# Image2StyleGAN: How to Embed Images Into the StyleGAN Latent Space?

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

Yipeng Qin
KAUST
yipeng.qin@kaust.edu.sa

Peter Wonka
KAUST
pwonka@gmail.com

## Abstract

*We propose an efficient algorithm to embed a given image into the latent space of StyleGAN. This embedding enables semantic image editing operations that can be applied to existing photographs. Taking the StyleGAN trained on the FFHQ dataset as an example, we show results for image morphing, style transfer, and expression transfer. Studying the results of the embedding algorithm provides valuable insights into the structure of the StyleGAN latent space. We propose a set of experiments to test what class of images can be embedded, how they are embedded, what latent space is suitable for embedding, and if the embedding is semantically meaningful.*

## 1. Introduction

Generative Adverserial Networks (GANs) are very successfully applied in various computer vision applications, *e.g.* texture synthesis [20, 37, 31], video generation [35, 34], image-to-image translation [11, 40, 1, 27] and object detection [21].

In the few past years, the quality of images synthesized by GANs has increased rapidly. Compared to the seminal DCGAN framework [28] in 2015, the current state-of-the-art GANs [14, 3, 15, 40, 41] can synthesize at a much higher resolution and produce significantly more realistic images. Among them, StyleGAN [15] makes use of an intermediate $W$ latent space that holds the promise of enabling some controlled image modifications. We believe that image modifications are a lot more exciting when it becomes possible to modify a given image rather than a randomly GAN generated one. This leads to the natural question if it is possible to embed a given photograph into the GAN latent space.

To tackle this question, we build an embedding algorithm that can map a given image $I$ in the latent space of StyleGAN pre-trained on the FFHQ dataset. One of our important insights is that the generalization ability of the pre-trained StyleGAN is significantly enhanced when using an extended latent space $W^+$ (See Sec. 3.3). As a consequence, somewhat surprisingly, our embedding algorithm is not only able to embed human face images, but also successfully embeds non-face images from different classes. Therefore, we continue our investigation by analyzing the quality of the embedding to see if the embedding is semantically meaningful. To this end, we propose to use three basic operations on vectors in the latent space: linear interpolation, crossover, and adding a vector and a scaled difference vector. These operations correspond to three semantic image processing applications: morphing, style transfer, and expression transfer. As a result, we gain more insight into the structure of the latent space and can solve the mystery why even instances of non-face images such as cars can be embedded.

Our contributions include:

- An efficient embedding algorithm which can map a given image into the extended latent space $W^+$ of a pre-trained StyleGAN.
- We study multiple questions providing insight into the structure of the StyleGAN latent space, e.g.: What type of images can be embedded? What type of faces can be embedded? What latent space can be used for the embedding?
- We propose to use three basic operations on vectors to study the quality of the embedding. As a result, we can better understand the latent space and how different classes of images are embedded. As a byproduct, we obtain excellent results on multiple face image editing applications including morphing, style transfer, and expression transfer.

## 2. Related Work

**High-quality GANs** Starting from the groundbreaking work by Goodfellow *et al.* [8] in 2014, the entire computer vision community has witnessed the fast-paced improvements on GANs in the past years. For image generation tasks, DCGAN [28] is the first milestone that lays down the foundation of GAN architectures as fully-convolutional neural networks. Since then, various efforts have been made

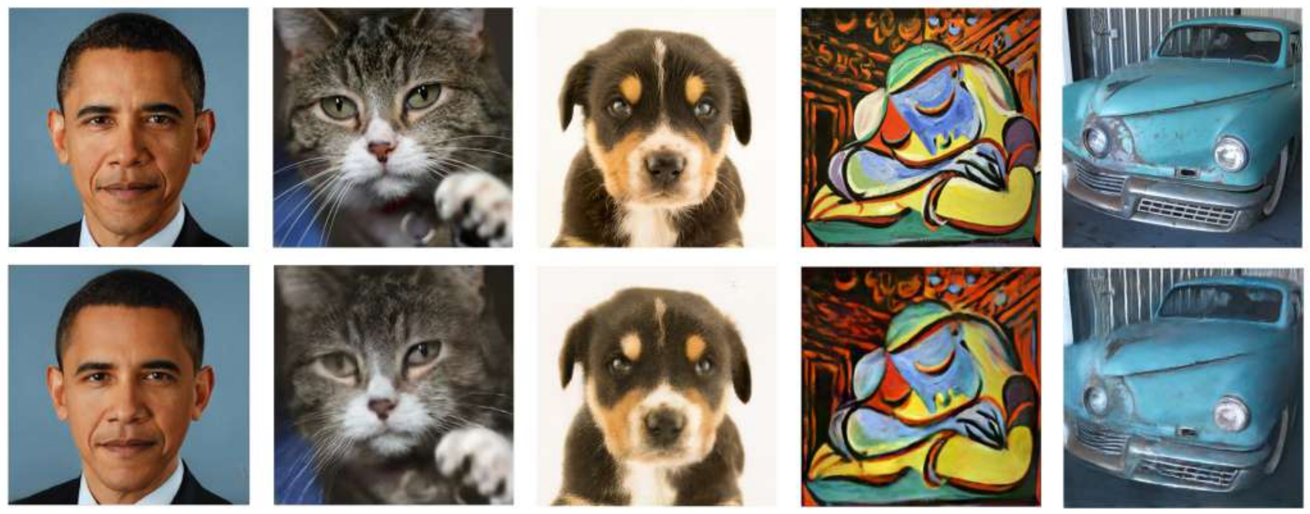


Figure 1: Top row: input images. Bottom row: results of embedding the images into the StyleGAN latent space.

to improve the performance of GANs from different aspects, *e.g*. the loss function [23, 2], the regularization or normalization [9, 25], and the architecture [9]. However, due to the limitation of computational power and the shortage of high-quality training data, these works are only tested with low resolution and poor quality datasets collected for classification / recognition tasks. Addressing this issue, Karras *et al*. collected the first high-quality human face dataset CelebA-HQ and proposed a progressive strategy to train GANs for high resolution image generation tasks [14]. Their ProGAN is the first GAN that can generate realistic human faces at a high resolution of $1024 \times 1024$. However, the generation of high-quality images from complex datasets (*e.g*. ImageNet) remains a challenge. To this end, Brock *et al*. proposed BigGAN and argued that the training of GANs benefit dramatically from large batch sizes [3]. Their BigGAN can generate realistic samples and smooth interpolations spanning different classes. Recently, Karras *et al*. collected a more diverse and higher quality human face dataset FFHQ and proposed a new generator architecture inspired by the idea of neural style transfer [10], which further improves the performance of GANs on human face generation tasks [15]. However, the lack of control over image modification ascribed to the interpretability of neural networks, is still an open problem. In this paper, we tackle the interpretability problem by embedding user-specified images back to the GAN latent space, which leads to a variety of potential applications.

**Latent Space Embedding** In general, there are two existing approaches to embed instances from the image space to the latent space: i) learn an encoder that maps a given image to the latent space (*e.g*. the Variational Auto-Encoder [16]); ii) select a random initial latent code and optimize it using gradient descent [39, 4]. Between them, the first approach provides a fast solution of image embedding by performing a forward pass through the encoder neural network. However, it usually has problems generalizing beyond the training dataset. In this paper, we decided to build on the second approach as the more general and stable solution. As a concurrently developed work, the Github repository stylegan-encoder [26] also demonstrated that the optimization-based approach leads to embeddings of very high visual quality.

**Perceptual Loss and Style Transfer** Traditionally, the low-level similarity between two images is measured in the pixel space with $L1/L2$ loss functions. While in the past years, inspired by the success of complex image classification [18, 22], Gatys *et al*. [7, 6] observed that the learned filters of the VGG image classification model [22] are excellent general-purpose feature extractors and proposed to use the covariance statistics of the extracted features to measure the high-level similarity between images *perceptually*, which is then formalized as the *perceptual loss* [12, 5]. To demonstrate the power of their method, they showed promising results on style transfer [6].

Specifically, they argued that different layers of the VGG neural network extract the image features at different scales and can be separated into *content* and *style*.

To accelerate the initial algorithm, Johnson *et al*. [12] proposed to train a neural network to solve the optimization problem of [6], which can transfer the style of a given image to any other image in real-time. The only limitation of their method is that they need to train separate neural networks for different style images. Finally, this issue is resolved by Huang and Belongie [10] with adaptive instance normalization. As a result, they can transfer arbitrary style in real-time.

## 3. What images can be embedded into the StyleGAN latent space?

We set out to study the question if it is even possible to embed images into the StyleGAN latent space. This question is not trivial, because our initial embedding experiments with faces and with other GANs resulted in faces that were no longer recognizable as the same person. Due to the improved variability of the FFHQ dataset and the superior quality of the StyleGAN architecture, there is a renewed hope that embedding existing images in the latent space is possible.

### 3.1. Embedding Results for Various Image Classes

To test our method, we collect a small-scale dataset of 25 diverse images spanning 5 categories (*i.e.* faces, cats, dogs, cars, and paintings). Details of the dataset are shown in the supplementary material. We use the code provided by StyleGAN [15] to preprocess the face images. This preprocess includes registration to a canonical face position.

To better understand the structure and attributes of the latent space, it is beneficial to study the embedding of a larger variety of image classes. We choose faces of cats, dogs, and paintings as they share the overall structure with human faces, but are depicted in a very different style. Cars are selected as they have no structural similarity to faces.

Figure 1 shows the embedding results consist of one example for each image class in the collected test dataset. It can be observed that the embedded Obama face is of very high perceptual quality and faithfully reproduces the input. However, it is noted that the embedded face is slightly smoothed and minor details are absent.

Going beyond faces, interestingly, we find that although the StyleGAN generator is trained on a human face dataset, the embedding algorithm is capable to go far beyond human faces. As Figure 1 shows, although slightly worse than those of human faces, we can obtain reasonable and relatively high-quality embeddings of cats, dogs and even paintings and cars. This reveals the effective embedding capability of the algorithm and the generality of the learned filters of the generator.

Another interesting question is how the quality of the pre-trained latent space affects the embedding. To conduct these tests we also used StyleGANs trained on cars, cats, ... The quality of these results is significantly lower, as shown in supplementary materials.

### 3.2. How Robust is the Embedding of Face Images?

**Affine Transformation** As Figure 2 and Table 1 show, the performance of StyleGAN embedding is very sensitive to affine transformations (translation, resizing and rotation). Among them, the translation seems to have the worst performance as it can fail to produce a valid face embedding. For resizing and rotation, the results are valid faces. However, they are blurry and lose many details, which are still worse than the normal embedding. From these observations, we argue that the generalization ability of GANs is sensitive to affine transformation, which implies that the learned representations are still scale and position dependent to some extent.

| Transformation | $L(\times 10^5)$ | $\|w^* - \bar{\mathbf{w}}\|$ |
|---|---|---|
| Translation (Right 140 pixels) | 0.782 | 48.56 |
| Translation (Left 160 pixels) | 0.406 | 44.12 |
| Zoom out (2X) | 0.225 | 38.04 |
| Zoom in (2X) | 0.718 | 40.55 |
| $90^\circ$ Rotation | 0.622 | 47.21 |
| $180^\circ$ Rotation | 0.599 | 42.93 |

Table 1: Embedding results of the transformed images. $L$ is the loss (Eq.1) after optimization. $\|w^* - \bar{\mathbf{w}}\|$ is the distance between the latent codes $w^*$ and $\bar{\mathbf{w}}$ (Section 5.1) of the average face [15].

**Embedding Defective Images** As Figure 3 shows, the performance of StyleGAN embedding is quite robust to defects in images. It can be observed that the embeddings of different facial features are independent of each other. For example, removing the nose does not have an obvious influence on the embedding of the eyes and the mouth. On the one hand, this phenomenon is good for general image editing applications. On the other hand, it shows that the latent space does not force the embedded image to be a complete face, i.e. it does not inpaint the missing information.

