Image2StyleGAN++: How to Edit the Embedded Images?

Rameen Abdal
KAUST
rameen.abdal@kaust.edu.sa

Yipeng Qin
Cardiff University
qiny16@cardiff.ac.uk

Peter Wonka
KAUST
pwonka@gmail.com

(a) (b) (c) (d)

Figure 1: (a) and (b): input images; (c): the "two-face" generated by naively copying the left half from (a) and the right half from (b); (d): the "two-face" generated by our Image2StyleGAN++ framework.

Abstract

We propose Image2StyleGAN++, a flexible image editing framework with many applications. Our framework extends the recent Image2StyleGAN [1] in three ways. First, we introduce noise optimization as a complement to the W^+ latent space embedding. Our noise optimization can restore high frequency features in images and thus significantly improves the quality of reconstructed images, e.g. *a big increase of PSNR from 20 dB to 45 dB. Second, we extend the global W^+ latent space embedding to enable local embeddings. Third, we combine embedding with activation tensor manipulation to perform high quality local edits along with global semantic edits on images. Such edits motivate various high quality image editing applications,* e.g. *image reconstruction, image inpainting, image crossover, local style transfer, image editing using scribbles, and attribute level feature transfer. Examples of the edited images are shown across the paper for visual inspection.*

1. Introduction

Recent GANs [18, 6] demonstrated that synthetic images can be generated with very high quality. This motivates research into embedding algorithms that embed a given photograph into a GAN latent space. Such embedding algorithms can be used to analyze the limitations of GANs [5], do image inpainting [8, 40, 38, 36], local image editing [41, 16], global image transformations such as image morphing and expression transfer [1], and few-shot video generation [35, 34].

In this paper, we propose to extend a very recent embedding algorithm, Image2StyleGAN [1]. In particular, we would like to improve this previous algorithm in three aspects. First, we noticed that the embedding quality can be further improved by including Noise space optimization into the embedding framework. The key insight here is that stable Noise space optimization can only be conducted if the optimization is done sequentially with W^+ space and not jointly. Second, we would like to improve the capabilities of the embedding algorithm to increase the local control over the embedding. One way to improve local control is to include masks in the embedding algorithm with undefined content. The goal of the embedding algorithm should be to find a plausible embedding for everything outside the mask, while filling in reasonable semantic content in the masked pixels. Similarly, we would like to provide the option of approximate embeddings, where the specified pixel colors are only a guide for the embedding. In this way, we aim to achieve high quality embeddings that can be controlled by user scribbles. In the third technical part of the paper, we investigate the combination of embedding algorithm and di-

rect manipulations of the activation maps (called activation tensors in our paper).

Our main contributions are:

1. We propose Noise space optimization to restore the high frequency features in an image that cannot be reproduced by other latent space optimization of GANs. The resulting images are very faithful reconstructions of up to 45 dB compared to about 20 dB (PSNR) for the previously best results.

2. We propose an extended embedding algorithm into the W^+ space of StyleGAN that allows for local modifications such as missing regions and locally approximate embeddings.

3. We investigate the combination of embedding and activation tensor manipulation to perform high quality local edits along with global semantic edits on images.

4. We apply our novel framework to multiple image editing and manipulation applications. The results show that the method can be successfully used to develop a state-of-the-art image editing software.

2. Related Work

Generative Adversarial Networks (GANs) [13, 29] are one of the most popular generative models that have been successfully applied to many computer vision applications, *e.g*. object detection [22], texture synthesis [21, 37, 31], image-to-image translation [15, 43, 28, 24] and video generation [33, 32, 35, 34]. Backing these applications are the massive improvements on GANs in terms of architecture [18, 6, 28, 15], loss function design [25, 2], and regularization [27, 14]. On the bright side, such improvements significantly boost the quality of the synthesized images. To date, the two highest quality GANs are StyleGAN [18] and BigGAN [6]. Between them, StyleGAN produces excellent results for unconditional image synthesis tasks, especially on face images; BigGAN produces the best results for conditional image synthesis tasks (*e.g*. ImageNet [9]). While on the dark side, these improvements make the training of GANs more and more expensive that nowadays it is almost a privilege of wealthy institutions to compete for the best performance. As a result, methods built on pre-trained generators start to attract attention very recently. In the following, we would like to discuss previous work of two such approaches: embedding images into a GAN latent space and the manipulation of GAN activation tensors.

Latent Space Embedding. The embedding of an image into the latent space is a longstanding topic in both machine learning and computer vision. In general, the embedding can be implemented in two ways: i) passing the input image through an encoder neural network (*e.g*. the Variational Auto-Encoder [20]); ii) optimizing a random initial latent code to match the input image [42, 7]. Between them, the first approach dominated for a long time. Although it has an inherent problem to generalize beyond the training dataset, it produces higher quality results than the naive latent code optimization methods [42, 7]. While recently, Abdal *et al*. [1] obtained excellent embedding results by optimizing the latent codes in an enhanced W^+ latent space instead of the initial Z latent space. Their method suggests a new direction for various image editing applications and makes the second approach interesting again.

Activation Tensor Manipulation. With fixed neural network weights, the expression power of a generator can be fully utilized by manipulating its activation tensors. Based on this observation, Bau [4] *et al*. investigated what a GAN can and cannot generate by locating and manipulating relevant neurons in the activation tensors [4, 5]. Built on the understanding of how an object is "drawn" by the generator, they further designed a semantic image editing system that can add, remove or change the appearance of an object in an input image [3]. Concurrently, Frühstück *et al*. [11] investigated the potential of activation tensor manipulation in image blending. Observing that boundary artifacts can be eliminated by by cropping and combining activation tensors at early layers of a generator, they proposed an algorithm to create large-scale texture maps of hundreds of megapixels by combining outputs of GANs trained on a lower resolution.

3. Overview

Our paper is structured as follows. First, we describe an extended version of the Image2StyleGAN [1] embedding algorithm (See Sec. 4). We propose two novel modifications: 1) to enable local edits, we integrate various spatial masks into the optimization framework. Spatial masks enable embeddings of incomplete images with missing values and embeddings of images with approximate color values such as user scribbles. In addition to spatial masks, we explore layer masks that restrict the embedding into a set of selected layers. The early layers of StyleGAN [18] encode content and the later layers control the style of the image. By restricting embeddings into a subset of layers we can better control what attributes of a given image are extracted. 2) to further improve the embedding quality, we optimize for an additional group of variables n that control additive noise maps. These noise maps encode high frequency details and enable embedding with very high reconstruction quality.

