A Thesis Submitted to the Department of Computer Science and Communications Engineering,
the Graduate School of Fundamental Science and Engineering of Waseda University
in Partial Fulfillment of the Requirements for the Degree of Master of Engineering

Anime Faces generation Corresponding to Human Faces

Advisor: Professor Hiroshi Ishikawa
Research guidance: Research on Computer Vision and Pattern Analysis

Wang Yiwen
5120FG52

July 18th, 2022

Abstract

Generative Adversial Networks have shown great performance in image-to-image translation, but usually it requires the original domain and target domain to be similar to some extent. When it comes to real human faces to Anime faces translation, however, the existing methods' performance can look too human-like, or like an anime that has no relationship with the input. This is because the anime domain is very different from the human face in both structure and texture. In this thesis, we show that with a fine-tuned generator, we can generate an anime face corresponding to the input human face by reverse generating on both domains, and then generating on a concatenated latent.

Acknowledgements

Here I would like to appreciate the advice and seminar arrangement from Prof. Ishikawa, without whom I will just girigiri around. And I would also express my thanks to Horiuchi-san and Tan-san in Ishikawa lab, who always answer my questions in terms of server and GPU usage. Finally, I would thank the company I interned for, named Pixelshift. It taught me to use tools such as Git and VScode to control my version and debug. Also, my coding ability is greatly trained here. Some habits such as keeping a working diary and always arranging my files well are from there.

July 18, 2022
Wang Yiwen

Contents

List of Figures

Chapter 1

Introduction

With the boom of social media and virtual reality techniques, people are thinking about how to project themselves into the digital world. We want the avatar to look like us to some extent, but also keep its potential to be different. There have been games allowing players to combine elements such as eyes, hair, clothes, and so on to create their own characters. Also, apps like Tiktok are providing filters to modify users' faces in a video to template styles. With the promise of machine learning for image manipulation, however, we expect the creation of our avatars in a click of a button, given reference to our photos, but with sufficient diversity. Anime can be a good domain for an avatar's appearance, whose facial appearance is human-like, but much more variate in terms of proportions and colors.

The problem of generating an image in the anime domain according to an image in the real human face domain can fall into the topic of image-to-image translation across domains. The task is clearly defined when we have paired data in two domains. But it also brings out the problem of finding the criteria to pair the data in two domains having no clear relations. For unpaired training, previous studies believe that two domains have relations in their higher level latent space.

In this thesis, we propose a simple yet effective method for image-to-image translation based on the fine-tuned StyleGAN generator. In particular, we show that by reverse generation on a style-transferred real human photo, we partially keep the real human's facial appearance but generate images in the target domain. Though having no constraints to ask for correspondence in the input and output, the front part of fine-tuned generators controls the contours and poses of a real human, and the latter layers fine-tuned in anime control the textures. The reverse-generated latent codes make sure that the generator is creating images with the same higher-level latent code.

Chapter 2

Literature Review

Existing methods of generating anime corresponding to human faces mainly consist of two directions: style transfer and image-to-image translation. Methods that implement the first direction reflect the input images well as style transferring mainly modifies the textures rather than the structures. The second direction, however, allows the exaggerated anime shapes but somehow loses the correspondence. We try to combine the advantages of both categories.

2.1 Image Generation

Generative Adversarial Networks (GAN)[9] may be used in a variety of applications, including image synthesis, semantic image editing, style transfer and so on. The generated face images have been greatly improved in terms of resolution, authenticity, and diversity. A GAN structure includes a generator and a discriminator. The task of a generator is to randomly generate synthetic data that resembles the real images. While a discriminator's task is to distinguish between true and false. Ideally, a discriminator should output high scores for all real pictures and low scores for all fake pictures.

Although people have been able to use deep learning to forge various real-life images of good quality before, they have not been able to generate satisfactory anime images. Until the advent of StyleGAN[12], non-realistic image generation tasks such as anime were also attained.[2]

StyleGAN[12] implements the GAN structure for the whole pipeline but innovates the generator compared with previous works. The generator consists of a mapping network f and a synthesis network g. g takes the style latent w $\in$W output from f and the noise n into stacked layers of AdaIN[10] and progressively processes a random constant vector into an image.

$$x_i^{'} = x_i + n_i, \tag{2.1}$$

$$\mathrm{AdaIN}(\mathrm{x_i^{'}}, \mathrm{w}) = \mathrm{w_{s,i}} \frac{\mathrm{x_i^{'}} - \mu(\mathrm{x_i^{'}})}{\sigma(\mathrm{x_i^{'}})} + \mathrm{w_{b,i}}. \tag{2.2}$$

where each feature map x_i is normalized separately, and then scaled and biased using the corresponding scalar components from style w. [12]

Though the mapping network provides so-called styles, f takes no input of images during forwarding. It is actually a Multi-layer Perceptron[7] that takes in a random vector z $\in$Z and outputs a vector w $\in$W of the same shape. f is used to fit W into any distribution that the target domain may want, not only Gaussian.

StyleGAN[12] has successfully generated images that people can not tell whether they are fake or not. And it could be promising to make use of the structure in our anime generation. But the problem is how to add control into StyleGAN structure to generate images that satisfy our task, which is 1) the identification as the human image input 2) diversity as in the anime domain.

Another model named GANs N'Roses [6] also tries to generate the mapping between the anime domain and the human faces domain. Its effect is as the second column showed in Figure A.1, Figure A.2, and Figure A.3. This paper's approach is a multi-modal image-to-image framework that uses style and content to simply formalize the mapping, in which the most important step is to understand exactly what the content and the style are. The paper adopts a specific definition: the content is what varies when the face image is subjected to a series of data augmentation transformations, while the style is unchanged. Here, data augmentation transformations involve scaling, rotation, cropping, etc. These allow the network to realize that content is essentially where the face part is in the image, and the style is how the facial appearance is rendered.

2.2 Style Transfer

The popular work of Gatys et al.[8] firstly introduces the method of style transfer, in which they separate content and style and show their connections with perceived artistic imagery. By convolution layers, images are represented with high-level latent code and compared with images from different domains.

AdaIN[10] later gives out a simple yet effective approach to enable arbitrary style transfer in real-time. In particular, it introduces Instance Normalization to normalize while keeping the content of a single image and then inserts style with scale and bias.

StyleGAN[12] implements AdaIN on a large scale and successfully generates images of humans who don't exist in the world. A section inside the work named style mixing inventively concatenates styles from two sources and shows that with the mixed styles, the output has features from both image sources. The paper also claims that the style in shallower layers tends to control large features such as poses and hairstyles. While that in deeper layers tend to control delicate features such as eyes and skin details.

Taesung et al.[18] try to explicitly separate the style and content by switching the combination of the latent code representing them. They break the content by cropping the images into pieces and introducing a piece-wise discriminator to maintain the style.

FreezeD[16] makes use of the idea of style mixing and extends it to the experiment when training the model. It fine-tunes the StyleGAN[12] discriminator with the target source.

Thus the front part of the discriminator can extract information in the FFHQ[13] domain, and the latter part of the discriminator can extract information in the target domain. By providing the latent code, FreezeD generates corresponding images across quite unrelated domains such as human and eagle.