### 3.3. Which Latent Space to Choose?

There are multiple latent spaces in StyleGAN [15] that could be used for an embedding. Two obvious candidates are the initial latent space $Z$ and the intermediate latent space $W$. The 512-dimensional vectors $w \in W$ are obtained from the 512-dimensional vectors $z \in Z$ by passing them through a fully connected neural network. An important insight of our work is that it is not easily possible to embed into $W$ or $Z$ directly. Therefore, we propose to embed into an extended latent space $W^+$. $W^+$ is a concatenation of 18 different 512-dimensional $w$ vectors, one for each layer of the StyleGAN architecture that can receive input via AdaIn. As shown in Figure 5 (c)(d), embedding into $W$ directly does not give reasonable results. Another interesting question is how important the learned network weights are for the result. We answer this question in Figure 5 (b)(e) by showing an embedding into a network that is

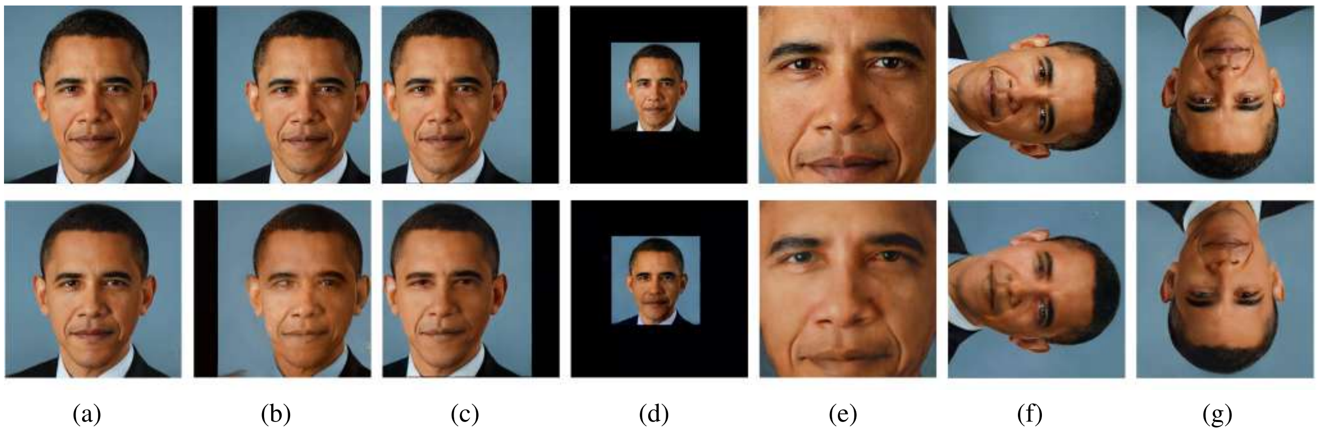


Figure 2: Top row: the input images. Bottom row: the embedded results. (a) Standard embedding results. (b) Translation 140 pixels to the right. (c) Translation 160 pixels to the left. (d) Zoom out by $2X$. (e) Zoom in by $2X$. (f) 90° rotation. (g) 180° rotation.

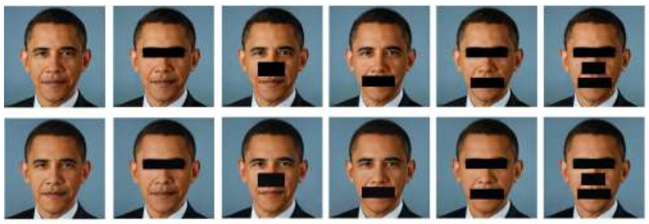


Figure 3: Stress test results on defective image embedding. Top row: the input images. Bottom row: the embedded results.

simply initialized with random weights.

## 4. How Meaningful is the Embedding?

We propose three tests to evaluate if an embedding is semantically meaningful. Each of these tests can be conducted by simple latent code manipulations of vectors $w_i$ and these tests correspond to semantic image editing applications in computer vision and computer graphics: morphing, expression transfer, and style transfer. We consider a test successful if the resulting manipulation results in high quality images.

### 4.1. Morphing

Image morphing is a longstanding research topic in computer graphics and computer vision, e.g. [36, 29, 30, 32, 38, 17]). Given two embedded images with their respective latent vectors $w_1$ and $w_2$, morphing is computed by a linear interpolation, $w = \lambda w_1 + (1 - \lambda)w_2, \lambda \in (0, 1)$, and subsequent image generation using the new code $w$. As Figure 4 shows, our method generates high-quality morphing between face images (row 1,2,3) but fails on non-face images in both in-class (row 4) and inter-class (row 5) morphing.

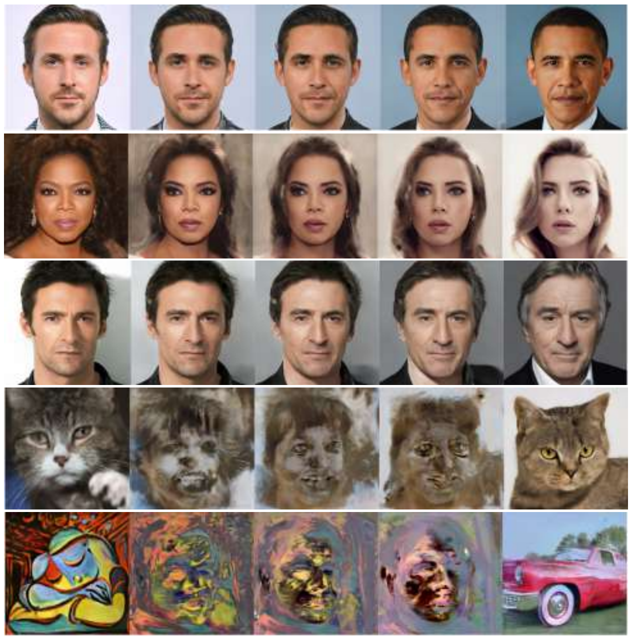


Figure 4: Morphing between two embedded images (the left-most and right-most ones).

Interestingly, it can be observed that there are contours of human faces in the intermediate images of the inter-class morphing, which shows that the latent space structure of this StyleGAN is dedicated to human faces. We therefore conjecture that non-face images are actually embedded the following way. The initial layers create a face like structure but the later layers paint over this structure so that it is no longer recognizable. While an extensive study of morphing itself is beyond the scope of this paper, we believe that the face morphing results are excellent and might be superior

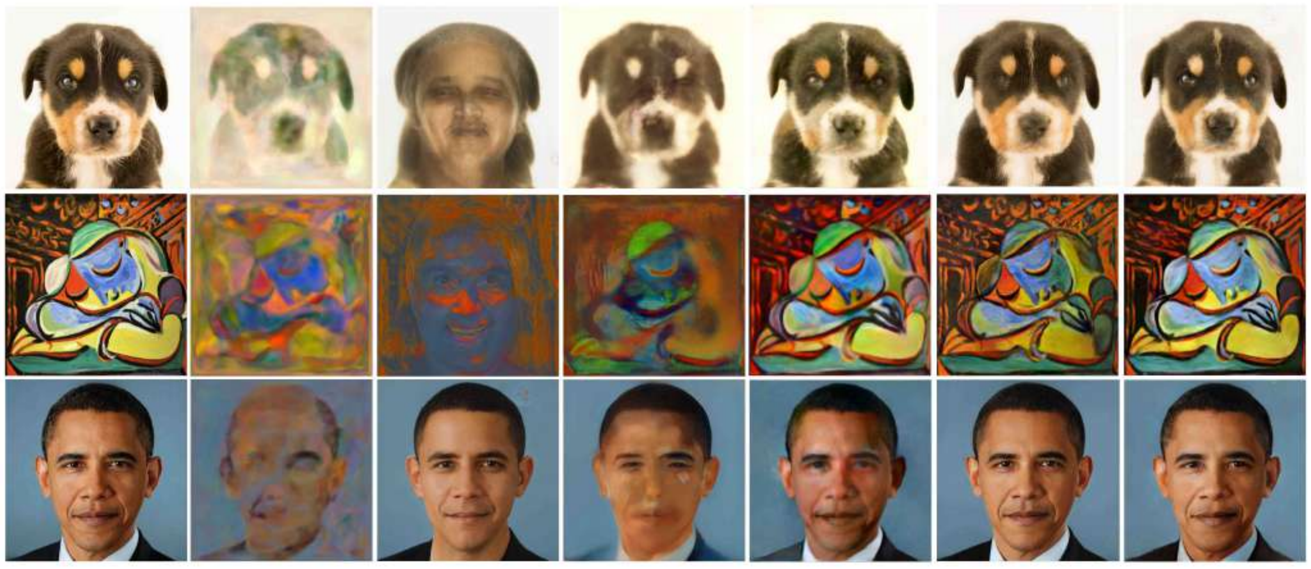


Figure 5: (a) Original images. Embedding results into the original space $W$: (b) using random weights in the network layers; (c) with $\bar{\mathbf{w}}$ initialization; (d) with random initialization. Embedding results into the $W+$ space: (e) using random weights in the network layers; (f) with $\bar{\mathbf{w}}$ initialization; (g) with random initialization.

to the current state of the art. We leave this investigation to future work.

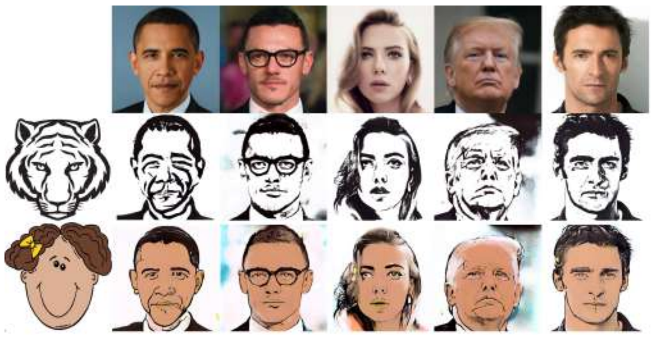


Figure 6: First column: style image; Second column: embedded stylized image using style loss from $conv4\_2$ layer of VGG-16; Third to Sixth column: style transfer by replacing latent code of last 9 layers of base image with the embedded style image.

### 4.2. Style Transfer

Given two latent codes $w_1$ and $w_2$, style transfer is computed by a crossover operation [15]. We show the style transfer results between an embedded stylized image and other face images (Figure 6) and between embedded images from different classes (Figure 8).

More specifically in Figure 8, we retain the latent codes of the embedded content image for the first 9 layers (corresponding to spatial resolution $4^2 - 64^2$) and override the latent codes with the ones of the style image for the last 9 layers (corresponding to spatial resolution $64^2 - 1024^2$). Our method is able to transfer the low level features (*e.g.* colors and textures) but fails to faithfully maintain the content structure of non-face images (second column Figure 8), especially the painting. This phenomenon reveals that the generalization and expressing power of StyleGAN is more likely to reside in the style layers corresponding to higher spatial resolutions.

### 4.3. Expression Transfer and Face Reenactment

Given three input vectors $w_1, w_2, w_3$, expression transfer is computed as $w = w_1 + \lambda(w_3 - w_2)$, where $w_1$ is the latent code of the target image, $w_2$ corresponds to a neutral expression of the source image, and $w_3$ corresponds to a more distinct expression. For example, $w_3$ could correspond to a smiling face and $w_2$ to an expressionless face of the same person. To eliminate the noise (*e.g.* background noise), we heuristically set a lower bound threshold on the $L2 - norm$ of the channels of difference latent code, below which, the channel is replaced by a zero vector. For the above experiment, the selected value of the threshold is 1. We normalize the resultant vectors to control the intensity of an expression in a particular direction. Such code is relatively independent of the source faces and can be used to transfer expressions (Figure 7). We believe that these

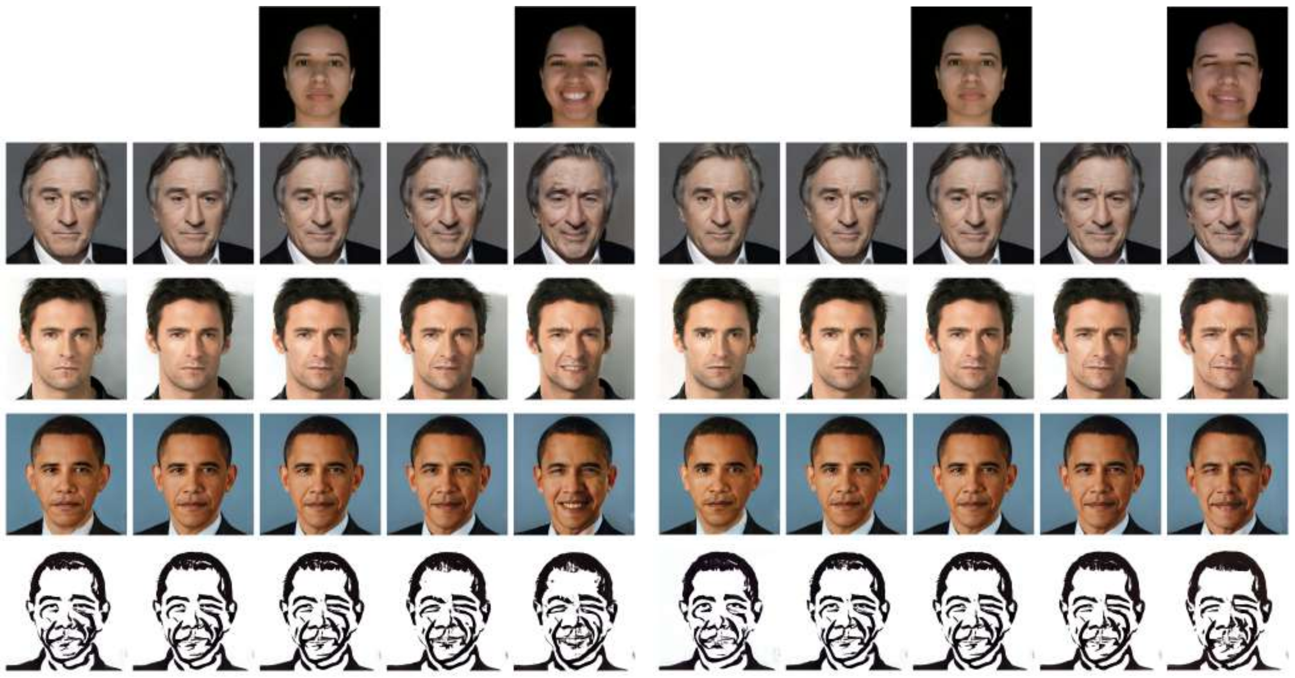


Figure 7: Results on expression transfer. The first row shows the reference images from IMPA-FACES3D [24] dataset. In the following rows, the middle image in each of the examples is the embedded image, whose expression is gradually transferred to the reference expression (on the right) and the opposite direction (on the left) respectively. More results are included in the supplementary material.