Second, we explore multiple operations to directly manipulate activation tensors (See Sec. 5). We mainly explore

(a) (b) (c) (d) (e)

Figure 2: Joint optimization. (a): target image; (b): image embedded by jointly optimizing w and n using perceptual and pixel-wise MSE loss; (c): image embedded by jointly optimizing w and n using the pixel-wise MSE loss only; (d): the result of the previous column with n resampled; (e): image embedded by jointly optimizing w and n using perceptual and pixel-wise MSE loss for w and pixel-wise MSE loss for n.

(a) (b) (c) (d)

Figure 3: Alternating optimization. (a): target image; (b): image embedded by optimizing w only; (c): taking w from the previous column and subsequently optimizing n only; (d): taking the result from the previous column and optimizing w only.

spatial copying, channel-wise copying, and averaging,

Interesting applications can be built by combining multiple embedding steps and direct manipulation steps. As a stepping stone towards building interesting application, we describe in Sec. 6 common building blocks that consist of specific settings of the extended optimization algorithm.

Finally, in Sec. 7 we outline multiple applications enabled by Image2StyleGAN++: improved image reconstruction, image crossover, image inpainting, local edits using scribbles, local style transfer, and attribute level feature transfer.

4. An Extended Embedding Algorithm

We implement our embedding algorithm as a gradient-based optimization that iteratively updates an image starting from some initial latent code. The embedding is performed into two spaces using two groups of variables; the semantically meaningful W^+ space and a Noise space N_s encoding high frequency details. The corresponding groups of variables we optimize for are $w \in W^+$ and $n \in N_s$. The inputs to the embedding algorithm are target RGB images x and y (they can also be the same image), and up to three spatial masks (M_s, M_m, and M_p)

Algorithm 1 is the generic embedding algorithm used in the paper.

4.1. Objective Function

Our objective function consists of three different types of loss terms, *i.e.* the pixel-wise MSE loss, the perceptual loss [17, 10], and the style loss [12].

$$\begin{aligned} L = & \lambda_s L_{style}(M_s, G(w,n), y) \\ & + \frac{\lambda_{mse_1}}{N} \|M_m \odot (G(w,n) - x)\|_2^2 \\ & + \frac{\lambda_{mse_2}}{N} \|(1 - M_m) \odot (G(w,n) - y)\|_2^2 \\ & + \lambda_p L_{percept}(M_p, G(w,n), x) \end{aligned} \tag{1}$$

Where M_s, M_m , M_p denote the spatial masks, $\odot$ denotes the Hadamard product, G is the StyleGAN generator, n are the Noise space variables, w are the W^+ space variables, L_{style} denotes style loss from $conv3\_3$ layer of an Ima-

Figure 4: First column: original image; Second column: image embedded in W^+ Space (PSNR 19 to 22 dB); Third column: image embedded in W^+ and Noise space (PSNR 39 to 45 dB).

geNet pretrained VGG-16 network [30], $L_{percept}$ is the perceptual loss defined in Image2StyleGAN [1]. Here, we use layers $conv1\_1$, $conv1\_2$, $conv2\_2$ and $conv3\_3$ of VGG-16 for the perceptual loss. Note that the perceptual loss is computed for four layers of the VGG network. Therefore, M_p needs to be downsampled to match the resolutions of the corresponding VGG-16 layers in the computation of the loss function.

4.2. Optimization Strategies

Optimization of the variables $w \in W^+$ and $n \in N_s$ is not a trivial task. Since only $w \in W^+$ encodes semantically meaningful information, we need to ensure that as much information as possible is encoded in w and only high frequency details in the Noise space.

The first possible approach is the joint optimization of both groups of variables w and n. Fig.2 (b) shows the result using the perceptual and the pixel-wise MSE loss. We can observe that many details are lost and were replaced with high frequency image artifacts. This is due to the fact that the perceptual loss is incompatible with optimizing noise maps. Therefore, a second approach is to use pixel-wise MSE loss only (see Fig. 2 (c)). Although the reconstruction is almost perfect, the representation (w, n) is not suitable for image editing tasks. In Fig. 2 (d), we show that too much of the image information is stored in the noise layer, by resampling the noise variables n. We would expect to obtain another very good, but slightly noisy embedding. Instead, we obtain a very low quality embedding. Also, we show the result of jointly optimizing the variables and using perceptual and pixel-wise MSE loss for w variables and pixel-wise MSE loss for the noise variable. Fig. 2 (e) shows the reconstructed image is not of high perceptual quality. The PSNR score decreases to 33.3 dB. We also tested these optimizations on other images. Based on our results, we do not recommend using joint optimization.

The second strategy is an alternating optimization of the variables w and n. In Fig. 3, we show the result of optimizing w while keeping n fixed and subsequently optimizing n while keeping w fixed. In this way, most of the information is encoded in w which leads to a semantically meaningful embedding. Performing another iteration of optimizing w (Fig. 3 (d)) reveals a smoothing effect on the image and the PSNR reduces from 39.5 dB to 20 dB. Subsequent Noise space optimization does not improve PSNR of the images. Hence, repetitive alternating optimization does not improve the quality of the image further. In summary, we recommend to use alternating optimization, but each set of variables is only optimized once. First we optimize w, then n.

Algorithm 1: Semantic and Spatial component embedding in StyleGAN

Input: images $x, y \in \mathbb{R}^{n \times m \times 3}$; masks M_s, M_m, M_p; a pre-trained generator $G(\cdot, \cdot)$; gradient-based optimizer F'.

Output: the embedded code (w, n)

```
Initialize() the code (w, n) = (w', n');
while not converged do
    Loss ← L(x, y, M_s, M_m, M_p);
    (w, n) ← (w, n) − ηF'(∇_{w,n} L, w, n);
end
```

5. Activation Tensor Manipulations

Due to the progressive architecture of StyleGAN, one can perform meaningful tensor operations at different layers of the network [11, 4]. We consider the following editing operations: spatial copying, averaging, and channel-wise copying. We define activation tensor A_l^I as the output of the l-th layer in the network initialized with variables (w, n) of the embedded image I. They are stored as ten-

Figure 5: First and second column: input image; Third column: image generated by naively copying the left half from the first image and the right half from the second image; Fourth column: image generated by our extended embedding algorithm.

sors $A_l^I \in R^{W_l \times H_l \times C_l}$. Given two such tensors A_l^I and B_l^I, copying replaces high-dimensional pixels $\in R^{1\times 1\times C_l}$ in A_l^I by copying from B_l^I. Averaging forms a linear combination $\lambda A_l^I + (1-\lambda)B_l^I$. Channel-wise copying creates a new tensor by copying selected channels from A_l^I and the remaining channels from B_l^I. In our tests we found that spatial copying works a bit better than averaging and channel-wise copying.