Swapping Autoencoder[17] proposes some swapping strategies for auto-encoding that can address a range of image transformation problems, including texture swapping. It converts the input image into latent code, which is divided into structure code and texture code and merges into a newly generated image by exchanging these two parts. Compared with other methods using conditional GAN, the framework proposed by this method can be obtained with one or very small samples during the training phase. Compared with other methods using latent codes, the code space of this method is learned rather than sampled from a fixed distribution, which makes the method more flexible. Its effect is as the third and fourth column showed in Figure A.1, Figure A.2, and Figure A.3.

2.3 Image to Image Translation

The pioneering work of CycleGAN[19] solves the mapping between two domains with unpaired data and two sets of generators and discriminators in each. This model structure projects images from two domains back and forth and uses Cycle Consistency Loss[19] to measure the quality of generation.

TraVelGAN[5], however, points out that Cycle Consistency Loss limits the ability to map between domains that are too different. Here they borrow the often used idea from Natural Language Processing that the distance between the meanings of words can be computed by the calculation between vectors. Thus TraVelGAN[5] adds a model to learn high-level semantic representations of each image.

U-GAT-IT[14] incorporates a new attention module to pay attention to the differences between two domains and provides a reference for the mapping generator. It also experiments on human faces and anime faces and gets state-of-the-art performance.

FreezeG[15] shows that a generator can have a single mapping between a latent space and images domain and fine-tunes on g in a StyleGAN[12] as Figure 2.1 shows. In FreezeG[15], a trained g will always generate the same images if input the same latent z. In other words, the latent space Z and output images' space X are single mapping. Thus given any real human images x, we can find its corresponding latent z as the reverse generation process showed in Figure 2.2. For the process in Figure 2.2, a learnable vector z_0 is input into a fixed g and output a image $\mathrm{x}^{'}$. By computing the difference between x and $\mathrm{x}^{'}$ as Equation 2.3, we can update gradient on z_0 until x and $\mathrm{x}^{'}$ are almost the same.

$$L = \alpha L_{\text{perceptual}}(x, x^{'}) + (1 - \alpha) L_{\text{MSE}}(x, x^{'}). \tag{2.3}$$

where $L_{\text{perceptual}}$[11] compares high-level feature differences between images. And L_{MSE} is the Mean Square Error loss which compares per pixel difference between images. α decides how much we rely on each loss. This process will be call function $r_g(x)$.

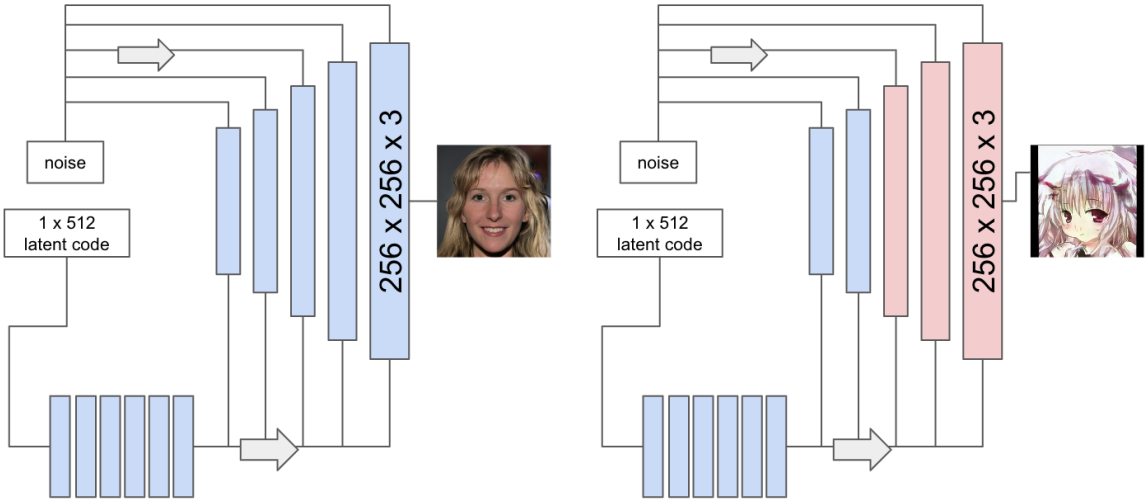

Figure 2.1: FreezeG[15] pretrains the generator as the left. Then fixes the front blocks and fine-tune the final blocks on anime dataset as the right.

Input a real human face image x and we get its latent z by reverse generating $z = r_g(x)$. Vice versa, input z into g and we get $x = g(z)$. Moreover, by experiments FreezeG shows that $g_{\text{fin}}(z)$ generate mapping images across domains that look like the human faces input, because g_{fin} and g share partially the same parameters.

StyleGAN[12] shows that the font blocks of the synthesis network tend to control pose, general hairstyle, face shape, etc. Because they compute with coarse spatial resolutions. While the final blocks are more related to smaller scale facial appearances, such as hair streams, eyes' status of open/close and etc, because dealing with more delicate resolutions. This explains why the idea of FreezeG works in cartoon that has salient human features such as Simpsons, for the fine-tuned synthesis network blocks helps modify the cartoon style textures, while the front blocks in the synthesis network maintain the input real human face's outlines.

Performance for experiments on anime, however, is not as satisfying. The generated $g_{\text{fin}}(z)$ seems no relation with the input image x. Possibly it is because anime varies more in terms of proportion and size of face features. And the fine-tuned blocks are not able to edit the outlines.

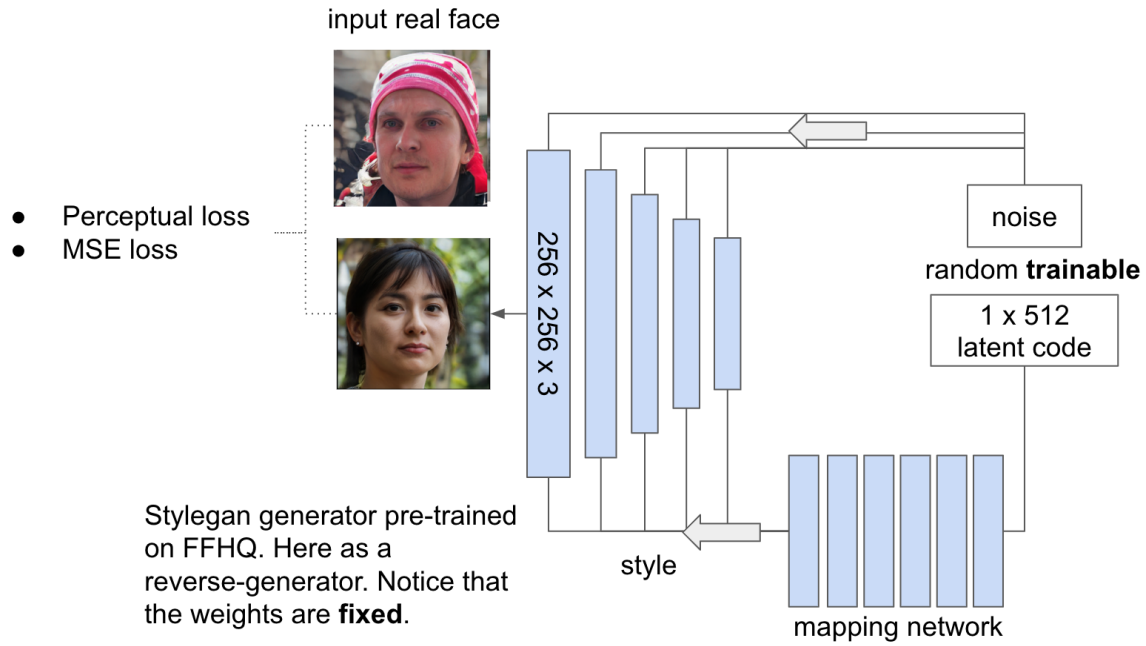

Figure 2.2: Reverse Generation. A learnable vector z_0 is input into a fixed g and output an image $x^{'}$. By computing the difference between x and $x^{'}$ as Equation 2.3, we can update gradient on z_0 until x and $x^{'}$ are almost the same.

