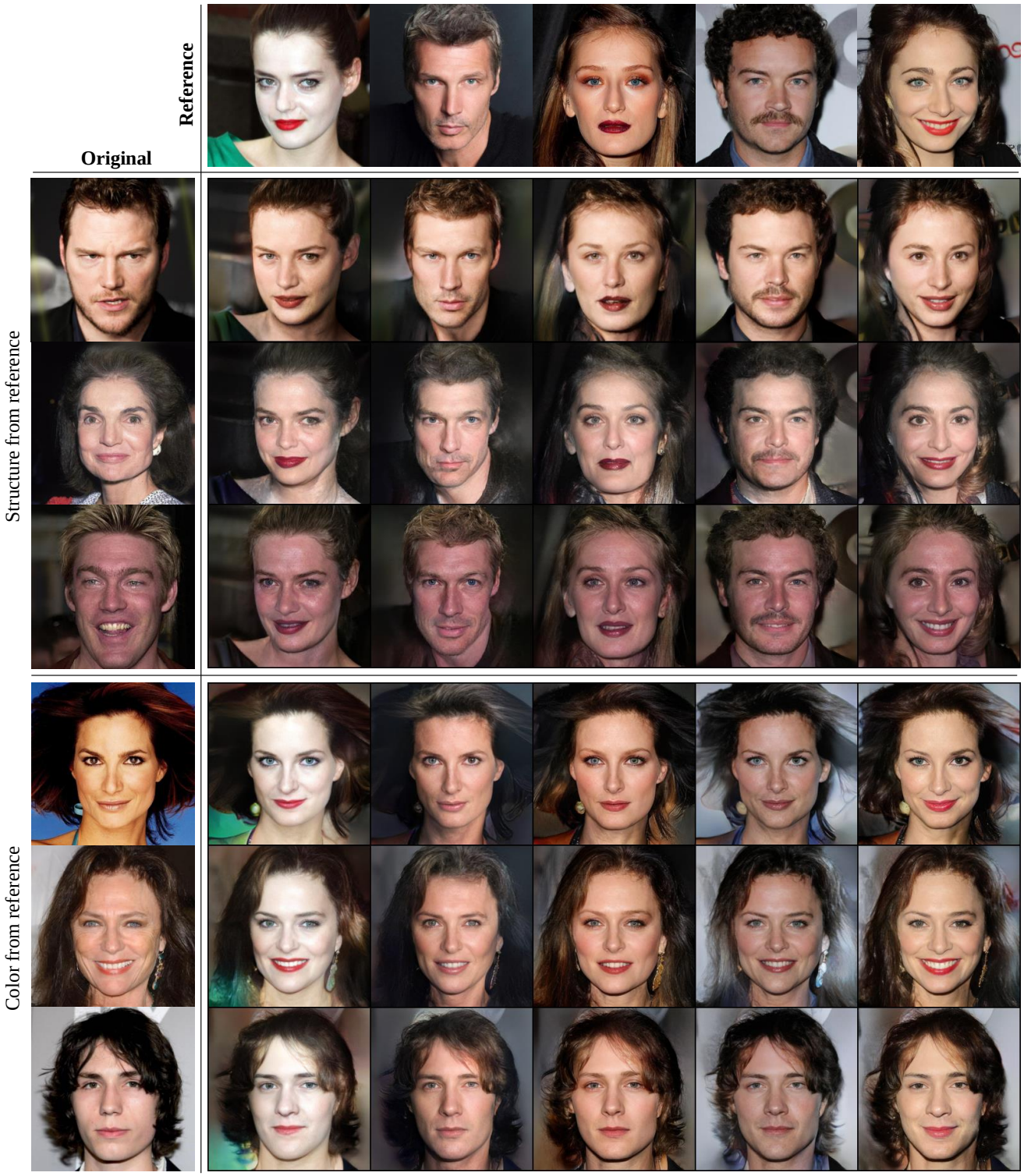


Figure 19: The results of style mixing in our method. Please refer to E.6 for details.

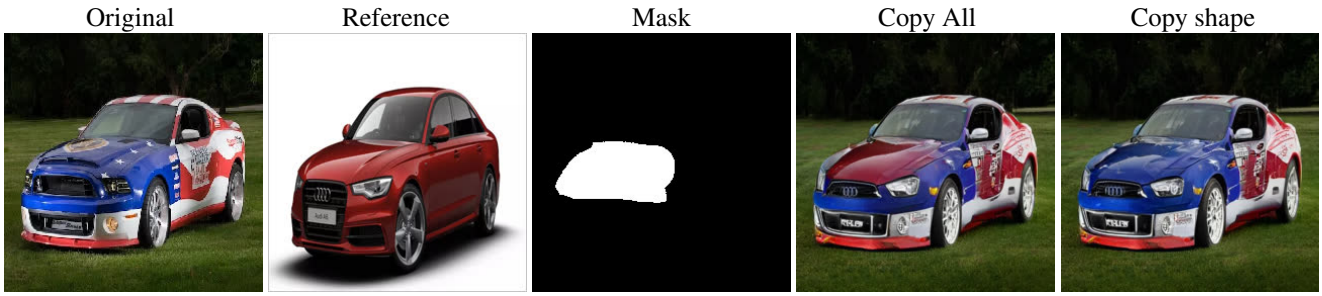


Figure 20: 4th column shows the transplantation of all identity including shape, texture, and color. The rightmost image shows the transplantation of structure alone.

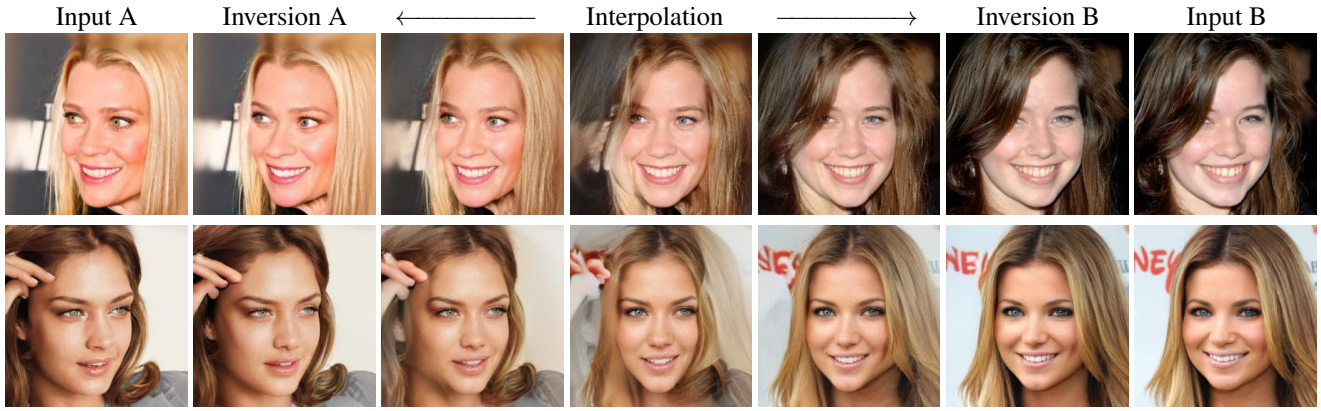


Figure 21: Failure cases of interpolation in our method due to extreme pose difference.

Figure 22: Failure cases of transplantation in our method due to different sizes of the masks.