6. Frequently Used Building Blocks

We identify four fundamental building blocks that are used in multiple applications described in Sec. 7. While terms of the loss function can be controlled by spatial masks (M_s, M_m, M_p), we also use binary masks w_m and n_m to indicate what subset of variables should be optimized during an optimization process. For example, we might set w_m to only update the w variables corresponding to the first k layers. In general, w_m and n_m contain 1s for variables that should be updated and 0s for variables that should remain constant. In addition to the listed parameters, all building blocks need initial variable values w_{ini} and n_{ini}. For all experiments, we use a 32GB Nvidia V100 GPU.

Masked W^+ optimization (W_l): This function optimizes $w \in W^+$, leaving n constant. We use the following parameters in the loss function (L) Eq. 1: $\lambda_s = 0$, $\lambda_{mse_1} = 10^{-5}$, $\lambda_{mse_2} = 0$, $\lambda_p = 10^{-5}$. We denote the function as:

$$\begin{aligned} &W_l(M_p, M_m, w_m, w_{ini}, n_{ini}, x) = \\ &\arg\min_{w_m} \lambda_p L_{percept}(M_p, G(w,n), x) + \\ &\qquad \frac{\lambda_{mse_1}}{N} \|M_m \odot (G(w,n) - x)\|_2^2 \end{aligned} \tag{2}$$

where w_m is a mask for W^+ space. We either use Adam [19] with learning rate 0.01 or gradient descent with learning rate 0.8, depending on the application. Some common settings for Adam are: $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1e^{-8}$. In Sec. 7, we use Adam unless specified.

Masked Noise Optimization (Mk_n): This function optimizes $n \in N_s$, leaving w constant. The Noise space N_s has dimensions $\{\mathbb{R}^{4\times 4}, \ldots, \mathbb{R}^{1024\times 1024}\}$. In total there are 18 noise maps, two for each resolution. We set following parameters in the loss function (L) Eq. 1: $\lambda_s = 0$, $\lambda_{mse_1} = 10^{-5}$, $\lambda_{mse_2} = 10^{-5}$, $\lambda_p = 0$. We denote the function as:

$$\begin{aligned} &Mk_n(M, w_{ini}, n_{ini}, x, y) = \\ &\arg\min_{n} \frac{\lambda_{mse_2}}{N} \|M_m \odot (G(w,n) - x)\|_2^2 + \\ &\qquad \frac{\lambda_{mse_1}}{N} \|(1 - M_m) \odot (G(w,n) - y)\|_2^2 \end{aligned} \tag{3}$$

For this optimization, we use Adam with learning rate 5, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1e^{-8}$. Note that the learning rate is very high.

Masked Style Transfer(M_{st}): This function optimizes w to achieve a given target style defined by style image y. We set following parameters in the loss function (L) Eq. 1: $\lambda_s = 5 \times 10^{-7}$, $\lambda_{mse_1} = 0$, $\lambda_{mse_2} = 0$, $\lambda_p = 0$. We denote the function as:

$$\begin{aligned} &M_{st}(M_s, w_{ini}, n_{ini}, y) = \\ &\quad \arg\min_{w} \lambda_s L_{style}(M_s, G(w,n), y) \end{aligned} \tag{4}$$

where w is the whole W^+ space. For this optimization, we use Adam with learning rate 0.01, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1e^{-8}$.

Masked activation tensor operation (I_{att}): This function describes an activation tensor operation. Here, we represent the generator $G(w, n, t)$ as a function of W^+ space variable w, Noise space variable n, and input tensor t. The operation is represented by:

$$\begin{aligned} &I_{att}(M_1, M_2, w, n_{ini}, l) = \\ &\quad G(w, n, M_1 \odot (A_l^{I_1}) + (1 - M_2) \odot (B_l^{I_2})) \end{aligned} \tag{5}$$

where $A_l^{I_1}$ and $B_l^{I_2}$ are the activations corresponding to images I_1 and I_2 at layer l, and M_1 and M_2 are the masks downsampled using nearest neighbour interpolation to match the $H_l \times W_l$ resolution of the activation tensors.

7. Applications

In the following we describe various applications enabled by our framework.

Algorithm 2: Improved Image Reconstruction

Input: image $I_m \in \mathbb{R}^{n \times m \times 3}$
Output: the embedded code (w_{out}, n_{out})

$(w_{ini}, n_{ini}) \leftarrow$ initialize();
$w_{out} = W_l(1, 1, 1, w_{ini}, n_{ini}, I_m)$;
$n_{out} = Mk_n(1, w_{out}, n_{ini}, I_m, 0)$;

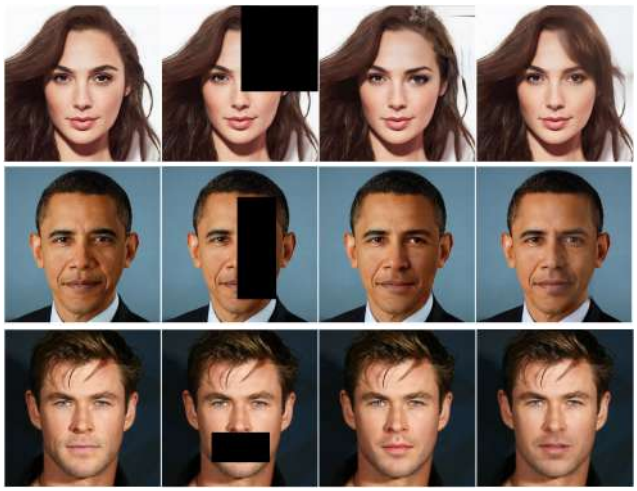

Figure 6: First column: original image; Second column: defective image ; Third column: inpainted image via partial convolutions [23]; Fourth column: inpainted image using our method.

7.1. Improved Image Reconstruction

As shown in Fig. 4, any image can be embedded by optimizing for variables $w \in W^+$ and $n \in N_s$. Here we describe the details of this embedding (See Alg. 2). First, we initialize: w_{ini} is a mean face latent code [18] or random code sampled from $U[-1, 1]$ depending on whether the embedding image is a face or a non-face, and n_{ini} is sampled from a standard normal distribution $N(0, I)$ [18]. Second, we apply masked W^+ optimization (W_l) without using spatial masks or masking variables. That means all masks are set to 1. I_m is the target image we try to reconstruct. Third, we perform masked noise optimization (Mk_n), again without making use of masks. The images reconstructed are of high fidelity. The PNSR score range of 39 to 45 dB provides an insight of how expressive the Noise space in StyleGAN is. Unlike the W^+ space, the Noise space is used for spatial reconstruction of high frequency features. We use 5000 iterations of W_l and 3000 iterations of Mk_n to get PSNR scores of 44 to 45 dB. Additional iterations did not improve the results in our tests.