Chapter 3

Proposed Method

StyleGAN[12], while has been able to successfully mass-generate fake real-life human images, as well as many other image domains, is an uncontrolled generator. In other words, we cannot directly use StyleGAN[12] to complete the correspondence generation from the real-life faces domain to the anime domain here, although it can generate perfect pictures in the two domains respectively. FreezeG[15] attempts to use model grafting to establish a connection between two domains. However, as Figure 3.2 shows, the outputs reflect no facial appearance from the input. Because there are no constraints for any levels of features.

Based on these problems, we propose a method to generate similar anime face images for input human face images. This method will use the structure of StyleGAN[12] to generate and reverse generate on our source domain and target domain, which are human face images and anime face images, respectively. We believe a mapping between different domains share the same latent, and through reverse generation to find the latent vector. We will expound on the method, dubbed Cross Domains Reverse Generation in Section 3.3. Before that, generation and reverse generation based on StyleGAN[12] and FreezeG[15] will be introduced as preliminaries.

3.1 Generation

Here generation refers to the process of generating images from a latent vector, into a specific domain. The structure of our generator is the same as StyleGAN[12], which consists of a synthesis network and a mapping network. The synthesis network uses a progressive generation method, which enlarges the resolution two times in width and height per block. Also, every block inside the generator consists of convolution layers and AdaIN[10] layers, which will adjust the style of the resolution as the 'style' and 'noise' suggest. For the mapping network, it takes in a random latent vector and outputs a latent vector in the same shape, as a style to the synthesis network. For the reason to use the mapping network, it is because we want to shape the style into any distribution that the target domain may want, rather than being decided by human experience in the input.

The main difference between our work here and styleGAN is that we fine-tune the gen-

erator of styleGAN. To generate images in the anime domain, there are two ways. First, we prepare an anime dataset that is as large and diversified as FFHQ[13], and we train as much time as styleGAN does. The alternative is to fine-tune the final layers of the synthesis network in the anime domain. It turns out that it takes much less time and the number of images input to fine-tune than train from scratch. And the result is satisfying enough. By fine-tuning, we mean fixing blocks of the mapping network and the front blocks of the synthesis network in a generator, and only training the final blocks of the synthesis network. In this way, we not only generate anime images at a small cost, but also get a special generator, whose front part is trained from human face images, and the final part is trained from anime face images. To generate an anime face that looks like a human face, we suppose that the contour and main pose of the result are similar to the human face, and the details and textures should fall into the anime domain. StyleGAN[12] has analyzed and confirmed that the front parts of a generator control mainly the pose and big facial appearances of an image, while the final parts control the details. So the fine-tuned generator provides us with a potential tool to solve the task.

3.2 Reverse Generation

Reverse generation is literately the reverse process of generation, which is given an image and generate its corresponding latent vector. Specifically, when given a generated human face image from an FFHQ[13] trained generator, it must have a corresponding latent vector, for it is needed as an input when generation. Then by fixing the generator, and initiating a random and learnable latent vector, we can keep generating on this latent, comparing the output image and the given image, and editing the latent, until the output image is almost the same as the given image. For the criteria of comparison, we use a combination of perceptual loss and mean square error loss. Notice that the output will not be exactly the same as the given image, but without careful observation, it should not be told. Because the reverse generation process is also a machine learning process, which has a threshold for epochs and accuracy. Similarly, for a generated anime image from a fine-tuned generator, we can reverse generate a latent vector for it by fixing the fine-tuned generator.

When given an image not generated, we can also use reverse generation to find its latent. Because we are supposing that our generator is perfect enough to generate all images in the human faces domain. So it is in the anime domain.

3.3 Cross Domain Reverse Generation

In this section, we propose the main point of this thesis. Beforehand we have mentioned reverse generating a real human face x with a synthesis network g trained on the real human face dataset FFHQ[13]. Similarly we can reverse generate a real human face x with a fin-tuned synthesis network g_{fin}, as shown in Figure 3.1, though the result must not be the same as x, because g_{fin} generate images in anime domain. But under the measurement of

combined loss of Perceptual Loss and Mean Square Error Loss, we can get the 'closest' latent $z^{'} = r_{g_{\text{fin}}}(x)$ for a real human face image in the anime face domain. The examples of reverse generation are shown in Figure 4.10 and Figure 4.9

Perceptual loss plays a very important role in the process of cross-domain reverse generation. Compared to other loss functions, such as MSE, perceptual loss measures the difference between averages rather than the absolute difference between each pixel. Thus the function is often used to compare high-level differences, like content and style discrepancies, between images. Also, this allows us to evaluate whether two images are similar, rather than whether they are identical. It is very important in a cross-domain generation because we hope that the generated anime characters are similar to the input real people, and we also know that similar anime characters and real people cannot be exactly the same.

It is worth noting that we use both perceptual loss and MSE loss in the design of the loss function, in which the latter is to control some necessary details, such as we want anime characters to have eyes where human eyes are. To trade-off between detail consistency and similarity, we conduct some experiments on the weighting of perceptual loss and MSE loss and choose the best combination.

After reverse generation, we obtain the latent code and generate anime images from it. Compared with other works, we do not impose explicit constraints on the output image and the input image. In other words, we did not require any degree of correspondence between images of two domains when training the generator. They are independent networks, training with different datasets respectively. This method is out on the idea that a mapping between two different domains should share the same latent at a higher semantic level.

Finally, we experiment with doing some edit on x for it to be more easily reverse generated by $r_{g_{\text{fin}}}(x)$. Reducing the fine-tuned layers in g_{fin} will generate images $x^{'}$ with more bolder contours, and closer to the style of paintings. And it turns out that reverse generation $r_{g_{\text{fin}}}(x)$ gives better anime effect.

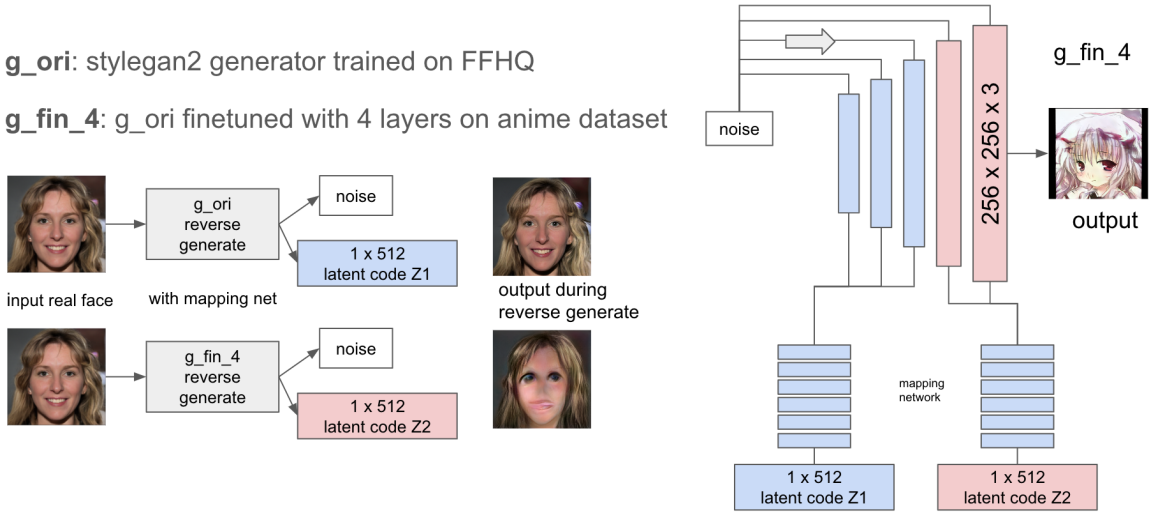

Figure 3.1: Reverse Generation across domains and concatenate the latent vectors to provide for the fine-tuned generator.