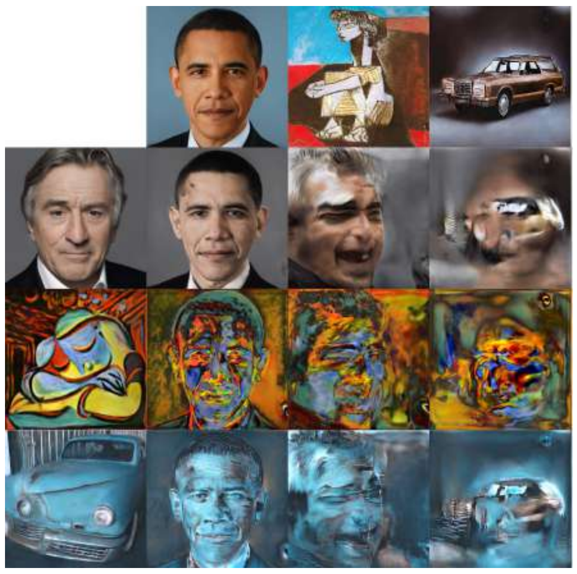


Figure 8: Style transfer between the embedded style image (first column) and the embedded content images (first row).

expression transfer results are also of very high quality. Additional results are available in supplementary materials and the accompanying video.

## 5. Embedding Algorithm

Our method follows a straightforward optimization framework [4] to embed a given image onto the manifold of the pre-trained generator. Starting from a suitable initialization $w$, we search for an optimized vector $w^*$ that minimizes the loss function that measures the similarity between the given image and the image generated from $w^*$. Algorithm 1 shows the pseudo-code of our method. An interesting aspect of this work is that not all design choices lead to good results and that experimenting with the design choices provides further insights into the embedding.

**Algorithm 1:** Latent Space Embedding for GANs

```
Input: An image $I \in \mathbb{R}^{n \times m \times 3}$ to embed; a
       pre-trained generator $G(\cdot)$.
Output: The embedded latent code $w^*$ and the
        embedded image $G(w^*)$ optimzed via $F'$.
1 Initialize latent code $w^* = w$;
2 while not converged do
3     $L \leftarrow L_{percept}(G(w^*), I) + \frac{\lambda}{N}\|G(w^*) - I\|_2^2$ ;
4     $w^* \leftarrow w^* - \eta F'(\nabla_{w^*} L)$;
5 end
```

| Data class | $w$ Init. | $L(\times 10^5)$ | $\|w^* - \bar{\mathbf{w}}\|$ |
|---|---|---|---|
| Face | $w = \bar{\mathbf{w}}$ | **0.309** | **30.67** |
| | Random | 0.351 | 35.60 |
| Cat | $w = \bar{\mathbf{w}}$ | 0.752 | 70.86 |
| | Random | **0.740** | 70.97 |
| Dog | $w = \bar{\mathbf{w}}$ | 0.922 | 74.78 |
| | Random | **0.845** | 75.14 |
| Painting | $w = \bar{\mathbf{w}}$ | 3.530 | 103.61 |
| | Random | **3.451** | 105.29 |
| Car | $w = \bar{\mathbf{w}}$ | 1.390 | 82.53 |
| | Random | **1.269** | 82.60 |

Table 2: Algorithmic choice justification on the latent code initialization. $w$ Init. is the initialization method for the latent code $w$. $L$ is the mean of the loss (Eq.1) after optimization. $\|w^* - \bar{\mathbf{w}}\|$ is the distance between the latent codes $w^*$ and $\bar{\mathbf{w}}$ of the average face [15].

## 5.1. Initialization

We investigate two design choices for the initialization. The first choice is random initialization. In this case, each variable is sampled independently from a uniform distribution $\mathcal{U}[-1, 1]$. The second choice is motivated by the observation that the distance to the mean latent vector $\bar{\mathbf{w}}$ can be used to identify low quality faces [15]. Therefore, we propose to use $\bar{\mathbf{w}}$ as initialization and expect the optimization to converge to a vector $w^*$ that is closer to $\bar{\mathbf{w}}$.

To evaluate these two design choices, we compared the loss values and the distance $\|w^* - \bar{\mathbf{w}}\|$ between the optimized latent code $w^*$ and $\bar{\mathbf{w}}$ after optimization. As Table 2 shows, initializing $w = \bar{\mathbf{w}}$ for face image embeddings not only makes the optimized $w^*$ closer to $\bar{\mathbf{w}}$, but also achieves a much lower loss value. However, for images in other classes (*e.g.* dog), random initialization proves to be the better option. Intuitively, the phenomenon suggests that the distribution has only one cluster of faces, the other instances (*e.g.* dogs, cats) are scattered points surrounding the cluster without obvious patterns. Qualitative results are shown in Figure 5 (f)(g).

## 5.2. Loss Function

To measure the similarity between the input image and the embedded image during optimization, we employ a loss function that is a weighted combination of the VGG-16 perceptual loss [12] and the pixel-wise MSE loss:

$$w^* = \min_{w} L_{percept}(G(w), I) + \frac{\lambda_{mse}}{N}\|G(w) - I\|_2^2 \quad (1)$$

where $I \in \mathbb{R}^{n\times n\times 3}$ is the input image, $G(\cdot)$ is the pre-trained generator, $N$ is the number of scalars in the image (*i.e.* $N = n \times n \times 3$), $w$ is the latent code to optimize,

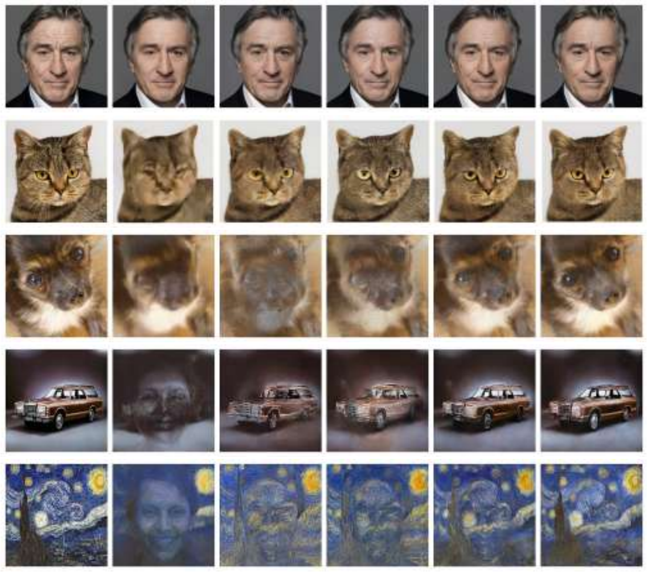


Figure 9: Algorithmic choice justification on the loss function. Each row shows the results of an image from the five different classes in our test dataset respectively. From left to right, each column shows: (1) the original image; (2) pixel-wise MSE loss only; (3) perceptual loss on VGG-16 $conv3\_2$ layer only; (4) pixel-wise MSE loss and VGG-16 $conv3\_2$; (5) perceptual loss (Eq.2) only; (6) our loss function (Eq.1). More results are included in the supplementary material.

$\lambda_{mse} = 1$ is empirically obtained for good performance. For the perceptual loss term $L_{percept}(\cdot)$ in Eq.1, we use:

$$L_{percept}(I_1, I_2) = \sum_{j=1}^{4} \frac{\lambda_j}{N_j}\|F_j(I_1) - F_j(I_2)\|_2^2 \quad (2)$$

where $I_1, I_2 \in \mathbb{R}^{n\times n\times 3}$ are the input images, $F_j$ is the feature output of VGG-16 layers $conv1\_1$, $conv1\_2$, $conv3\_2$ and $conv4\_2$ respectively, $N_j$ is the number of scalars in the $jth$ layer output, $\lambda_j = 1$ for all $j$s are empirically obtained for good performance.

Our choice of the perceptual loss together with the pixel-wise MSE loss comes from the fact that the pixel-wise MSE loss alone cannot find a high quality embedding. The perceptual loss therefore acts as some sort of regularizer to guide the optimization into the right region of the latent space.

We perform an ablation study to justify our choice of loss function in Eq.1. As Figure 9 shows, using the pixel-wise MSE loss term alone (column 2) embeds the general colors well but fails to catch the features of non-face images. In addition, it has a smoothing effect that does not preserve the details even for the human faces. Interestingly, due to the pixel-wise MSE loss working in the pixel space and ignor-

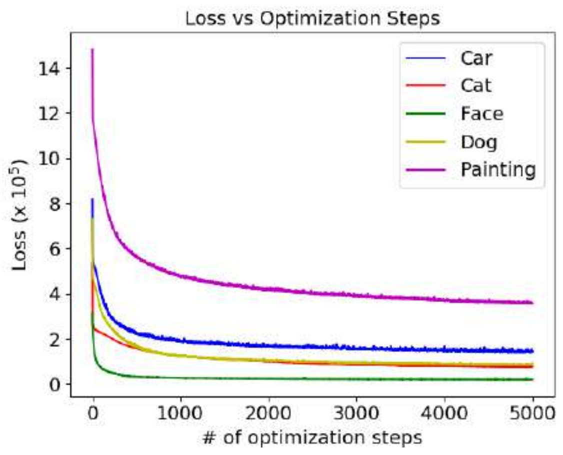


Figure 10: Loss values *vs*. the number of optimization steps.

ing the differences in feature space, its embedding results on non-face images (*e.g*. the car and the painting) have a tendency towards the average face of the pre-trained StyleGAN [15]. This problem is addressed by the perceptual losses (column 3, 5) that measures image similarity in the feature space. Since our embedding task requires the embedded image to be close to the input at all scales, we found that matching the features at multiple layers of the VGG-16 network (column 5) works better than using only a single layer (column 3). This further motivates us to combine the pixel-wise MSE loss with the perceptual loss (column 4, 6) from that the pixel-wise MSE loss can be viewed as the lowest level perceptual loss at pixel scale. Column 6 of Figure 9 shows the embedding results of our final choice (pixel-wise MSE + multi-layer perceptual loss), which achieves the best performance among different algorithmic choices.

## 5.3. Other Parameters

We use the Adam optimizer with a learning rate of $0.01$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1e^{-8}$ in all our experiments. We use 5000 gradient descent steps for the optimization, taking less than 7 minutes per image on a 32GB Nvidia TITAN V100 GPU.

To justify our choice of 5000 optimization steps, we investigated the change in the loss function as a function of the number of iterations. As Figure 10 shows, the loss value of the human face image drops the quickest and converges at around $1000$ optimization steps; those of the cat, the dog and the car images converge slower at around $3000$ optimization steps; while the painting curve is the slowest and converges around $5000$ optimization steps. We choose to optimize the loss function for $5000$ steps in all our experiments.

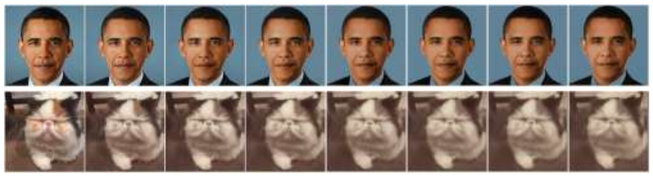


Figure 11: Stress test results on iterative embedding. The left most column shows the original images and the subsequent columns are the results of iterative embedding.

**Iterative Embedding** We tested the robustness of the proposed method on iterative embedding, *i.e*. we iteratively take the embedding results as new input images and do the embedding again. This process is repeated seven times. As Figure 11 shows, although it is guaranteed that the input image exists in the model distribution after the first embedding, the performance of the proposed method slowly degenerates (more details are lost) with the number of iterative embedding. The reason for this observation may be that the employed optimization approach suffers from slow convergence around local optimum. For the embeddings other than human faces, the stochastic initial latent codes may also be a factor for the degradation. In summary, these observations show that our embedding approach can reach reasonably "good" embeddings on the model distribution easily, although "perfect" embeddings are hard to reach.