Figure 7: Inpainting using different initializations w_{ini}.

Algorithm 3: Image Crossover

Input: images $I_1, I_2 \in \mathbb{R}^{n \times m \times 3}$; mask M_{blur}
Output: the embedded code (w_{out}, n_{out})

$(w^*, n_{ini}) \leftarrow$ initialize();
$w_{out} = W_l(M_{blur}, M_{blur}, 1, w^*, n_{ini}, I_1)$
$+W_l(1 - M_{blur}, 1 - M_{blur}, 1, w^*, n_{ini}, I_2)$;
$n_{out} = Mk_n(M_{blur}, w_{out}, n_{ini}, I_1, I_2)$;

7.2. Image Crossover

We define the image crossover operation as copying parts from a source image y into a target image x and blending the boundaries. As initialization, we embed the target image x to obtain the W^+ code w^*. We then perform masked W^+ optimization (W_l) with blurred masks M_{blur} to embed the regions in x and y that contribute to the final image. Blurred masks are obtained by convolution of the binary mask with a Gaussian filter of suitable size. Then, we perform noise optimization. Details are provided in Alg. 3.

Other notations are the same as described in Sec 7.1. Fig. 5 and Fig. 1 show example results. We deduce that the reconstruction quality of the images is quite high. For the experiments, we use 1000 iterations in the function masked W^+ optimization and 1000 iterations in Mk_n.

7.3. Image Inpainting

Algorithm 4: Image Inpainting

Input: image $I_{def} \in \mathbb{R}^{n \times m \times 3}$; masks M, M_{blur+}
Output: the embedded code (w_{out}, n_{out})

$(w_{ini}, n_{ini}) \leftarrow$ initialize();
$w_{out} = W_l(1 - M, 1 - M, w_m, w_{ini}, n_{ini}, I_{def})$;
$n_{out} = Mk_n(1 - M_{blur+}, w_{out}, n_{ini}, I_{def}, G(w_{out}))$;

In order to perform a semantically meaningful inpainting, we embed into the early layers of the W^+ space to predict the missing content and in the later layers to maintain color consistency. We define the image x as a defec-

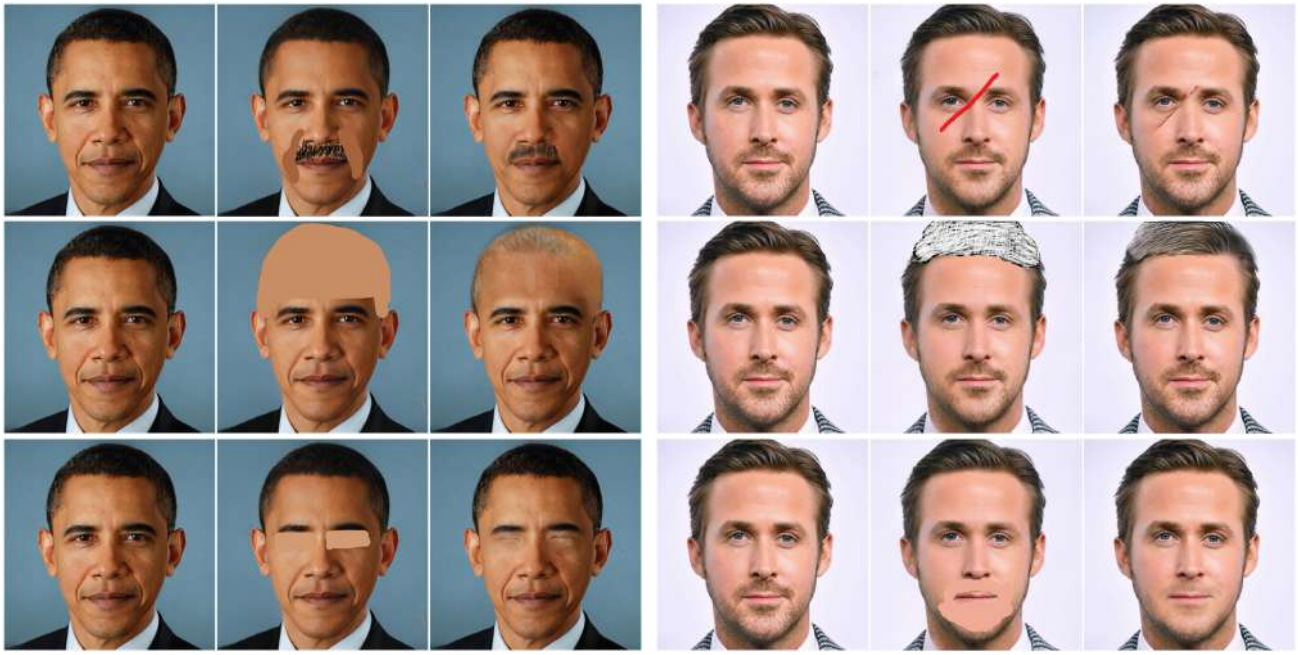

Figure 8: Column 1 & 4: base image; Column 2 & 5: scribbled image ; Column 3 & 6: result of local edits.

tive image (I_{def}). Also, we use the mask w_m where the value is 1 corresponding to the first 9 (1 to 9), 17^{th} and 18^{th} layer of W^+. As an initialization, we set w_{ini} to the mean face latent code [18]. We consider M as the mask describing the defective region. Using these parameters, we perform the masked W^+ optimization W_l. Then we perform the masked noise optimization Mk_n using M_{blur+} which is the slightly larger blurred mask used for blending. Here λ_{mse_2} is taken to be 10^{-4}. Other notations are the same as described in Sec 7.1. Alg. 4 shows the details of the algorithm. We perform 200 steps of gradient descent optimizer for masked W^+ optimization W_l and 1000 iterations of masked noise optimization Mk_n. Fig.6 shows example inpainting results. The results are comparable with the current state of the art, partial convolution [23]. The partial convolution method frequently suffers from regular artifacts (see Fig.6 (third column)). These artifacts are not present in our method. In Fig.7 we show different inpainting solutions for the same image achieved by using different initializations of w_{ini} , which is an offset to mean face latent code sampled independently from a uniform distribution $U[-0.4, 0.4]$. The initialization mainly affects layers 10 to 16 that are not altered during optimization. Multiple inpainting solutions cannot be computed with existing state-of-the-art methods.