Figure 3.2: Output examples from FreezeG[15]. The first row is the input human face images. And the second is the output, every column respectively.

Chapter 4

Experimental Results

In this chapter, we will first show the best result. And then according to the order of image processing, fine-tuning, reverse generation, and generation, we show the results of different hyper parameters' combinations.

4.1 Best results

4.1.1 Purpose

Here we recall that the input of our task is a real human face image, and the output should be an image in the anime domain that can tell having the same identification as the input. All the results here will be the same five people as input. For more examples please check the appendix B from Figure B.1 to Figure B.6.

4.1.2 Procedure

We firstly have an FFHQ[13] pre-trained model g. Then we fine-tune its last final 4 blocks with an anime dataset [1] obtained from Kaggle, which is a website organizing machine learning related competitions. Then we get g_{fin4}, in which 4 represents for 4 blocks fine-tuned. By reverse generation on real human images x with g we get the latent z. And we only use the final block of g_{fin4} together with all the blocks except for the last in g to form a new generator g_{fin4use1}. We go through z into g_{fin4use1} and get $x^{'}$. Finally we reverse generate $x^{'}$ with g_{fin4}.

4.1.3 Results

Figure 4.1 shows our best results. The first row is the input human faces images, and the second row is the corresponding output, each column respectively.

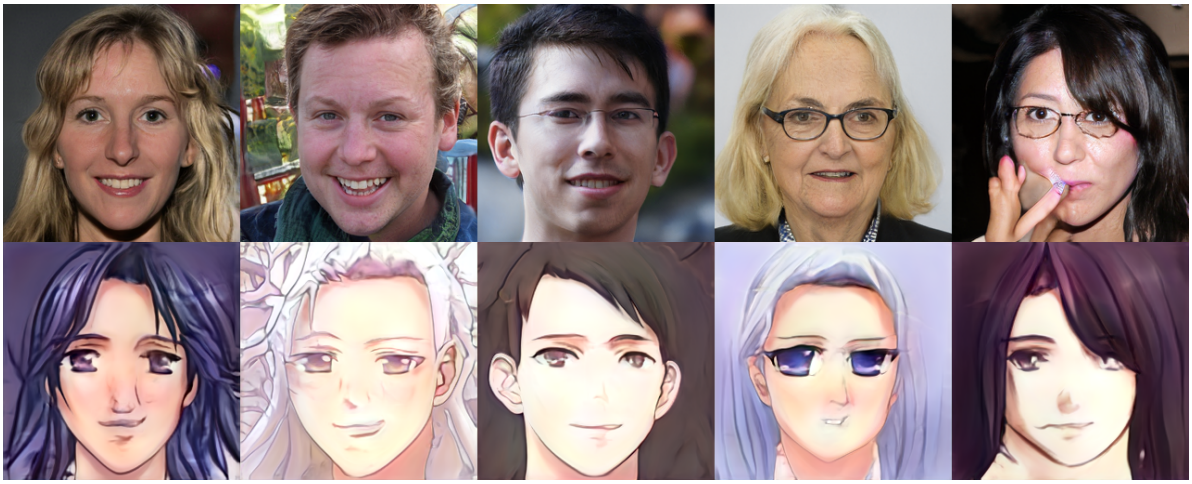

Figure 4.1: The first row is the input images. And the second is the generated corresponding images with our method in the anime domain.

4.1.4 Consideration

The poses and facial appearances are basically kept. And different inputs get output in different styles randomly. The features of anime such as big eyes and sharp chins are drawn. However, some details like eyeglasses are missing. From Figure B.1 to Figure B.6 in Appendix B we can observe that for some human examples, the reverse generation does well, such as women with long hair and big eyes. For children also. It can be explained by the majority of long hair female and feminine characters in the dataset.

Moreover, we see that some salient colors in the input could affect the color of the whole image in the output from Appendix B. And the contours of the anime faces' hair are usually not clear, which should be improved with extra constraints.

4.2 Image Processing

4.2.1 Purpose

We use an anime dataset [1] consisting of 60k+ images obtained from Kaggle. Most of them are of sizes around 60x60. To use a dataset as input for training, we need to resize all the images to the same size. Meanwhile, for face generation, a unified field of view is necessary, which means that the face should be put into a similar position with a similar size in an image.

4.2.2 Procedure

We use a super-resolution model called CARN Model[4] to do super-resolution for original images of too small size. Compared with resizing tools from torchvision library, CARN

Figure 4.2: Input images of different sizes, with low resolution

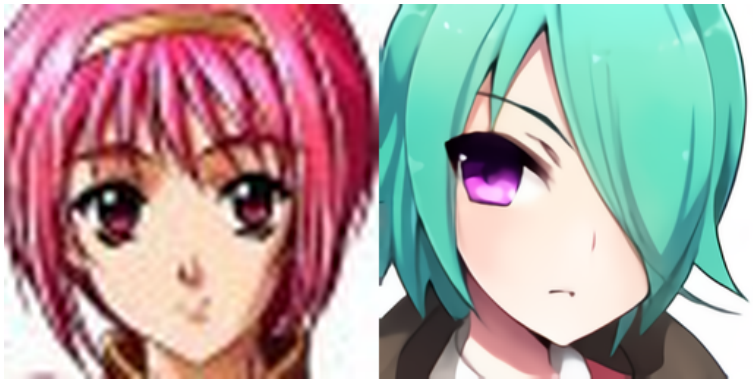

Figure 4.3: The left side image is resized to 256x256 with torch vision and resampled with LANCZOS. The right image is super resoluted with CARN Model[4].

Model [4] gets a better performance, which uses the structure of ResNet to compensate for the missing information during progressive enlargement and is good for practical use. Then we crop and resize all the images into a size of 256x256.

4.2.3 Results

Figure 4.2 are the samples of the original dataset, which are of different sizes and too small to be used for training a model with 256x256 output. Figure 4.3 shows the comparison between different methods of re-sampling.

4.3 Fine-tune

4.3.1 Purpose

Fine-tuning with a pre-trained model, we can use relatively less input and fewer computation resources. For a pre-trained model, we use the official version [3] from Nvidia.

4.3.2 Procedure

For the hyper-parameters, steps are 50000, batch size 8, and learning rate 0.002. The environment is Pytorch 1.4.0. On three Tesla V100, it takes around 13 hours to fine-tune. Here we show the results of fine-tuning with different numbers of blocks.