## 6. Conclusion

We proposed an efficient algorithm to embed a given image into the latent space of StyleGAN. This algorithm enables semantic image editing operations, such as image morphing, style transfer, and expression transfer. We also used the algorithm to study multiple aspects of the StyleGAN latent space. We proposed experiments to analyze what type of images can be embedded, how they are embedded, and how meaningful the embedding is. Important conclusions of our work are that embedding works best into the extended latent space $W+$ and that any type of image can be embedded. However, only the embedding of faces is semantically meaningful.

Our framework still has several limitations. First, we inherit image artifacts present in pre-trained StyleGAN that we illustrate in supplementary materials. Second, the optimization takes several minutes and an embedding algorithm that can work in under a second would be more appealing for interactive editing.

In future work, we hope to extend our framework to process videos in addition to static images. Further, we would like to explore embeddings into GANs trained on three-dimensional data, such as point clouds or meshes.

**Acknowledgement** This work was supported by the KAUST Office of Sponsored Research (OSR) under Award

No. OSR-CRG2017-3426.

## References

[1] Yazeed Alharbi, Neil Smith, and Peter Wonka. Latent filter scaling for multimodal unsupervised image-to-image translation. *arXiv preprint arXiv:1812.09877*, 2018. 1

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

[3] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2019. 1, 2

[4] A. Creswell and A. A. Bharath. Inverting the generator of a generative adversarial network. *IEEE Transactions on Neural Networks and Learning Systems*, 2018. 2, 6

[5] Alexey Dosovitskiy and Thomas Brox. Generating images with perceptual similarity metrics based on deep networks. In *Advances in neural information processing systems*, pages 658–666, 2016. 2

[6] L. A. Gatys, A. S. Ecker, and M. Bethge. A neural algorithm of artistic style. *arXiv*, Aug 2015. 2

[7] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Texture synthesis using convolutional neural networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, 2015. 2

[8] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, 2014. 1

[9] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5767–5777, 2017. 2

[10] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, 2017. 2

[11] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *CVPR*, 2017. 1

[12] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, 2016. 2, 7

[13] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016. 11

[14] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018. 1, 2

[15] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *arXiv preprint arXiv:1812.04948*, 2018. 1, 2, 3, 5, 7, 8, 11

[16] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 2

[17] Pavel Korshunov and Touradj Ebrahimi. Using face morphing to protect privacy. In *2013 10th IEEE International Conference on Advanced Video and Signal Based Surveillance*, pages 208–213. IEEE, 2013. 4

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*. 2012. 2

[19] Samuli Laine. Feature-based metrics for exploring the latent space of generative models, 2018. 11

[20] Chuan Li and Michael 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. 1

[21] Jianan Li, Xiaodan Liang, Yunchao Wei, Tingfa Xu, Jiashi Feng, and Shuicheng Yan. Perceptual generative adversarial networks for small object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 1

[22] S. Liu and W. Deng. Very deep convolutional neural network based image classification using small training sample size. In *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, Nov 2015. 2

[23] Xudong Mao, Qing Li, Haoran Xie, Raymond Y.K. Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 2

[24] Jesús P Mena-Chalco, Luiz Velho, and RM Cesar Junior. 3d human face reconstruction using principal components spaces. In *Proceedings of WTD SIBGRAPI Conference on Graphics, Patterns and Images*, 2011. 6, 23

[25] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018. 2

[26] Dmitry Nikitko. stylegan-encoder. `https://github.com/Puzer/stylegan-encoder`, 2019. 2

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

[28] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015. 1

[29] Ulrich Scherhag, Christian Rathgeb, Johannes Merkle, Ralph Breithaupt, and Christoph Busch. Face recognition systems under morphing attacks: A survey. *IEEE Access*, 7, 2019. 4

[30] Clemens Seibold, Wojciech Samek, Anna Hilsmann, and Peter Eisert. Detection of face morphing attacks by deep learning. In *International Workshop on Digital Watermarking*, pages 107–120. Springer, 2017. 4

[31] Ron Slossberg, Gil Shamai, and Ron Kimmel. High quality facial surface and texture synthesis via generative adversarial networks. In *European Conference on Computer Vision*, pages 498–513. Springer, 2018. 1

[32] Mark Steyvers. Morphing techniques for manipulating face images. *Behavior Research Methods, Instruments, & Computers*, 31(2):359–369, 1999. 4

[33] Timo Aila Tero Karras, Samuli Laine. Stylegan - official tensorflow implementation. `https://github.com/NVlabs/stylegan`, 2018. 11

[34] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. Mocogan: Decomposing motion and content for video generation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 1

[35] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. In *Advances in Neural Information Processing Systems 29*. 2016. 1

[36] George Wolberg. Image morphing: a survey. *The Visual Computer*, 14(8), 1998. 4

[37] Wenqi Xian, Patsorn Sangkloy, Varun Agrawal, Amit Raj, Jingwan Lu, Chen Fang, Fisher Yu, and James Hays. Texturegan: Controlling deep image synthesis with texture patches. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 1

[38] Fei Yang, Eli Shechtman, Jue Wang, Lubomir Bourdev, and Dimitris Metaxas. Face morphing using 3d-aware appearance optimization. In *Proceedings of Graphics Interface 2012*, pages 93–99. Canadian Information Processing Society, 2012. 4

[39] Jun-Yan Zhu, Philipp Krhenbhl, Eli Shechtman, and Alexei A. Efros. Generative visual manipulation on the natural image manifold. *Lecture Notes in Computer Science*, page 597613, 2016. 2

[40] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networkss. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017. 1

[41] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 465–476. Curran Associates, Inc., 2017. 1

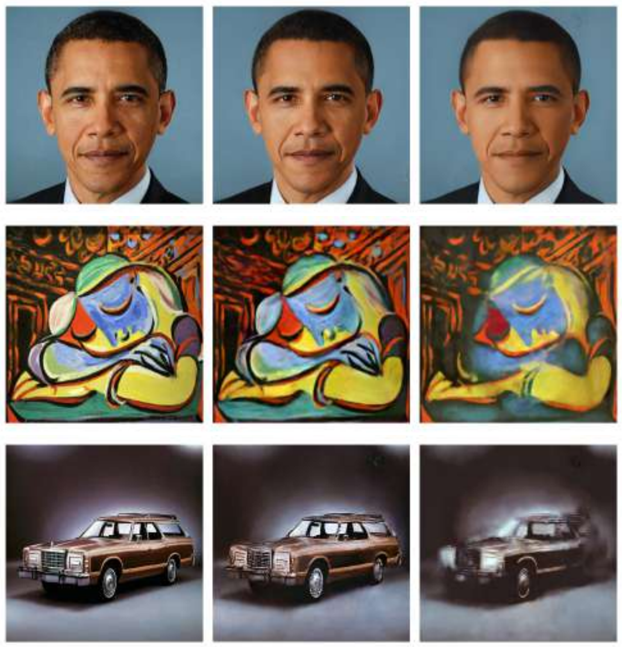


Figure 12: First column: original image ($1024 \times 1024$). Second column: embedded image with the perceptual loss applied to resized images of $256 \times 256$ resolution. Third column: embedded image with the perceptual loss applied to the images at the original $1024 \times 1024$ resolution.

## 7. Additional Materials on Embedding

**Dataset** In order to test our embedding algorithm, we collect a small dataset of 25 images in five different categories: human faces, cats, dogs, cars and paintings (Figure 17).

**Additional Embedding Results** To further support our findings about the initial latent code in the main paper, we show more results in Figure 13. It can be observed that: for face images, initializing the optimization with the mean face latent code works better; while for non-face images, using the latent codes randomly sampled from a multivariate uniform distribution is a better option.

**Quantitative Results on Defective Image Embedding** Table 3 shows the corresponding quantitative results on defective image embedding (Figure 3 in the main paper). The results show that compared to non-defective faces, the embedded images of defective faces are farther from the mean face. This reaffirms that the valid faces form a cluster around the mean face.

**Inherent Circular Artifacts of StyleGAN** Interestingly, we observed that the StyleGAN model trained on the FFHQ dataset (officially released [15, 33]) inherently creates circular artifacts in the generated images, which are also observable in our embedding results (Figure 21). These artifacts are thus independent of our embedding algorithm and may be resolved by employing better pretrained models in the future.

| Defect | $L(\times 10^5)$ | $\|w^* - \bar{\mathbf{w}}\|$ |
|---|---|---|
| non-defective | 0.204 | 29.19 |
| Eyes | 0.271 | 34.90 |
| Nose | 0.311 | 39.20 |
| Mouth | 0.301 | 37.04 |
| Eyes and Mouth | 0.233 | 39.62 |
| Eyes, Nose and Mouth | 0.285 | 37.59 |

Table 3: Quantitative results on defective image embedding (Figure 3 in the main paper). $L$ is the loss after optimization. $\|w^* - \bar{\mathbf{w}}\|$ is the distance between the latent codes $w^*$ and $\bar{\mathbf{w}}$ of the average face.

**Limitation of the ImageNet-based Perceptual loss** All existing perceptual losses utilize the classifiers trained on the ImageNet dataset (*e.g.* VGG-16, VGG-19), which are restricted to the resolution of $224 \times 224$. While in our paper, we aim to embed images of high resolution ($1024 \times 1024$) that are much larger than that of ImageNet images. Such inconsistency in the resolution may disable the learned image filters as they are scale-dependent. To this end, we follow the common practice [13, 19] and use a simple resizing trick to compute the perceptual loss on resized images of $256 \times 256$ resolution. As Figure 12 shows, the embedding results with the resizing trick outperform the ones at the original resolution. However, small details are lost during the resizing, which can slightly smoothen the embedding results. We expect to get better results with future perceptual losses that work on higher resolutions.

**StyleGANs trained on Other Datasets** To support our insights on the learned distribution, we further tested our embedding algorithm on the StyleGANs trained on three more datasets: the LSUN-Car ($512 \times 384$), LSUN-Cat ($256 \times 256$) and LSUN-Bedroom ($256 \times 256$) datasets. The embedding results are shown in Figure 18. It can be observed that the quality of the embedding is poor compared to that of the StyleGAN trained on the FFHQ dataset. The linear interpolation (image morphing) results of LSUN-Cat, LSUN-Car, and LSUN-Bedroom StyleGANs are shown in Figure 19 (a), (b) and (c) respectively. Interestingly, we observed that linear interpolation fails on the LSUN-Cat and LSUN-Car StyleGANs. Recall that the FFHQ human face dataset is of very high quality in terms of scale, alignment, color, poses *etc*., we believe that the low quality of

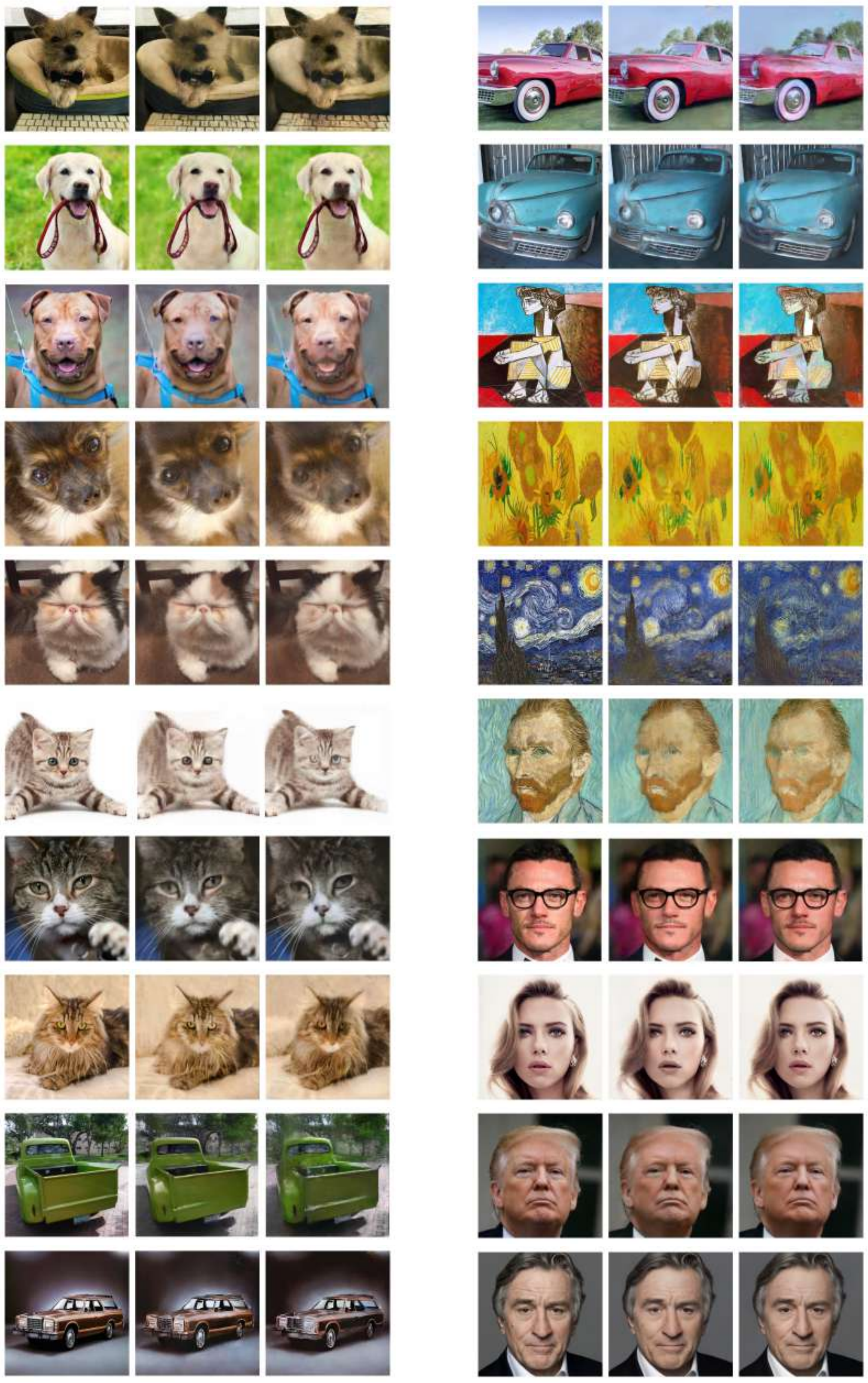


Figure 13: Additional Embedding Results into $W+$ space. Left column: the original images. Middle column: the embedded images with random latent code initialization. Right column: the embedded images with $\bar{w}$ latent code initialization.