7.4. Local Edits using Scribbles

Another application is performing semantic local edits guided by user scribbles. We show that simple scribbles can be converted to photo-realistic edits by embedding into the first 4 to 6 layers of W^+ (See Fig.8). This enables us to do local edits without training a network. We define an image x as a scribble image (I_{scr}). Here, we also use the mask w_m where the value is 1 corresponding to the first 4,5 or 6 layers of the W^+ space. As initialization, we set the w_{ini} to w^* which is the W^+ code of the image without scribble. We

Figure 9: First column: base image; Second column: mask area; Third column: style image; Fourth column: local style transfer result.

Algorithm 5: Local Edits using Scribble

Input: image $I_{scr} \in \mathbb{R}^{n \times m \times 3}$; masks M_{blur}
Output: the embedded code (w_{out}, n_{out})

$(w^*, n_{ini}) \leftarrow$ initialize();
$w_{out} = W_l(1, 1, w_m, w^*, n_{ini}, I_{scr})$
$+\lambda \|w^* - w_{out}\|_2$;
$n_{out} = Mk_n(M_{blur}, w_{out}, n_{ini}, I_{scr}, G(w_{out}))$;

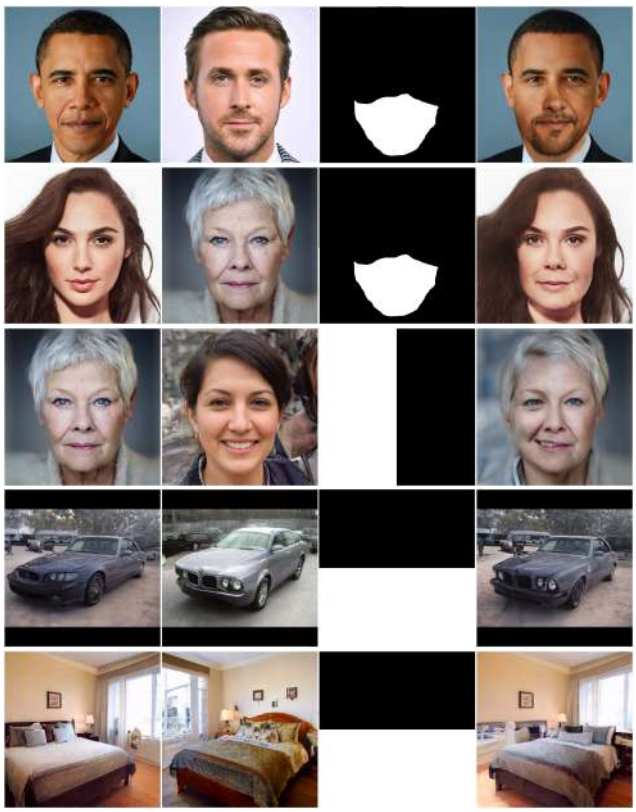

Figure 10: First column: base image; Second column: attribute image; Third column: mask area; Fourth column: image generated via attribute level feature transfer.

Algorithm 6: Local Style Transfer

Input: images $I_1, I_2 \in \mathbb{R}^{n \times m \times 3}$; masks M_{blur}
Output: the embedded code (w_{out}, n_{out})

$(w^*, n_{ini}) \leftarrow$ initialize();
$w_{out} = W_l(M_{blur}, M_{blur}, 1, w^*, n_{ini}, I_1)$
$+M_{st}(1 - M_{blur}, w^*, n_{ini}, I_2)$;
$n_{out} = Mk_n(M_{blur}, w_{out}, n_{ini}, I_1, G(w_{out}))$;

perform masked W^+ optimization using these parameters. Then we perform masked noise optimization Mk_n using M_{blur}. Other notations are the same as described in Sec 7.1. Alg. 5 shows the details of the algorithm. We perform 1000 iterations using Adam with a learning rate of 0.1 of masked W^+ optimization W_l and then 1000 steps of masked noise optimization Mk_n to output the final image.

7.5. Local Style Transfer

Local style transfer modifies a region in the input image x to transform it to the style defined by a style reference image. First, we embed the image in W^+ space to obtain the code w^*. Then we apply the masked W^+ optimization W_l along with masked style transfer M_{st} using blurred mask M_{blur}. Finally, we perform the masked noise optimization Mk_n to output the final image. Alg. 6 shows the details of the algorithm. Results for the application are shown in Fig.9. We perform 1000 steps to obtain of W_l along with M_{st} and then perform 1000 iterations of Mk_n.

7.6. Attribute level feature transfer

We extend our work to another application using tensor operations on the images embedded in W^+ space. In this application we perform the tensor manipulation corresponding to the tensors at the output of the 4^{th} layer of StyleGAN. We feed the generator with the latent codes (w, n) of two images I_1 and I_2 and store the output of the fourth layer as intermediate activation tensors $A_l^{I_1}$ and $B_l^{I_2}$. A mask M_s specifies which values to copy from $A_l^{I_1}$ and which to copy from $B_l^{I_2}$. The operation can be denoted by $I_{att}(M_s, M_s, w, n_{ini}, 4)$. In Fig.10, we show results of the operation. A design parameter of this application is what style code to use for the remaining layers. In the shown example, the first image is chosen to provide the style. Notice, in column 2 of Fig.10, in-spite of the different alignment of the two faces and objects, the images are blended well. We also show results of blending for the LSUN-car and LSUN-bedroom datasets. Hence, unlike global edits like image morphing, style transfer, and expression transfer [1], here different parts of the image can be edited independently and the edits are localized. Moreover, along with other edits, we show a video in the supplementary material that further shows that other semantic edits e.g. masked image morphing can be performed on such images by linear interpolation of W^+ code of one image at a time.

8. Conclusion

We proposed Image2StyleGAN++, a powerful image editing framework built on the recent Image2StyleGAN. Our framework is motivated by three key insights: first, high frequency image features are captured by the additive noise maps used in StyleGAN, which helps to improve the quality of reconstructed images; second, local edits are enabled by including masks in the embedding algorithm, which greatly increases the capability of the proposed framework; third, a variety of applications can be created by combining embedding with activation tensor manipulation. From the high quality results presented in this paper, it can be concluded that our Image2StyleGAN++ is a promising framework for general image editing. For future work, in addition to static images, we aim to extend our framework to process and edit videos.

Acknowledgement This work was supported by the KAUST Office of Sponsored Research (OSR) under Award No. OSR-CRG2018-3730.