4.3.3 Results

As Figure 4.4,Figure 4.5,Figure 4.6,Figure 4.7,and Figure 4.8 show, fine-tuning different number of blocks give different artifact. Too few blocks, in particular, fail to generate fine result, because the capacity of learning is limited.

4.4 Reverse generate

4.4.1 Purpose

By reverse generation, we can get the latent vector with the shape 1x512 of an image. Within the same domain, the reverse generation can be perfect as Figure 4.9.

4.4.2 Procedure

With a generator g trained on human face images dataset FFHQ[13], we fix the parameters in g and input a random learnable latent z into it. We get images output as $x^{'} = g(z)$. By comparing $x^{'}$ with x through perceptual loss and mean square error, we update the gradient on z, until x and $x^{'}$ look almost the same. Figure 4.9 show x on the left and $x^{'}$ on the right, with perceptual loss 0.1103 and mean square error 0.0193. Note that $x^{'}$ is for confirmation. The output of this process is z. For training details, learning rate = 0.1, steps = 1000.

4.4.3 Results

With a generator g_{fin} fine-tuned on an anime dataset, we also experiment with reverse generating on real human images. Here $x^{'} = g_{\text{fin}}(z)$ looks wired as Figure 4.10 shows. It is reasonable because an image in the anime domain cannot be the same as an image in real human images domain, in terms of perceptual loss and mean square error. However, it is still worthwhile because $x^{'}$ is the closest mapping for x in an anime domain.

4.5 Use a different number of fine-tuned blocks

4.5.1 Purpose

We fine-tune a generator g with its final 4 blocks to get g_{fin4}, and experiment with generating images with fewer blocks fine-tuned.

4.5.2 Procedure

To be specific, there are 7 blocks in a generator's synthesis network. Let's dub it g_{fin4use4} for using the fisrt 3 blocks from g and the final 4 blocks from g_{fin4}. Similarly, g_{fin4use3} is for using the first 4 blocks from g and the final 3 blocks from g_{fin4}.

4.5.3 Results

As Figure 4.11 shows, the less the fine-tuned blocks are used, the more alike the output will be to the input.

4.6 Concatenate latent to generate

4.6.1 Purpose

With a guess that parameters in g should work well with z reverse generated by g, and parameters in fine-tuned blocks in g_{fin} should work well with z reverse generated by g_{fin}, we give different blocks in g_{fin} with different z and check the images generated.

4.6.2 Procedure

Let g_{fin4} be dubbed for fine-tuning g with its final 4 blocks. $z1$ for x being reverse generated by g, and $z2$ for x being reverse generated by g_{fin4}. We then give $z1$ to first 3 blocks in g_{fin4}, $z2$ to final 4 blocks in g_{fin4}, and generate.

4.6.3 Results

As Figure 4.12 shows, this method works when the number of fine-tuned layers is enough.

Figure 4.4: Pre-trained model's output. Pre-trained with dataset FFHQ[13] for 55k steps, using the other settings same as in StyleGAN[12].

Figure 4.5: Output from g_{fin1}, which is g fine-tuned with its final block on anime dataset.

Figure 4.6: Output from g_{fin2}, which is g fine-tuned with its final two blocks on the anime dataset.

Figure 4.7: Output from g_{fin3}, which is g fine-tuned with its final three blocks on the anime dataset.

Figure 4.8: Output from g_{fin4}, which is g fine-tuned with its final four blocks on the anime dataset.

Figure 4.9: On the left is the input image. On the right is the image generated from a latent vector, which is reverse generated with g.

Figure 4.10: On the left is the input image. On the right is the image generated from a latent vector, which is reverse generated with g_{fin4}.

Figure 4.11: From left to right is the original image, the image generate on z by g_{fin4use1}, by g_{fin4use2} and g_{fin4use3}

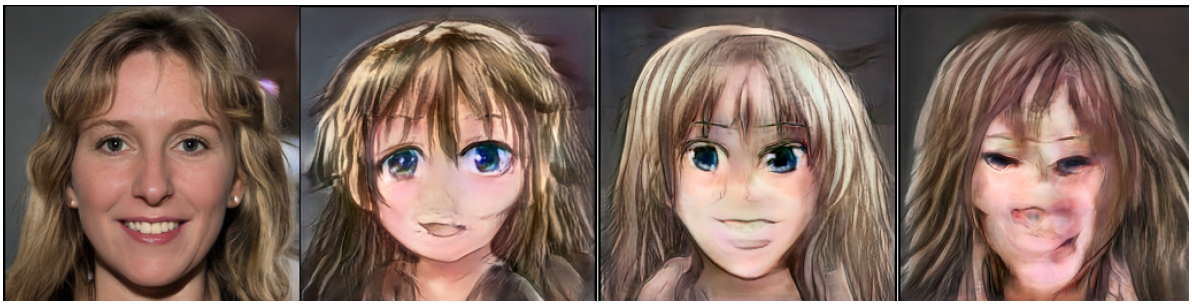

Figure 4.12: From left to right is the original image, concate latent for g_{fin4}, for g_{fin3}, and for g_{fin2}

Chapter 5

Conclusion

This thesis explores a method based on forward and reverse generation to create anime images similar to input real-life human faces images. Compared with traditional generation methods, the method proposed in this paper does not impose direct constraints on the generated images. Instead, it requires the image correspondence of the two domains in latent space. This high-level constraint makes the generated images both anime-style and similar to the input images.

In this thesis, various combinations of generation methods and hyper-parameters have been experimented with, such as fine-tuning different layers and epochs, assigning different weights to the loss function, and so on. The results of these experiments vary widely, indicating that even similar methods may produce large differences in the results under the framework of deep learning. Thus in the process of generation, it is important to keep the consistency of implementation details to obtain the same artifact.

References

[1] Anime face dataset. `https://www.kaggle.com/datasets/splcher/animefacedataset`. Accessed: 2022-06-29.

[2] Making anime faces with stylegan. `https://www.gwern.net/Faces`. Accessed: 2022-06-28.

[3] Nvidia stylegan2 networks. `https://github.com/NVlabs/stylegan2`. Accessed: 2022-06-29.

[4] Namhyuk Ahn, Byungkon Kang, and Kyung-Ah Sohn. Fast, accurate, and, lightweight super-resolution with cascading residual network. *CoRR*, abs/1803.08664, 2018.

[5] Matthew Amodio and Smita Krishnaswamy. Travelgan: Image-to-image translation by transformation vector learning. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, pages 8983–8992, 2019.

[6] Min Jin Chong and David Forsyth. Gans n'roses: Stable, controllable, diverse image to image translation (works for videos too!). *arXiv preprint arXiv:2106.06561*, 2021.

[7] Matt W Gardner and SR Dorling. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric environment*, 32(14-15):2627–2636, 1998.

[8] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style, 2015.

[9] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

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

[11] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 694–711, Cham, 2016. Springer International Publishing.

[12] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks, 2018.

[13] Vahid Kazemi and Josephine Sullivan. One millisecond face alignment with an ensemble of regression trees. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[14] Junho Kim, Minjae Kim, Hyeonwoo Kang, and Kwanghee Lee. U-gat-it: Unsupervised generative attentional networks with adaptive layer-instance normalization for image-to-image translation, 2019.

[15] Bryan Lee. Making anime faces with stylegan. `https://github.com/bryandlee/FreezeG`. Accessed: 2022-06-28.