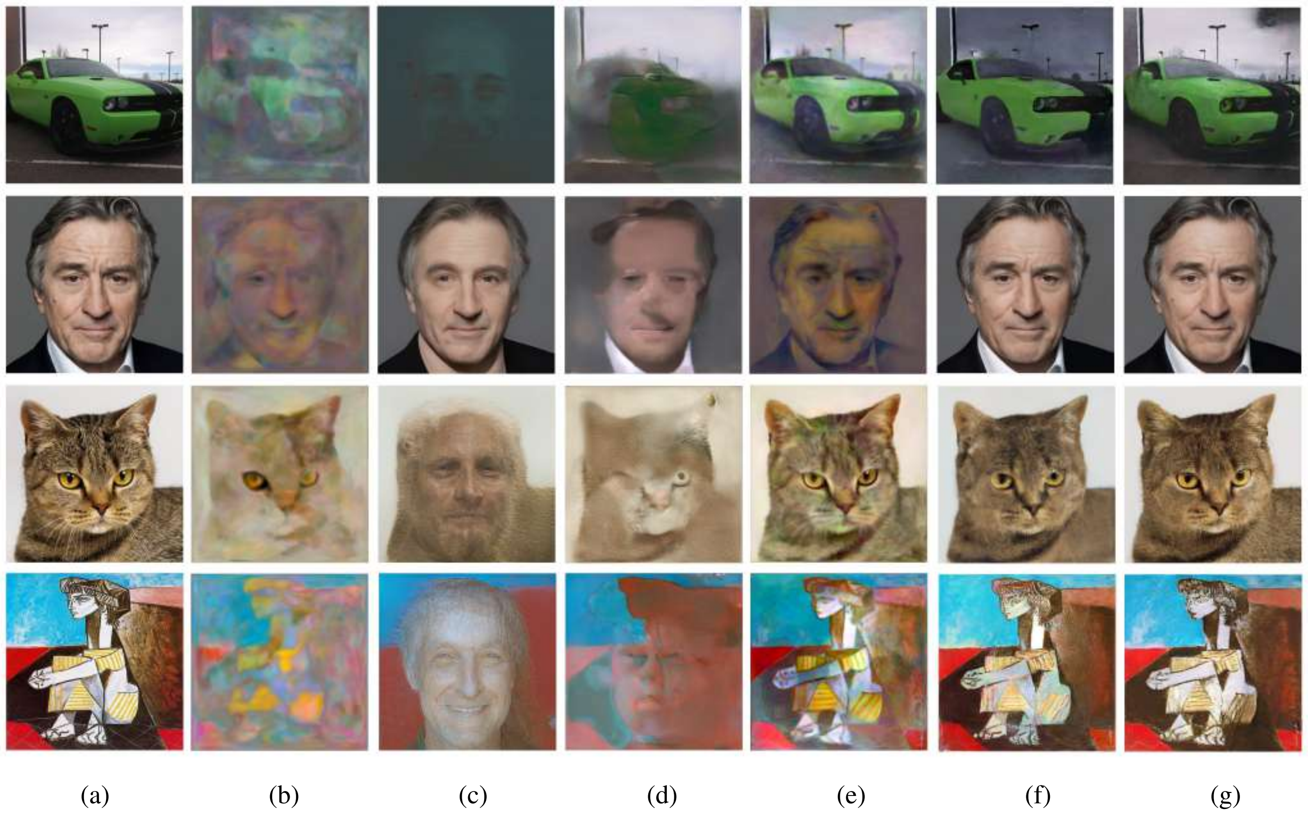


Figure 14: Additional results on the justification of latent space choice.(a) Original images. Embedding results into the original space $W$: (b) using random weights in the network layers; (c) with $\bar{\mathbf{w}}$ initialization; (d) with random initialization. Embedding results into the $W+$ space: (e) using random weights in the network layers; (f) with $\bar{\mathbf{w}}$ initialization; (g) with random initialization.

the LSUN datasets is the source of such failure. In other words, the quality of the data distribution is one of the key components to learn a meaningful model distribution.

**Additional Results on the Justification of Latent Space Choice** Figure 14 shows additional results (cat, dog, car) on the justification of our choice of latent space $W^+$. Similar to the main paper, we can observe that: (i) embedding into $W$ directly does not give reasonable results; (ii) the learned network weights is important to good embeddings.

**Clustering or Scattering?** To support our insight that only face images form a cluster in the latent space, we compute the $L2$ distances between the embeddings of all pairs of test images (Figure 20). It can be observed that the distances between the faces are relatively smaller than those of other classes, which justifies that they are close to each other in the $W^+$ space and form a cluster. For images in other classes, especially the paintings, the pairwise distances are much higher. This implies that they are scattered in the latent space.

**Justification of Loss Function Choice** Figure 22 validates the algorithmic choice of the loss function used in the main paper. It can be observed that (i) matching the image features at multiple layers of the VGG-16 network works better than at a single layer; (ii) the combination of pixel-wise MSE loss and perceptual loss works the best.

**Influence of Noise Channels** Figure 16 shows that restarting the embedding with a different noise leads to similar results. In addition, we observed significantly worse quality when resampling the noise during the embedding (at each update step). To this end, we kept the noise channel constant during the embedding for all our experiments.

## 8. Additional Results on Applications

Figure 15 shows additional results of the image morphing. Figure 23 shows the complete table of the style transfer results between different classes. The results support our insight that the multi-class embedding works by using an underlying human face structure (encoded in the first couple of layers) and painting powerful styles onto it (encoded in the

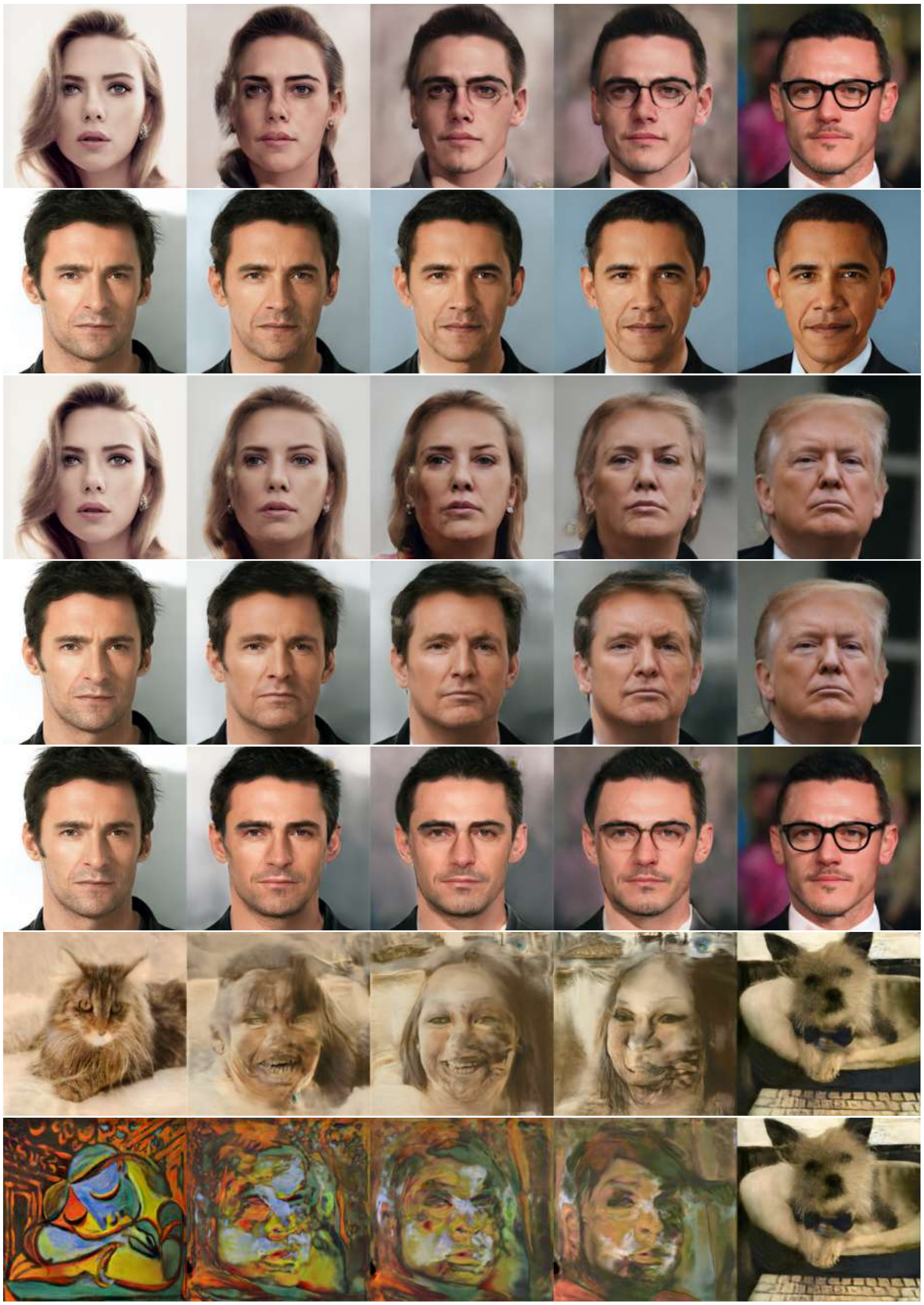


Figure 15: Additional morphing results between two embedded images (the left-most and right-most ones).

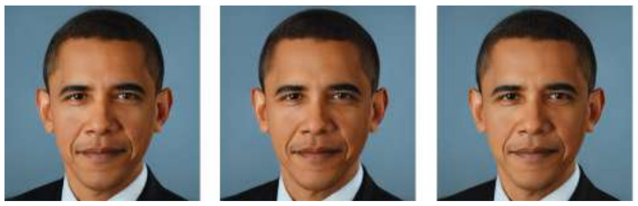


Figure 16: Image embedding using different constant noises.

latter layers). Figure 25 shows additional results on the expression transfer. We also include an accompanying video in the supplementary material to show it works with noisy images taken by a commodity camera in a typical office environment. The random walk results (of two classes 'human faces' and 'cars') from the embedded image towards the mean face image are also shown in videos.