References

[1] R. Abdal, Y. Qin, and P. Wonka. Image2stylegan: How to embed images into the stylegan latent space? In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4432–4441, 2019. 1, 2, 4, 8

[2] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 214–223, 2017. 2

[3] D. Bau, H. Strobelt, W. Peebles, J. Wulff, B. Zhou, J. Zhu, and A. Torralba. Semantic photo manipulation with a generative image prior. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, 38(4), 2019. 2

[4] D. Bau, J.-Y. Zhu, H. Strobelt, B. Zhou, J. B. Tenenbaum, W. T. Freeman, and A. Torralba. Gan dissection: Visualizing and understanding generative adversarial networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019. 2, 4

[5] D. Bau, J.-Y. Zhu, J. Wulff, W. Peebles, H. Strobelt, B. Zhou, and A. Torralba. Seeing what a gan cannot generate. In *Proceedings of the International Conference Computer Vision (ICCV)*, 2019. 1, 2

[6] A. Brock, J. Donahue, and K. Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2019. 1, 2

[7] A. Creswell and A. A. Bharath. Inverting the generator of a generative adversarial network. *IEEE Transactions on Neural Networks and Learning Systems*, 2018. 2

[8] U. Demir and G. Unal. Patch-based image inpainting with generative adversarial networks. *arXiv preprint arXiv:1803.07422*, 2018. 1

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009. 2

[10] A. Dosovitskiy and T. Brox. Generating images with perceptual similarity metrics based on deep networks. In *Advances in neural information processing systems*, pages 658–666, 2016. 3

[11] A. Frhstck, I. Alhashim, and P. Wonka. Tilegan. *ACM Transactions on Graphics*, 38(4):111, Jul 2019. 2, 4

[12] L. A. Gatys, A. S. Ecker, and M. 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

[13] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, 2014. 2

[14] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. In *Advances in neural information processing systems*, pages 5767–5777, 2017. 2

[15] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *CVPR*, 2017. 2

[16] Y. Jo and J. Park. Sc-fegan: Face editing generative adversarial network with user's sketch and color. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019. 1

[17] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, 2016. 3

[18] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. *arXiv preprint arXiv:1812.04948*, 2018. 1, 2, 6, 7, 11

[19] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. 2014. 5

[20] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 2

[21] C. Li and M. Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III*, 2016. 2

[22] J. Li, X. Liang, Y. Wei, T. Xu, J. Feng, and S. Yan. Perceptual generative adversarial networks for small object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 2

[23] G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro. Image inpainting for irregular holes using partial convolutions. *Lecture Notes in Computer Science*, page 89105, 2018. 6, 7, 11, 12, 13

[24] M.-Y. Liu, X. Huang, A. Mallya, T. Karras, T. Aila, J. Lehtinen, and J. Kautz. Few-shot unsueprvised image-to-image translation. In *arxiv*, 2019. 2

[25] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. P. Smolley. Least squares generative adversarial networks. *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 2

[26] Microsoft. Microsoft azure face. https://azure.microsoft.com/en-us/services/cognitive-services/face/. 11

[27] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018. 2

[28] T. Park, M.-Y. Liu, T.-C. Wang, and J.-Y. Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 2

[29] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015. 2

[30] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. 2014. 4

[31] R. Slossberg, G. Shamai, and R. Kimmel. High quality facial surface and texture synthesis via generative adversarial networks. In *European Conference on Computer Vision*, pages 498–513. Springer, 2018. 2

[32] S. Tulyakov, M.-Y. Liu, X. Yang, and J. Kautz. Mocogan: Decomposing motion and content for video generation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[33] C. Vondrick, H. Pirsiavash, and A. Torralba. Generating videos with scene dynamics. In *Advances in Neural Information Processing Systems 29*. 2016. 2

[34] T.-C. Wang, M.-Y. Liu, A. Tao, G. Liu, J. Kautz, and B. Catanzaro. Few-shot video-to-video synthesis. *arXiv preprint arXiv:1910.12713*, 2019. 1, 2

[35] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, G. Liu, A. Tao, J. Kautz, and B. Catanzaro. Video-to-video synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. 1, 2

[36] R. Webster, J. Rabin, L. Simon, and F. Jurie. Detecting overfitting of deep generative networks via latent recovery. 2019. 1

[37] W. Xian, P. Sangkloy, V. Agrawal, A. Raj, J. Lu, C. Fang, F. Yu, and J. Hays. Texturegan: Controlling deep image synthesis with texture patches. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[38] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. Huang. Free-form image inpainting with gated convolution. 2018. 1

[39] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang. Free-form image inpainting with gated convolution. *arXiv preprint arXiv:1806.03589*, 2018. 11, 12, 14

[40] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang. Generative image inpainting with contextual attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5505–5514, 2018. 1

[41] J.-Y. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros. Generative visual manipulation on the natural image manifold. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2016. 1

[42] J.-Y. Zhu, P. Krhenbhl, E. Shechtman, and A. A. Efros. Generative visual manipulation on the natural image manifold. *Lecture Notes in Computer Science*, page 597613, 2016. 2

[43] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networkss. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017. 2

9. Additional Results

9.1. Image Inpainting

To evaluate the results quantitatively, we use three standard metrics, SSIM, MSE loss and PSNR score to compare our method with the state-of-the-art Partial Convolution [23] and Gated Convolution [39] methods.

As different methods produce outputs at different resolutions, we bi-linearly interpolate the output images to test the methods at three resolutions 1024×1024, 512×512 and 256×256 respectively. We use 7 masks (Fig. 11) and 10 ground truth images (Fig. 12) to create 10 defective images (*i.e*. images with missing regions) for the evaluation. These masks and images are chosen to make the inpainting a challenging task: i) the masks are selected to contain very large missing regions, up to half of an image; ii) the ground truth images are selected to be of high variety that cover different genders, ages, races, *etc*.

Table 1 shows the quantitative comparison results. It can be observed that our method outperforms both Partial Convolution [23] and Gated Convolution [39] across all the metrics. More importantly, the advantages of our method can be easily verified by visual inspection. As Fig. 13 and Fig. 14 show, although previous methods (*e.g*. Partial convolution) perform well when the missing region is small, both of them struggle when the missing region covers a significant area (*e.g*. half) of the image. Specifically, Partial Convolution fails when the mask covers half of the input image (Fig. 13); due to the relatively small resolution (256×256) model, Gated Convolution can fill in the details of large missing regions, but of much lower quality compared to the proposed method (Fig. 14).

In addition, our method is flexible and can generate different inpainting results (Fig. 15), which cannot be fulfilled by any of the above-mentioned methods. All our inpainting results are of high perceptual quality.

Limitations Although better than the two state-of-the-art methods, our inpainting results still leave room for improvement. For example in Fig. 13, the lighting condition (first row), age (second row) and skin color (third and last row) are not learnt that well. We propose to address them in the future work.