[16] Sangwoo Mo, Minsu Cho, and Jinwoo Shin. Freeze the discriminator: a simple baseline for fine-tuning gans, 2020.

[17] Taesung Park, Jun-Yan Zhu, Oliver Wang, Jingwan Lu, Eli Shechtman, Alexei Efros, and Richard Zhang. Swapping autoencoder for deep image manipulation. *Advances in Neural Information Processing Systems*, 33:7198–7211, 2020.

[18] Taesung Park, Jun-Yan Zhu, Oliver Wang, Jingwan Lu, Eli Shechtman, Alexei Efros, and Richard Zhang. Swapping autoencoder for deep image manipulation. *Advances in Neural Information Processing Systems*, 33:7198–7211, 2020.

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

Appendix A

Comparison with Other Works

In Figure A.1, Figure A.2 and Figure A.3 we compare the results of our method with other works. From left to right, the first column is the input. The second from Guns' N Roses[6], the third and fourth from Swapping Autoencoder[17], and the fifth from our method.

The results are not cherry-picked. It can be seen that different methods have their advantages. Some give good details in the anime domain such as the light in the eye and the hair streams but lack diversity. Some are really good identifications of the input, but not in the anime domain, for which the shape and size of facial appearance almost no changes to the input human faces.

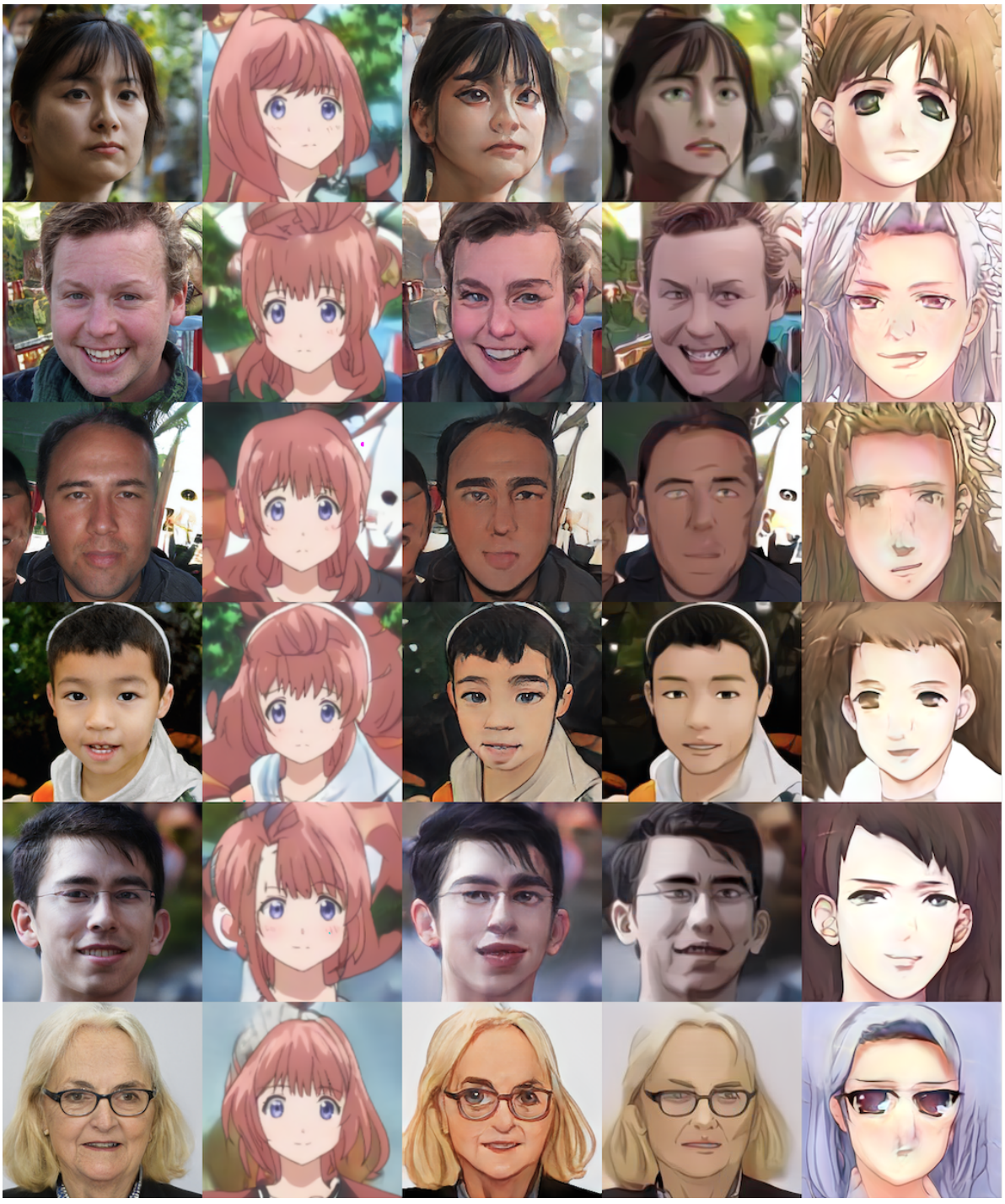

Figure A.1: Comparison with other works, part I. From left to right, the first column is the input. The second from Guns' N Roses[6], the third and fourth from Swapping Autoencoder[17], and the fifth from our method.