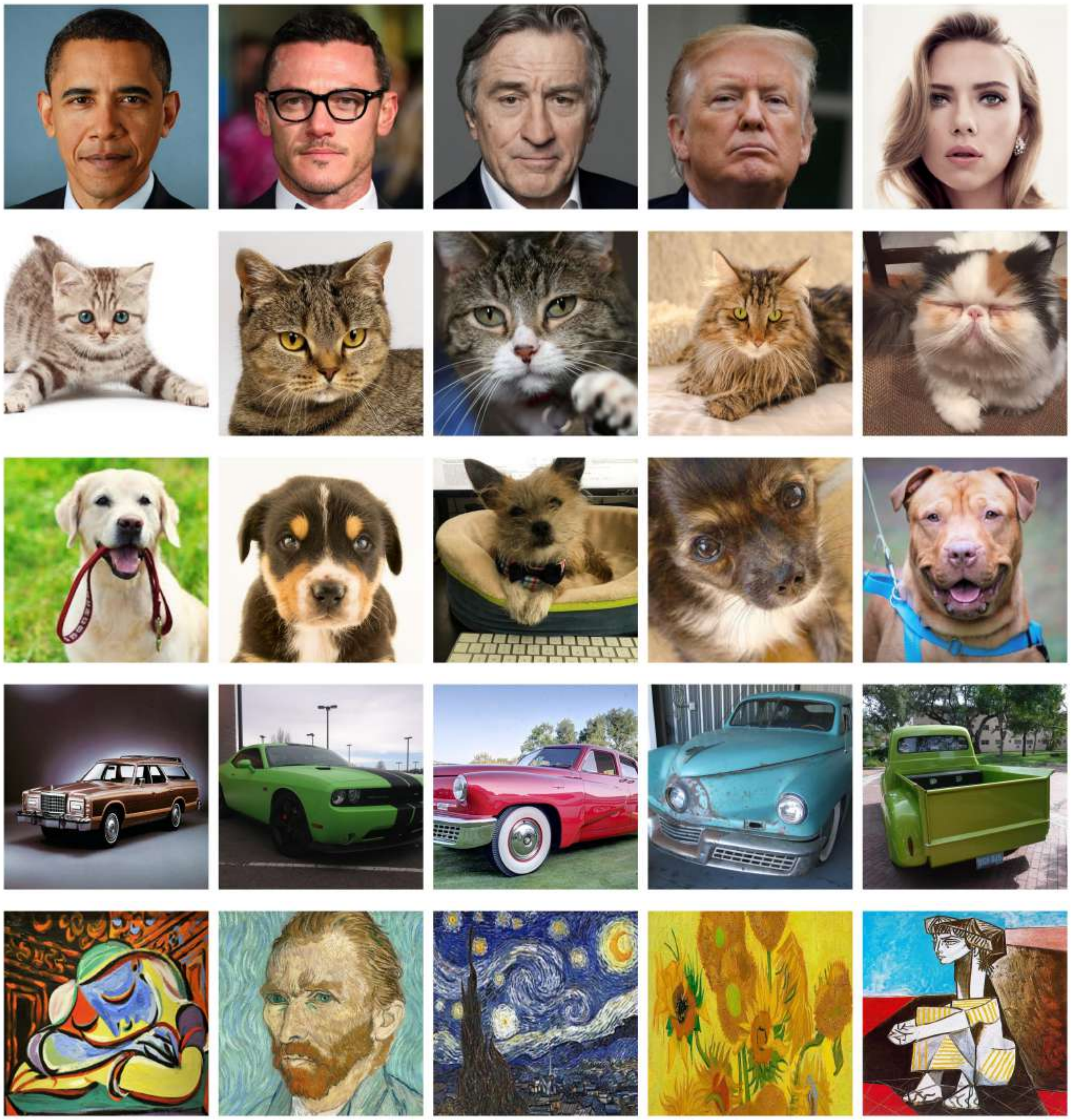
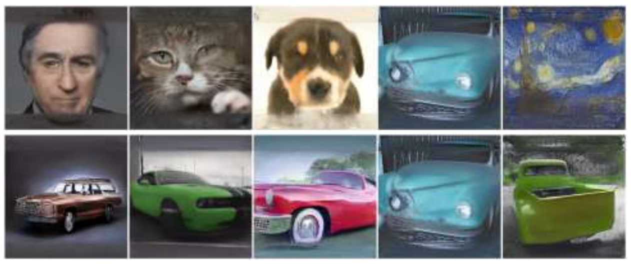
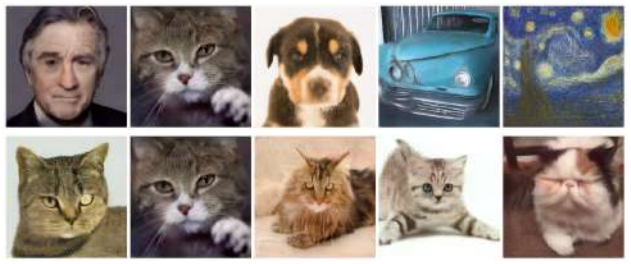


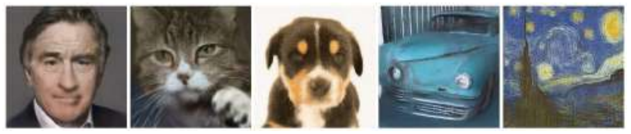
Figure 17: The collected 25 images of our dataset. First row: human faces. Second row: cats. Third row: dogs. Fourth row: cars. Fifth row: paintings.



(a)

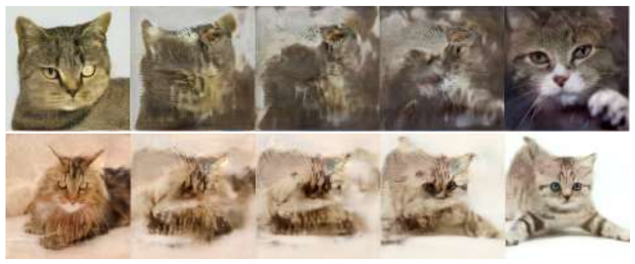


(b)

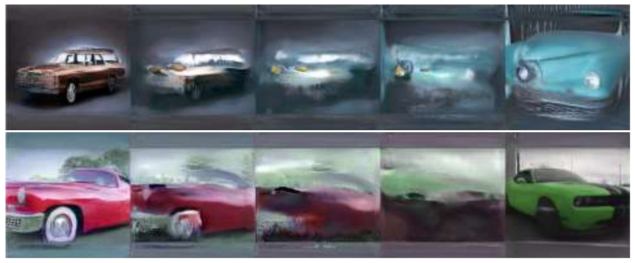


(c)

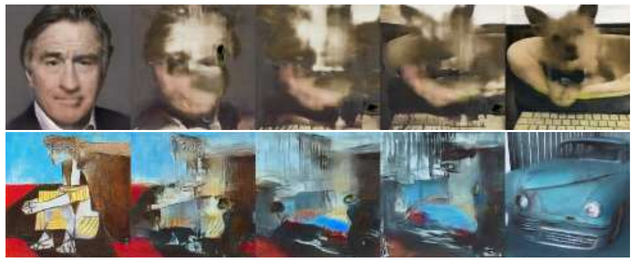
Figure 18: Embedding results of StyleGANs trained on (a) LSUN-Car, (b) LSUN-Cat and (c) LSUN-Bedroom datasets. For each subfigure, the first row shows the embedding results of the images in 5 different classes in our dataset. The second row shows the embedding results of the images of the corresponding class in our dataset ("cars" in (a) and "cats" in (b)). Note that (c) has only one row because we did not collect bedroom images in our dataset.



(a)

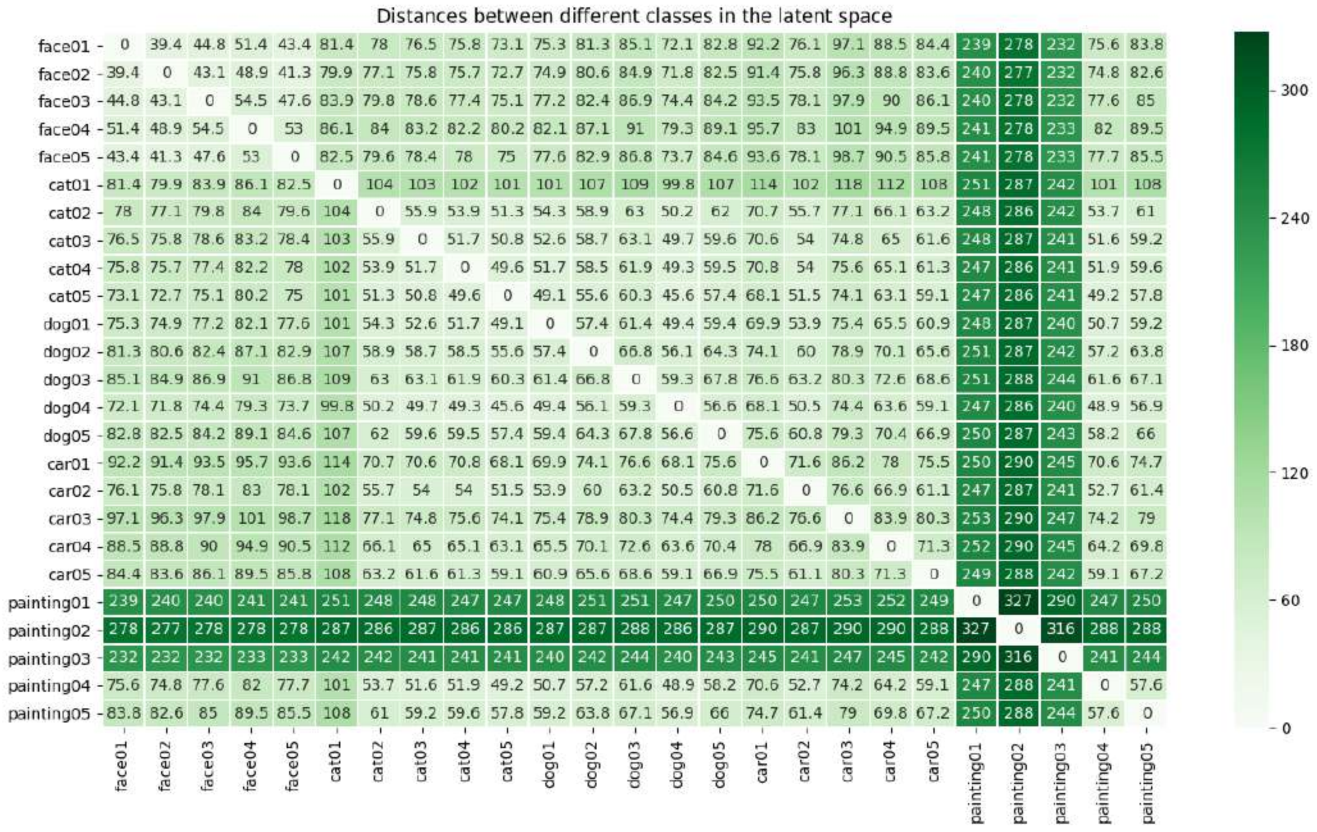


(b)



(c)

Figure 19: Results on linear interpolations (image morphing) in the latent spaces of StyleGANs trained on (a) LSUN-Cat (b) LSUN-Car (c) LSUN-Bedroom datasets.