Figure 11: Masks used in the quantitative evaluation of image inpainting methods.

9.2. Image Crossover

To further evaluate the expressibility of the Noise space, we show additional results on image crossover in Fig. 16. We show that the space is able to crossover parts of images from different races (see second and third column).

9.3. Local Edits using Scribbles

In order to evaluate the quality of the local edits using scribbles, we evaluate the face attribute scores [26] on edited images. We perform some common edits of adding baldness, adding a beard, smoothing wrinkles and adding a moustache on the face images to evaluate how photo-realistic the edited images are. Table 2 shows the average change in the confidence of the classifier after a particular edit is performed. We also show additional results of the Local edits in Fig. 17. For our method, one remaining challenge is that sometimes the edited region is overly smooth (e.g. first row).

9.4. Attribute Level Feature Transfer

We show a video in which attribute interpolation can be performed on the base image by copying the content from an attribute image. Here different attributes can be taken from different images embedded in the W^+ space and applied to the base image. These attributes can be independently interpolated and the results show that the blending quality of the framework is quite high. We also show additional results on LSUN Cars and LSUN Bedrooms in the video (also see Fig. 18). Notice that in the LSUN bedrooms, for instance, the style and the position of the beds can be customized without changing the room layout.

In order to evaluate the perceptual quality of attribute level feature transfer, we compute perceptual length [18] between the images produced by independently interpolated attributes (called masked interpolation). StyleGAN [18] showed that the metric evaluates how perceptually smooth the transitions are. Here, perceptual length measures the changes produced by feature transfer which may be affected especially by the boundary of the blending. The boundary may tend to produce additional artifacts or introduce additional features which is clearly undesirable.

We compute the perceptual length across 1000 samples using two masks shown in Fig. 11 (First and Seventh column). In Table 3 we show the results of the computation of the perceptual length (both for masked and non-masked interpolation) on FFHQ, LSUN Cars and LSUN Bedrooms pretrained StyleGAN. We compare these scores as the non-masked interpolation gives us the upper bound of the perceptual length for a model (in this case there is no constraint on what features of the face should change). As a particular area of the image is interpolated rather than the whole image, note that our results on FFHQ pretrained StyleGAN produce lower score than the non-masked interpolation. The low perceptual length score suggests that there is a less drastic change. Hence, we conclude that the output images have comparable perceptual quality with non-masked interpolation.

LSUN Cars and LSUN Bedrooms produce relatively higher perceptual length score. We attribute this result to the fact that the images in these datasets can translate and the position of the features is not fixed. Hence, the two images produced at random might have different orientation in which case the blending does not work as good.

Figure 12: Images used in the quantitative evaluation of image inpainting methods.

Method	Image Resolution (1024×1024)			Image Resolution (512×512)			Image Resolution (256×256)		
	SSIM	MSE	PSNR	SSIM	MSE	PSNR	SSIM	MSE	PSNR
Partial Convolution [23]	0.8957	199.39	21.83	0.8865	98.83	21.92	0.8789	48.39	22.17
Gated Convolution [39]	0.8693	246.46	19.65	0.8568	121.98	19.77	0.8295	61.82	19.41
Ours	**0.9176**	**180.69**	**22.35**	**0.9104**	**89.25**	**22.48**	**0.9009**	**43.85**	**22.65**

Table 1: Evaluation results of image inpainting methods using SSIM, MSE and PSNR score.

9.5. Channel wise feature average

We perform another operation denoted by $I_{att}(1, 0, w_x, , n_{ini}, 6)$, where w_x can be the W^+ code for images I_1 or I_2. In Fig. 19, we show the result of this operation which is initialized with two different W^+ codes. The resulting faces contain the characteristics of both faces and the styles are modulated by the input W^+ codes.