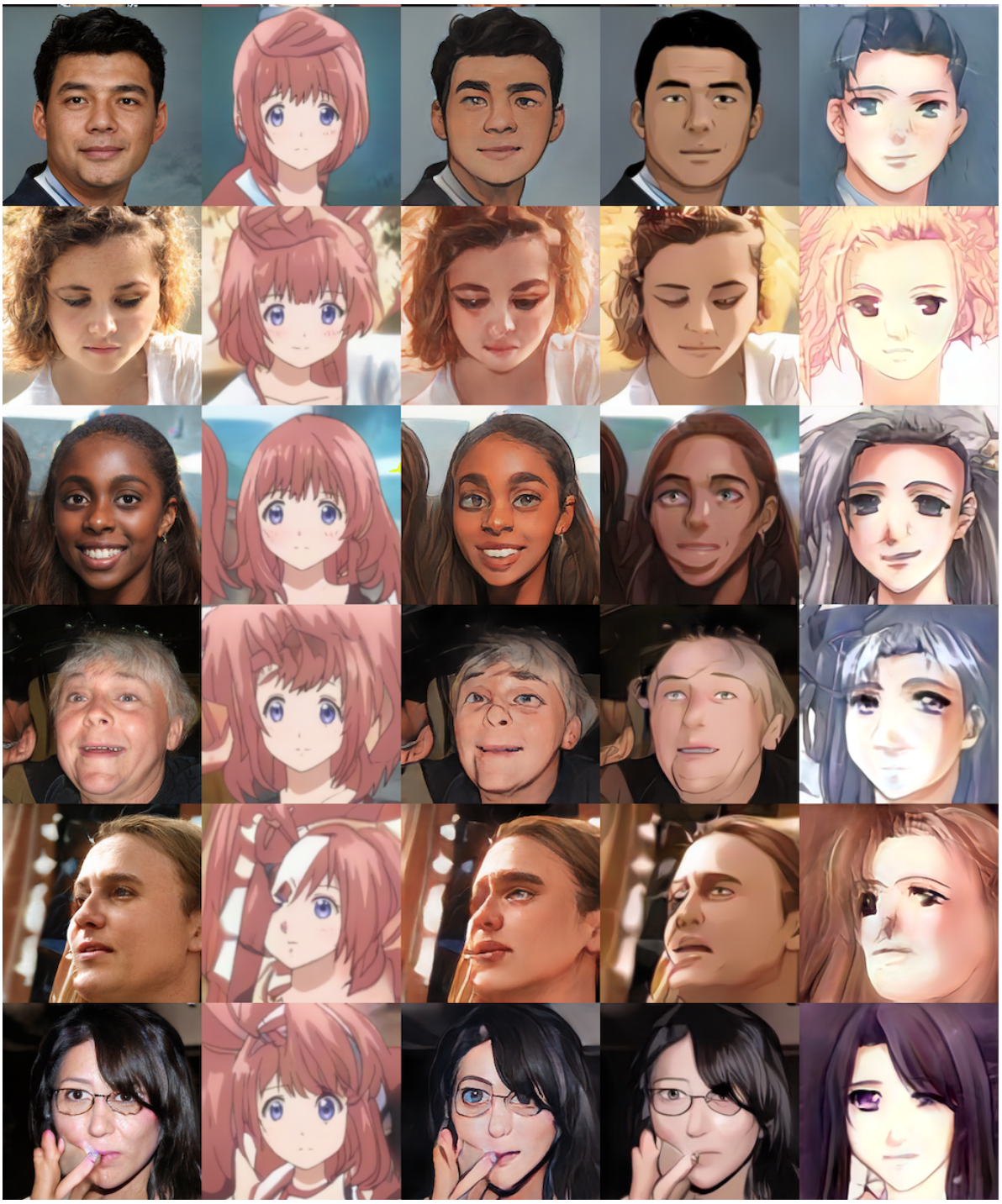

Figure A.2: Comparison with other works, part II. From left to right, the first column is the input. The second from Guns' N Roses[6], the third and fourth from Swapping Autoencoder[17], and the fifth from our method.

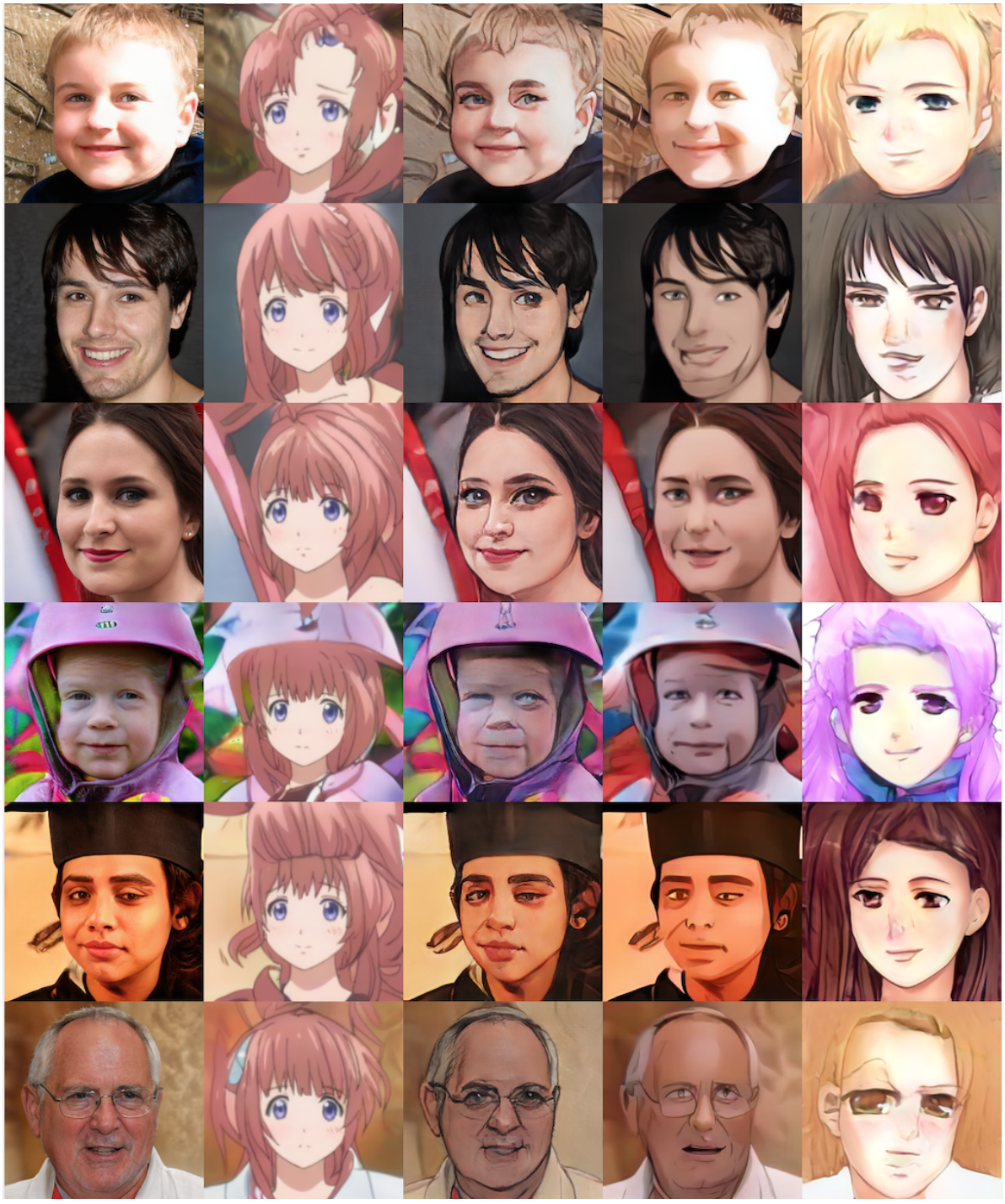

Figure A.3: Comparison with other works, part III. From left to right, the first column is the input. The second from Guns' N Roses[6], the third and fourth from Swapping Autoencoder[17], and the fifth from our method.

Appendix B

More examples

More examples of my generation are listed from Figure B.1 to Figure B.6. Every three images in a column is a group. In the first row is the input images. Second row the images after style transfer. The third row is our output with reverse generation.

- Environment of implementation:
  python3 tensorflow-gpu==1.14.0 scipy==1.3.3
  requests==2.22.0 Pillow==6.2.1 h5py==2.9.0
  imageio==2.9.0 imageio-ffmpeg==0.4.2 tqdm==4.49.0
  torch==1.4.0 torchvision==0.5.0 pandas numpy
  pillow==6.2.1 opencv-python scikit-learn
  matplotlib seaborn jupyterlab
  ninja lmdb wandb.