| | face01 | face02 | face03 | face04 | face05 | cat01 | cat02 | cat03 | cat04 | cat05 | dog01 | dog02 | dog03 | dog04 | dog05 | car01 | car02 | car03 | car04 | car05 | painting01 | painting02 | painting03 | painting04 | painting05 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| face01 | 0 | 39.4 | 44.8 | 51.4 | 43.4 | 81.4 | 78 | 76.5 | 75.8 | 73.1 | 75.3 | 81.3 | 85.1 | 72.1 | 82.8 | 92.2 | 76.1 | 97.1 | 88.5 | 84.4 | 239 | 278 | 232 | 75.6 | 83.8 |
| face02 | 39.4 | 0 | 43.1 | 48.9 | 41.3 | 79.9 | 77.1 | 75.8 | 75.7 | 72.7 | 74.9 | 80.6 | 84.9 | 71.8 | 82.5 | 91.4 | 75.8 | 96.3 | 88.8 | 83.6 | 240 | 277 | 232 | 74.8 | 82.6 |
| face03 | 44.8 | 43.1 | 0 | 54.5 | 47.6 | 83.9 | 79.8 | 78.6 | 77.4 | 75.1 | 77.2 | 82.4 | 86.9 | 74.4 | 84.2 | 93.5 | 78.1 | 97.9 | 90 | 86.1 | 240 | 278 | 232 | 77.6 | 85 |
| face04 | 51.4 | 48.9 | 54.5 | 0 | 53 | 86.1 | 84 | 83.2 | 82.2 | 80.2 | 82.1 | 87.1 | 91 | 79.3 | 89.1 | 95.7 | 83 | 101 | 94.9 | 89.5 | 241 | 278 | 233 | 82 | 89.5 |
| face05 | 43.4 | 41.3 | 47.6 | 53 | 0 | 82.5 | 79.6 | 78.4 | 78 | 75 | 77.6 | 82.9 | 86.8 | 73.7 | 84.6 | 93.6 | 78.1 | 98.7 | 90.5 | 85.8 | 241 | 278 | 233 | 77.7 | 85.5 |
| cat01 | 81.4 | 79.9 | 83.9 | 86.1 | 82.5 | 0 | 104 | 103 | 102 | 101 | 101 | 107 | 109 | 99.8 | 107 | 114 | 102 | 118 | 112 | 108 | 251 | 287 | 242 | 101 | 108 |
| cat02 | 78 | 77.1 | 79.8 | 84 | 79.6 | 104 | 0 | 55.9 | 53.9 | 51.3 | 54.3 | 58.9 | 63 | 50.2 | 62 | 70.7 | 55.7 | 77.1 | 66.1 | 63.2 | 248 | 286 | 242 | 53.7 | 61 |
| cat03 | 76.5 | 75.8 | 78.6 | 83.2 | 78.4 | 103 | 55.9 | 0 | 51.7 | 50.8 | 52.6 | 58.7 | 63.1 | 49.7 | 59.6 | 70.6 | 54 | 74.8 | 65 | 61.6 | 248 | 287 | 241 | 51.6 | 59.2 |
| cat04 | 75.8 | 75.7 | 77.4 | 82.2 | 78 | 102 | 53.9 | 51.7 | 0 | 49.6 | 51.7 | 58.5 | 61.9 | 49.3 | 59.5 | 70.8 | 54 | 75.6 | 65.1 | 61.3 | 247 | 286 | 241 | 51.9 | 59.6 |
| cat05 | 73.1 | 72.7 | 75.1 | 80.2 | 75 | 101 | 51.3 | 50.8 | 49.6 | 0 | 49.1 | 55.6 | 60.3 | 45.6 | 57.4 | 68.1 | 51.5 | 74.1 | 63.1 | 59.1 | 247 | 286 | 241 | 49.2 | 57.8 |
| dog01 | 75.3 | 74.9 | 77.2 | 82.1 | 77.6 | 101 | 54.3 | 52.6 | 51.7 | 49.1 | 0 | 57.4 | 61.4 | 49.4 | 59.4 | 69.9 | 53.9 | 75.4 | 65.5 | 60.9 | 248 | 287 | 240 | 50.7 | 59.2 |
| dog02 | 81.3 | 80.6 | 82.4 | 87.1 | 82.9 | 107 | 58.9 | 58.7 | 58.5 | 55.6 | 57.4 | 0 | 66.8 | 56.1 | 64.3 | 74.1 | 60 | 78.9 | 70.1 | 65.6 | 251 | 287 | 242 | 57.2 | 63.8 |
| dog03 | 85.1 | 84.9 | 86.9 | 91 | 86.8 | 109 | 63 | 63.1 | 61.9 | 60.3 | 61.4 | 66.8 | 0 | 59.3 | 67.8 | 76.6 | 63.2 | 80.3 | 72.6 | 68.6 | 251 | 288 | 244 | 61.6 | 67.1 |
| dog04 | 72.1 | 71.8 | 74.4 | 79.3 | 73.7 | 99.8 | 50.2 | 49.7 | 49.3 | 45.6 | 49.4 | 56.1 | 59.3 | 0 | 56.6 | 68.1 | 50.5 | 74.4 | 63.6 | 59.1 | 247 | 286 | 240 | 48.9 | 56.9 |
| dog05 | 82.8 | 82.5 | 84.2 | 89.1 | 84.6 | 107 | 62 | 59.6 | 59.5 | 57.4 | 59.4 | 64.3 | 67.8 | 56.6 | 0 | 75.6 | 60.8 | 79.3 | 70.4 | 66.9 | 250 | 287 | 243 | 58.2 | 66 |
| car01 | 92.2 | 91.4 | 93.5 | 95.7 | 93.6 | 114 | 70.7 | 70.6 | 70.8 | 68.1 | 69.9 | 74.1 | 76.6 | 68.1 | 75.6 | 0 | 71.6 | 86.2 | 78 | 75.5 | 250 | 290 | 245 | 70.6 | 74.7 |
| car02 | 76.1 | 75.8 | 78.1 | 83 | 78.1 | 102 | 55.7 | 54 | 54 | 51.5 | 53.9 | 60 | 63.2 | 50.5 | 60.8 | 71.6 | 0 | 76.6 | 66.9 | 61.1 | 247 | 287 | 241 | 52.7 | 61.4 |
| car03 | 97.1 | 96.3 | 97.9 | 101 | 98.7 | 118 | 77.1 | 74.8 | 75.6 | 74.1 | 75.4 | 78.9 | 80.3 | 74.4 | 79.3 | 86.2 | 76.6 | 0 | 83.9 | 80.3 | 253 | 290 | 247 | 74.2 | 79 |
| car04 | 88.5 | 88.8 | 90 | 94.9 | 90.5 | 112 | 66.1 | 65 | 65.1 | 63.1 | 65.5 | 70.1 | 72.6 | 63.6 | 70.4 | 78 | 66.9 | 83.9 | 0 | 71.3 | 252 | 290 | 245 | 64.2 | 69.8 |
| car05 | 84.4 | 83.6 | 86.1 | 89.5 | 85.8 | 108 | 63.2 | 61.6 | 61.3 | 59.1 | 60.9 | 65.6 | 68.6 | 59.1 | 66.9 | 75.5 | 61.1 | 80.3 | 71.3 | 0 | 249 | 288 | 242 | 59.1 | 67.2 |
| painting01 | 239 | 240 | 240 | 241 | 241 | 251 | 248 | 248 | 247 | 247 | 248 | 251 | 251 | 247 | 250 | 250 | 247 | 253 | 252 | 249 | 0 | 327 | 290 | 247 | 250 |
| painting02 | 278 | 277 | 278 | 278 | 278 | 287 | 286 | 287 | 286 | 286 | 287 | 287 | 288 | 286 | 287 | 290 | 287 | 290 | 290 | 288 | 327 | 0 | 316 | 288 | 288 |
| painting03 | 232 | 232 | 232 | 233 | 233 | 242 | 242 | 241 | 241 | 241 | 240 | 242 | 244 | 240 | 243 | 245 | 241 | 247 | 245 | 242 | 290 | 316 | 0 | 241 | 244 |
| painting04 | 75.6 | 74.8 | 77.6 | 82 | 77.7 | 101 | 53.7 | 51.6 | 51.9 | 49.2 | 50.7 | 57.2 | 61.6 | 48.9 | 58.2 | 70.6 | 52.7 | 74.2 | 64.2 | 59.1 | 247 | 288 | 241 | 0 | 57.6 |
| painting05 | 83.8 | 82.6 | 85 | 89.5 | 85.5 | 108 | 61 | 59.2 | 59.6 | 57.8 | 59.2 | 63.8 | 67.1 | 56.9 | 66 | 74.7 | 61.4 | 79 | 69.8 | 67.2 | 250 | 288 | 244 | 57.6 | 0 |

Figure 20: Heat map of the inter- and intra-class $L2$ distances between embedded images.

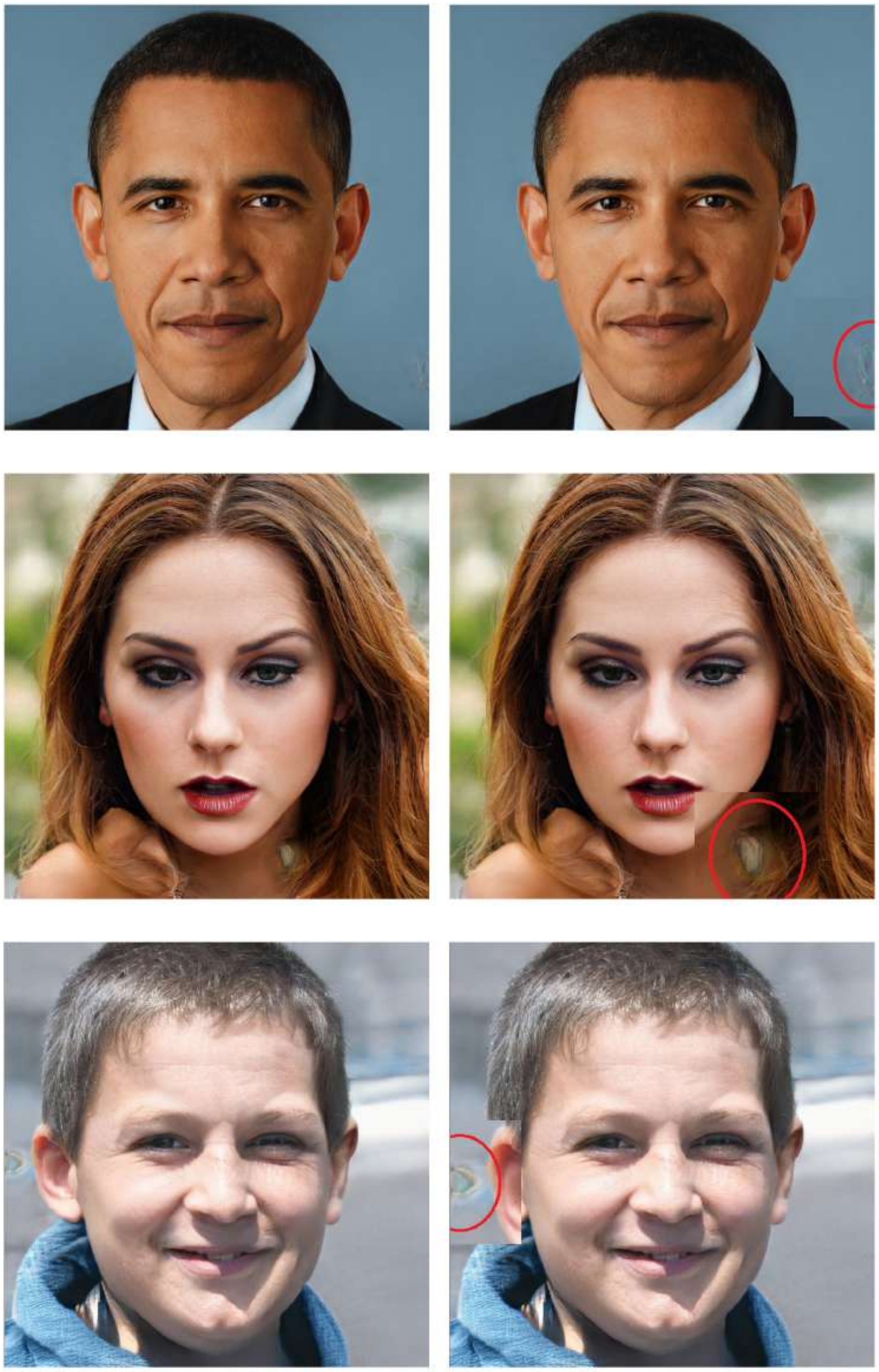


Figure 21: Inherent circular artifacts of StyleGAN. First row: circular artifacts in the embeded images. Second and third rows: randomly generated images. Left column: images with circular artifacts. Right column: highlighted artifacts by zooming in their local neighbourhood.

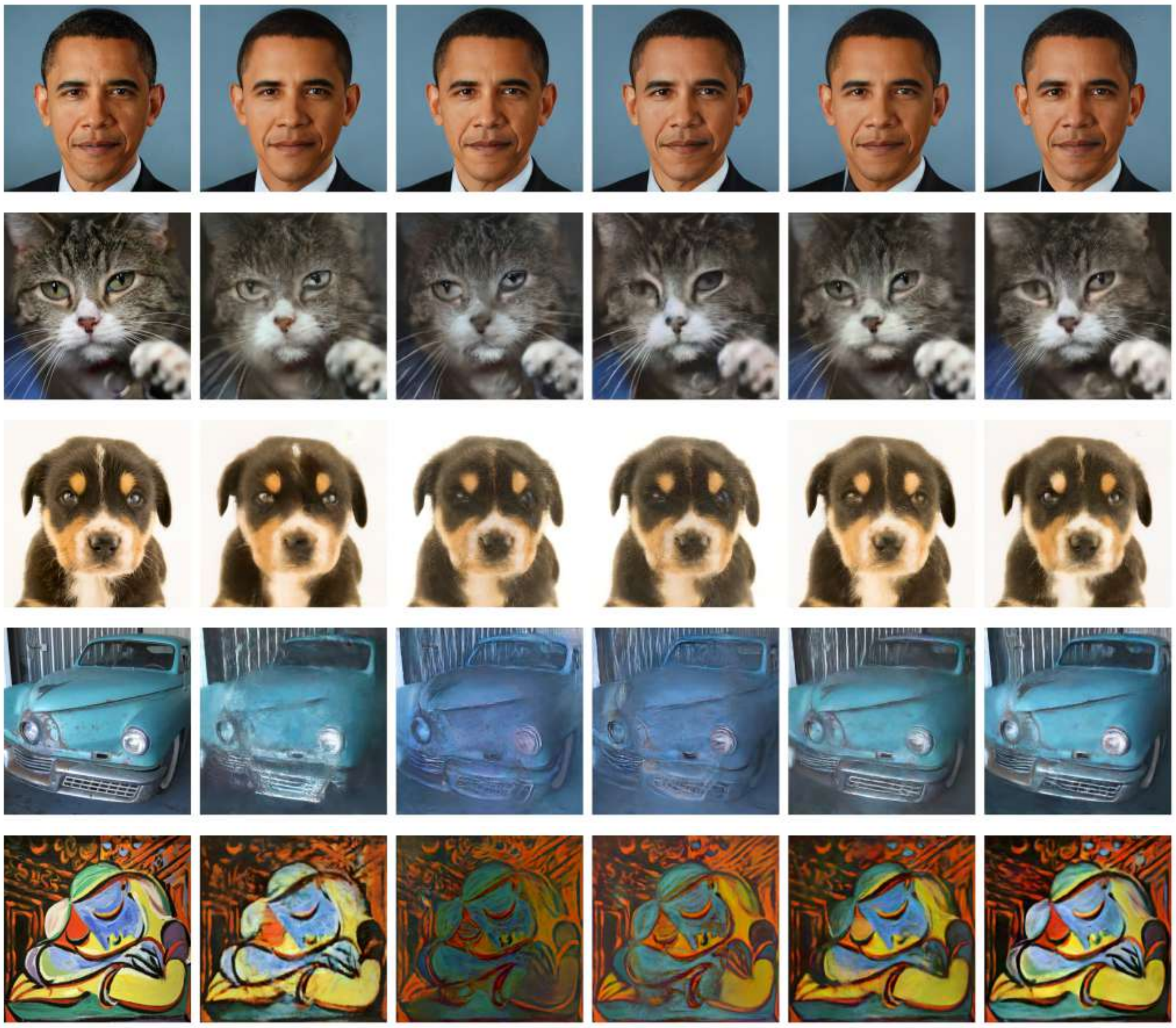


Figure 22: Additional results of the algorithmic choice justification on the loss function. Each row shows the results of an image from the five different classes in our test dataset respectively. From left to right, each column shows: (1) the original image; (2) pixel-wise MSE loss only; (3) perceptual loss on VGG-16 $conv3\_2$ layer only; (4) pixel-wise MSE loss and VGG-16 $conv3\_2$; (5) perceptual loss only; (6) our loss function .