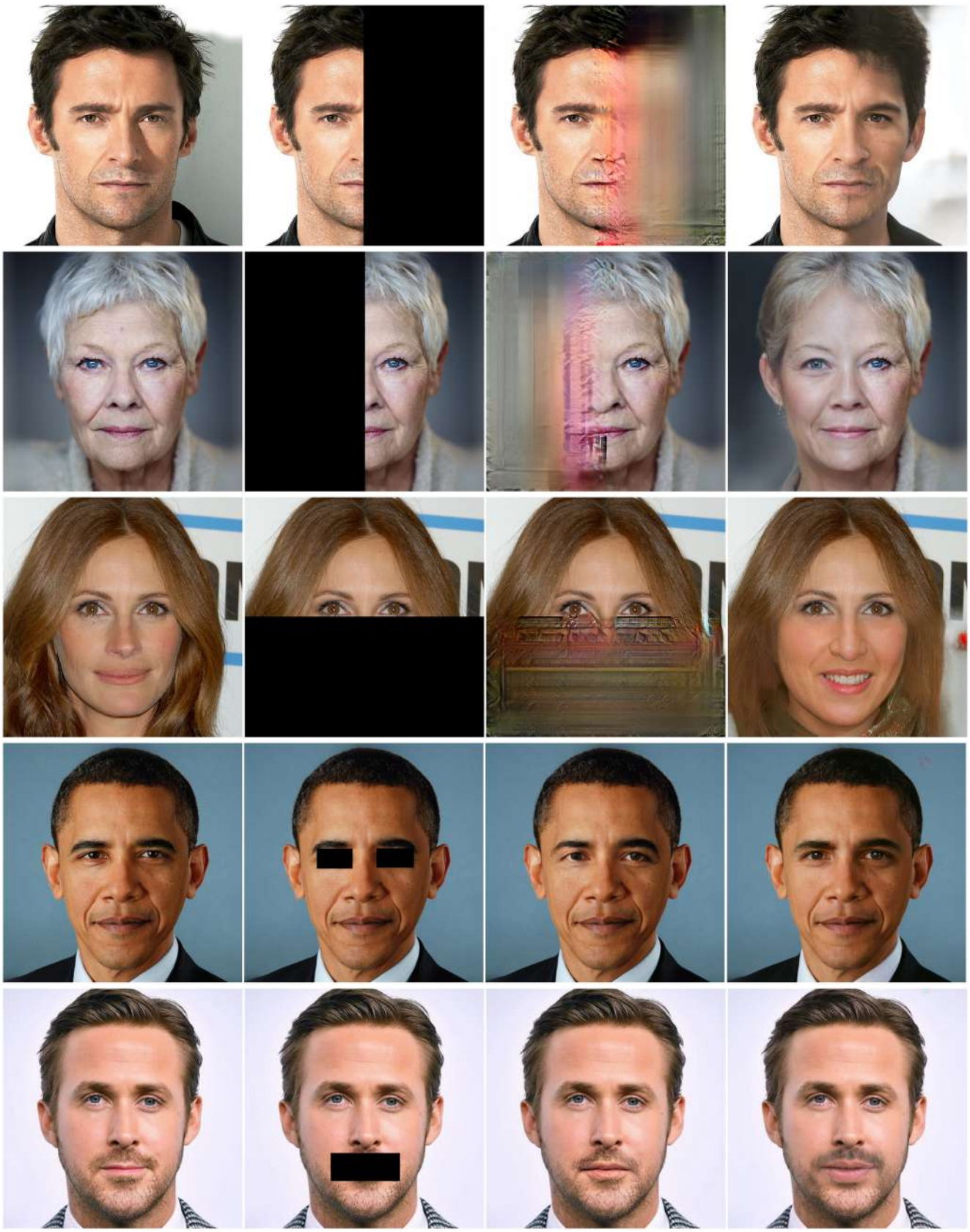

Figure 13: First column: original image; Second column: defective image; Third column: inpainted image via Partial Convolutions [23]; Fourth column: inpainted image using our method.

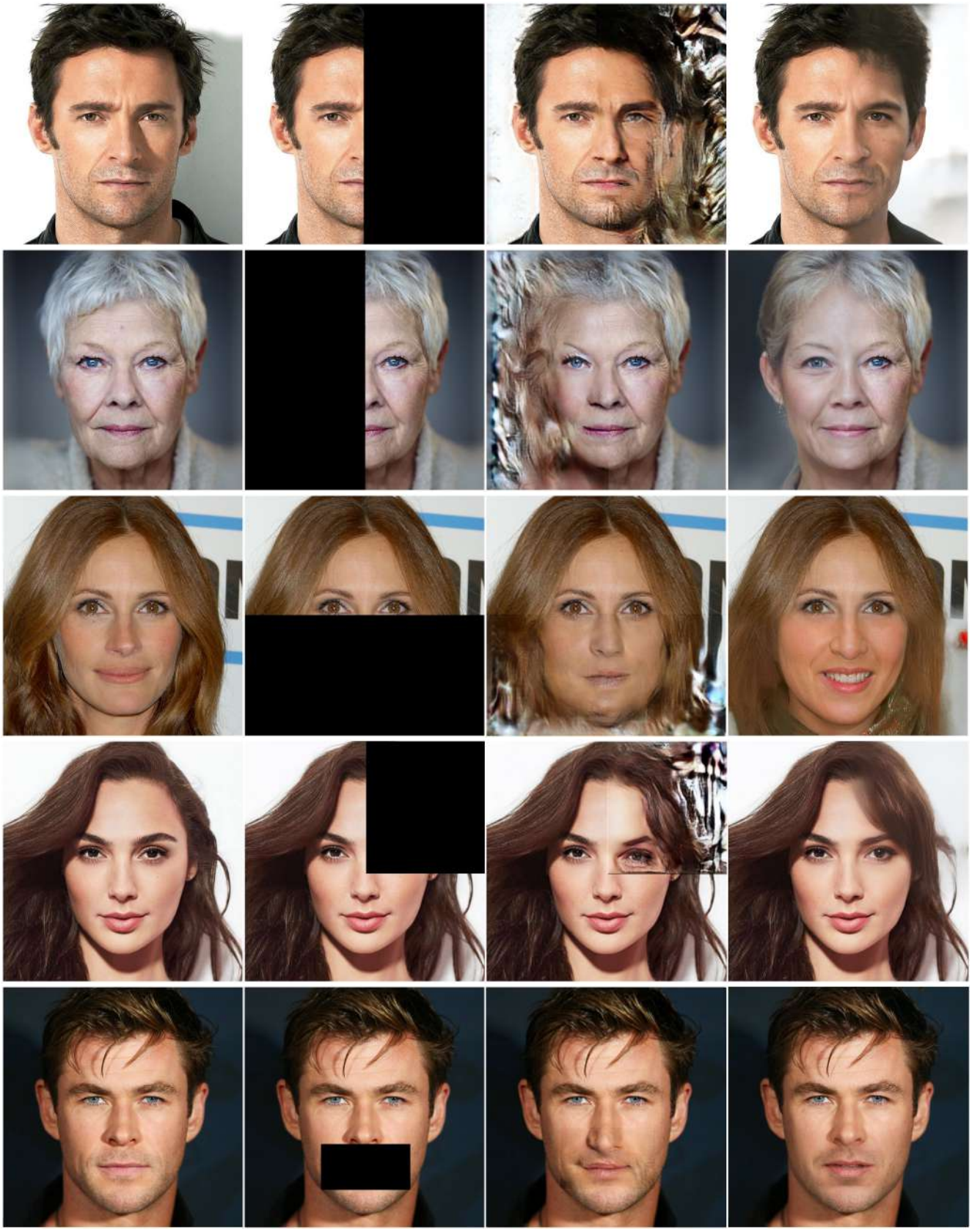

Figure 14: First column: original image; Second column: defective image; Third column: inpainted image via Gated Convolutions [39]; Fourth column: inpainted image using our method.

Edit	Attribute	Change in confidence
Wrinkle Smoothing	age	21%
Adding Baldness	bald	75%
Adding Beard	beard	42%
Adding Moustache	moustache	49%

Table 2: Changes in confidence scores of classifier after user edits.

Figure 15: Inpainting results using different w_{ini} initializations.

Pretrained model	Interpolation	Perceptual length (full)	Perceptual length (end)
FFHQ	Non-Masked	227.1	191.1
	Masked	112.1	89.8
LSUN Cars	Non-Masked	12388.1	6038.5
	Masked	4742.3	3057.9
LSUN Bedrooms	Non-Masked	2521.1	1268.7
	Masked	1629.8	938.1

Table 3: Perceptual length evaluation for masked and non-masked interpolation.

Figure 16: (a) and (b): input images; (c): the “two-face” generated by naively copying the left half from (a) and the right half from (b); (d): the “two-face” generated by our Image2StyleGAN++ framework.

Figure 17: Column 1 & 4: base image; Column 2 & 5: scribbled image ; Column 3 & 6: result of local edits.

Figure 18: First column: base image; Second column: mask area; Third column: attribute image; Fourth to Eighth column: image generated via attribute level feature transfer and masked interpolation.

Figure 19: First column: First Image; Second Column: Second Image; Third Column: Feature averaged image using W^+ code of first image; Fourth Column: Feature averaged image using W^+ code of second image.