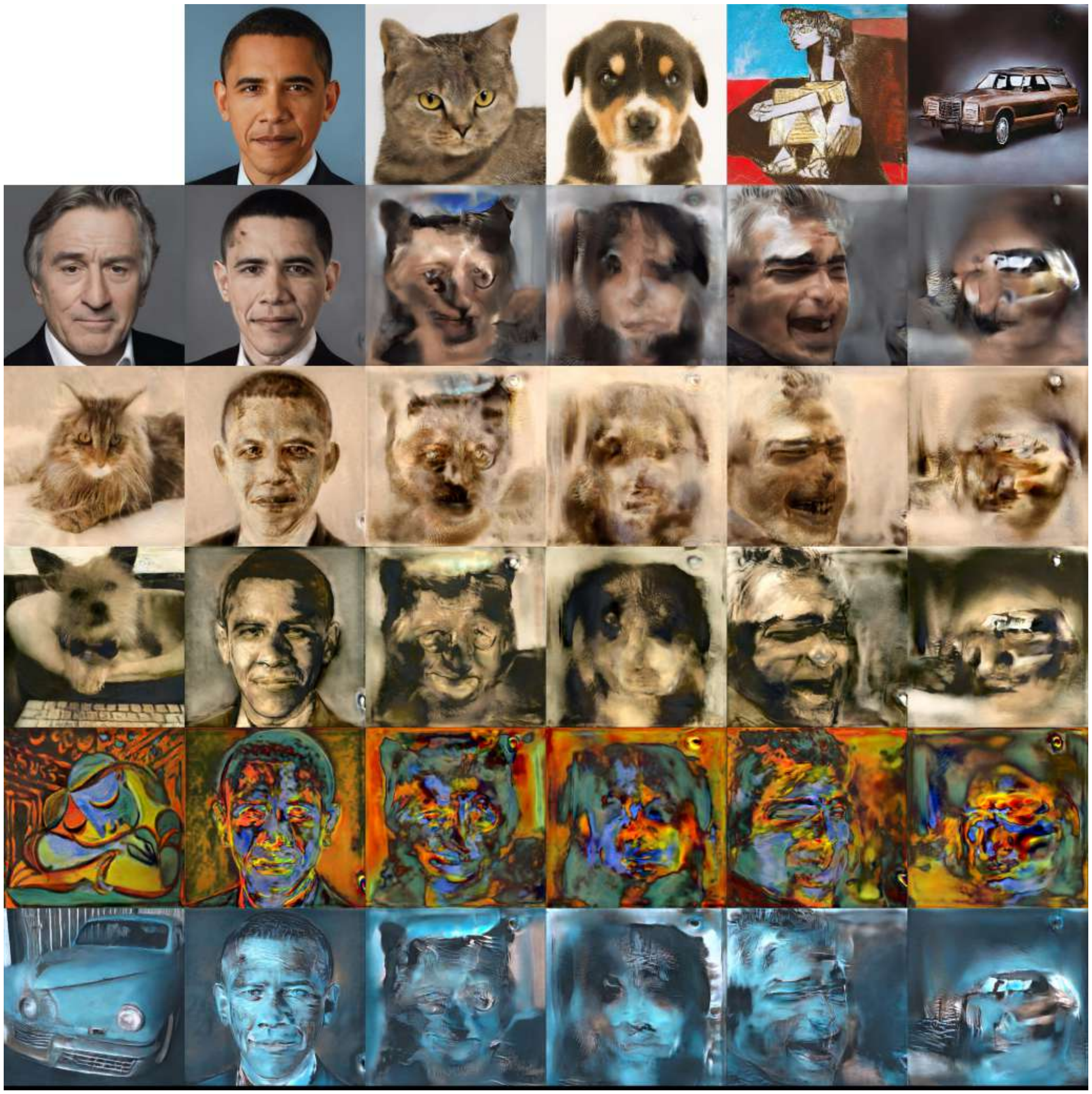
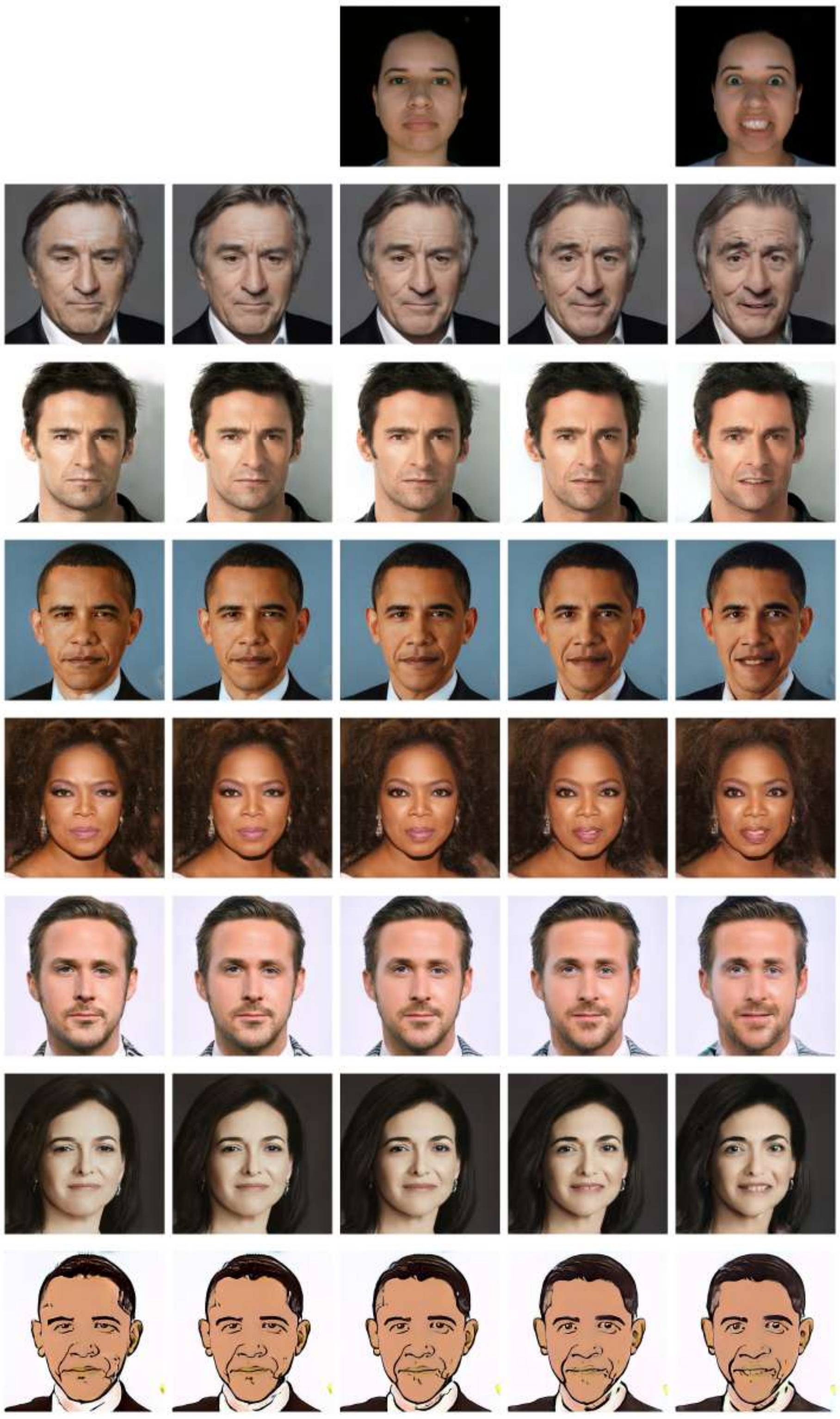
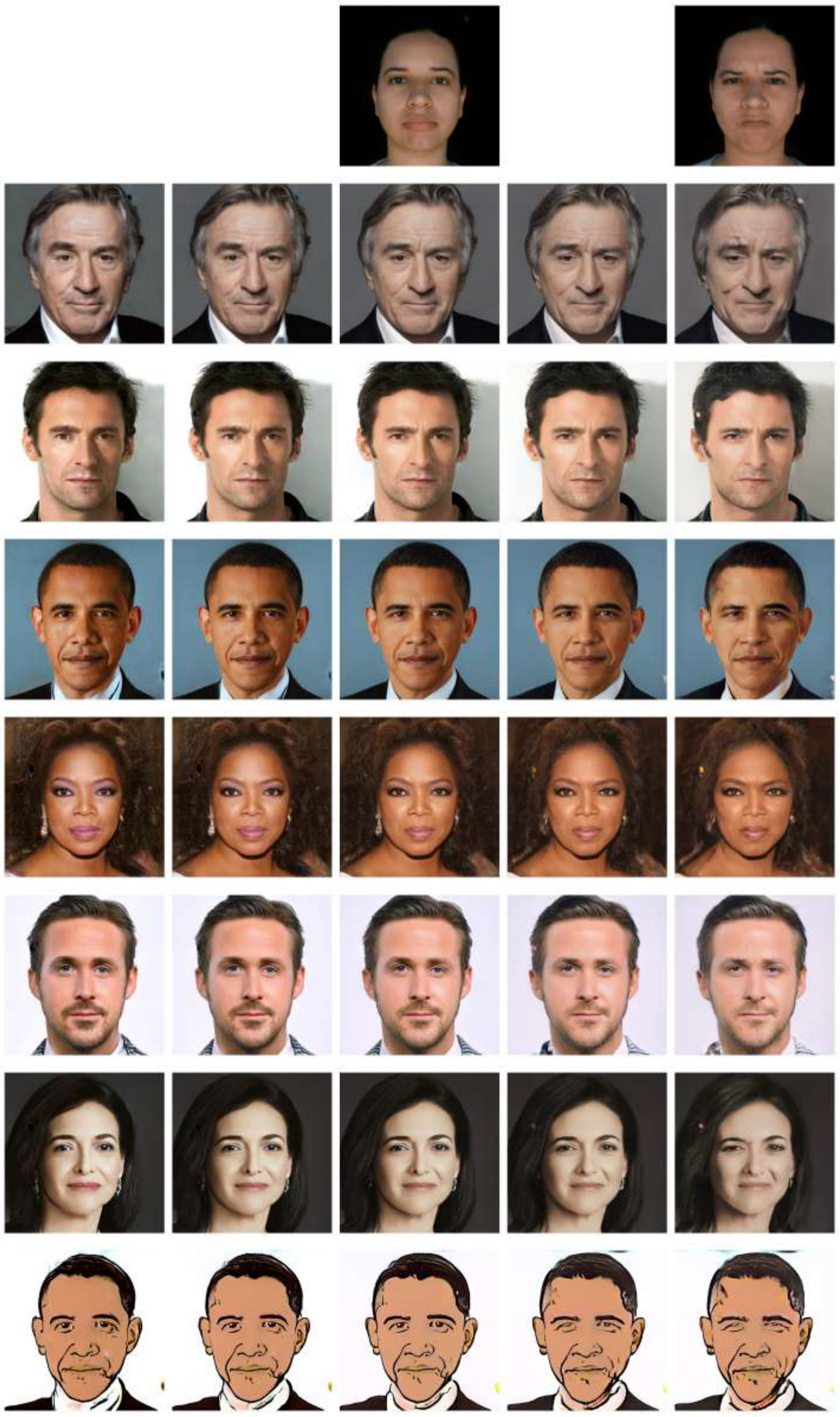


Figure 23: Complete table of the style transfer results. Left-most column: the embedded style image. First row: the embedded content images.



(a)



(b)

Figure 25: Additional results on expression transfer. In each subfigure, the first row shows the reference images from IMPA-FACES3D [24] dataset; in the following rows, the middle image in each of the examples is the embedded image, whose expression is gradually transferred to the reference expression (on the right) and the opposite direction (on the left) respectively.