- The code and implementation of this project can be found in github

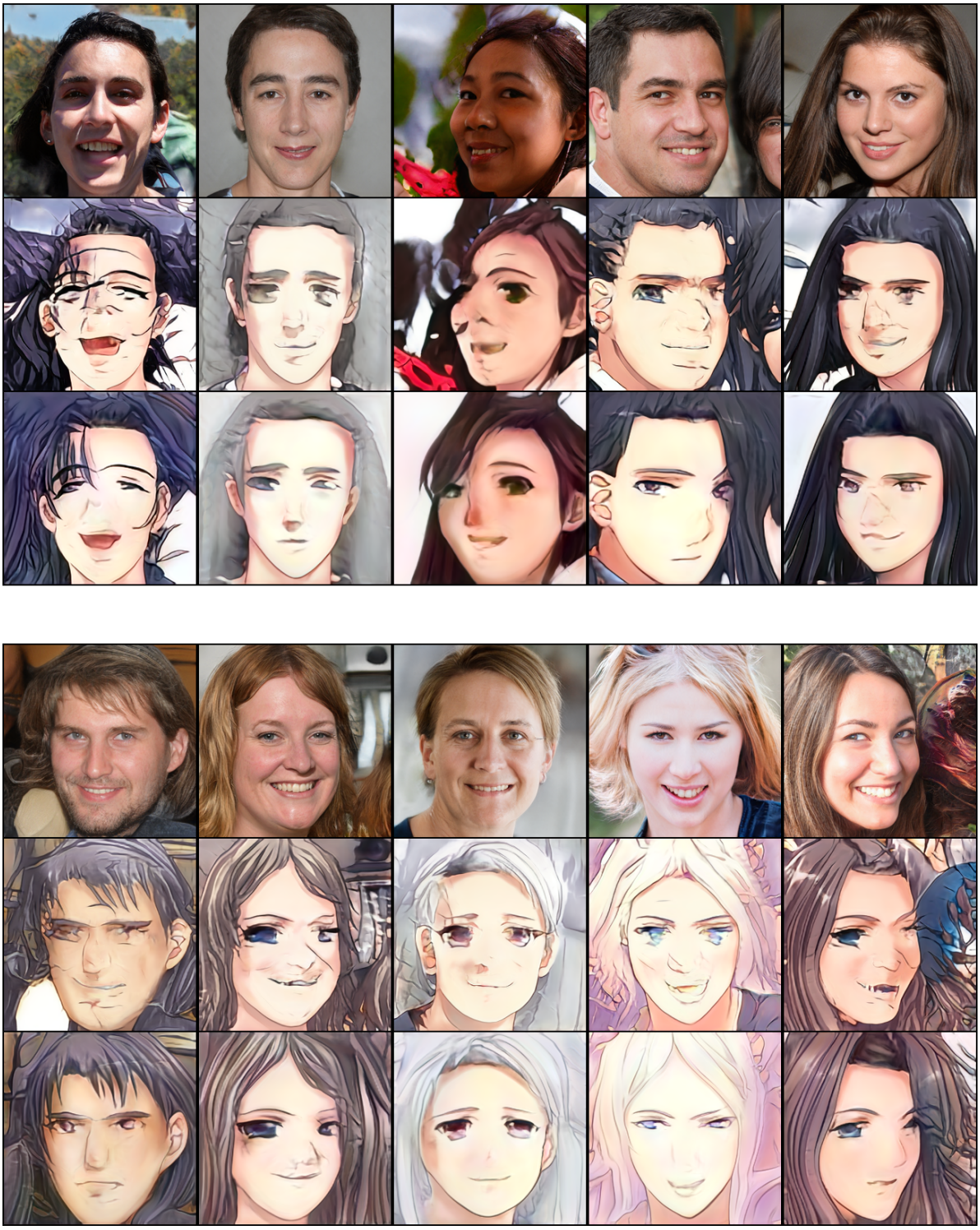

Figure B.1: Final examples part I. Every three images in a column is a group. In the first row is the input images. Second row the images after style transfer. The third row is our output with reverse generation.

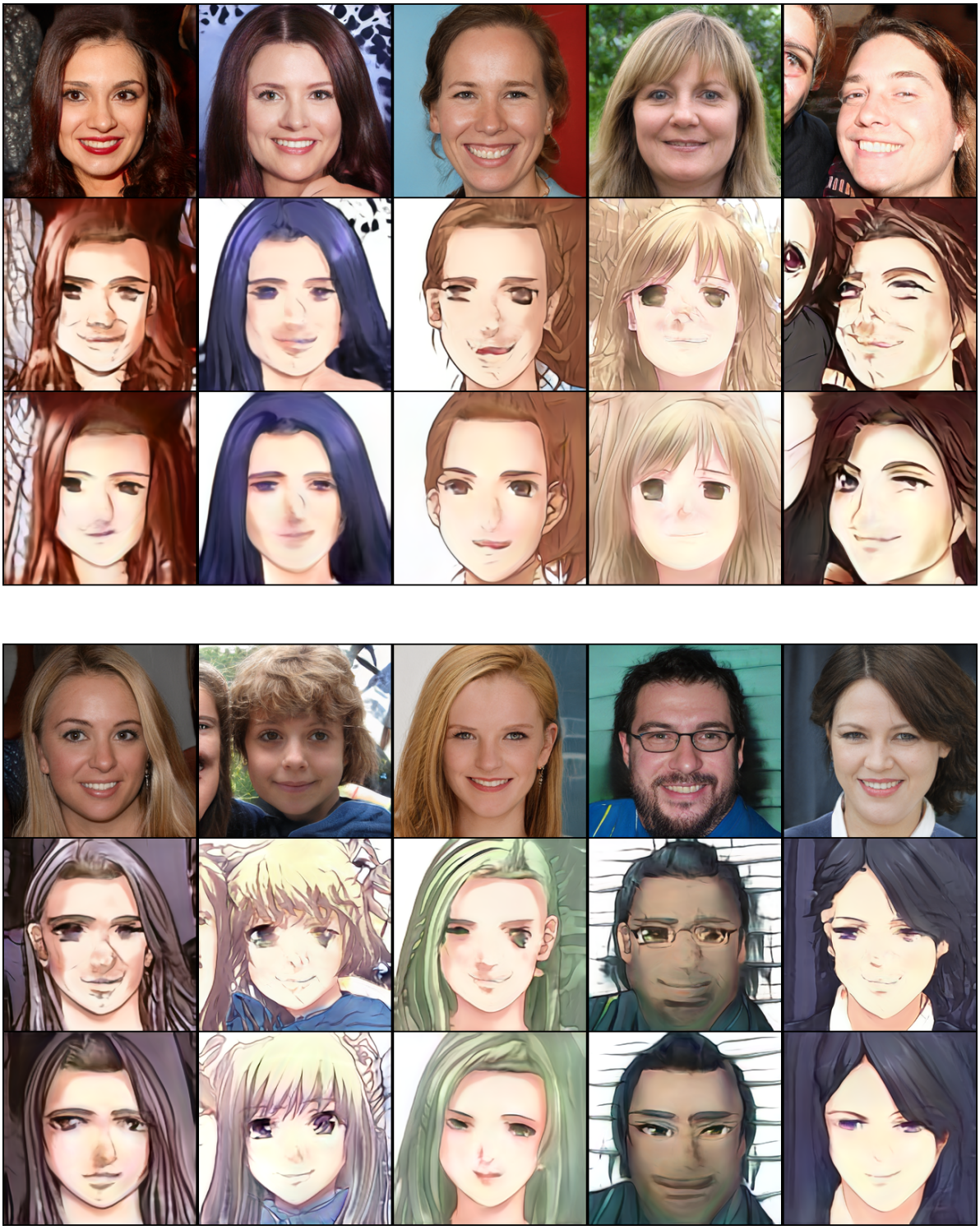

Figure B.2: Final examples part II. Every three images in a column is a group. In the first row is the input images. Second row the images after style transfer. The third row is our output with reverse generation.

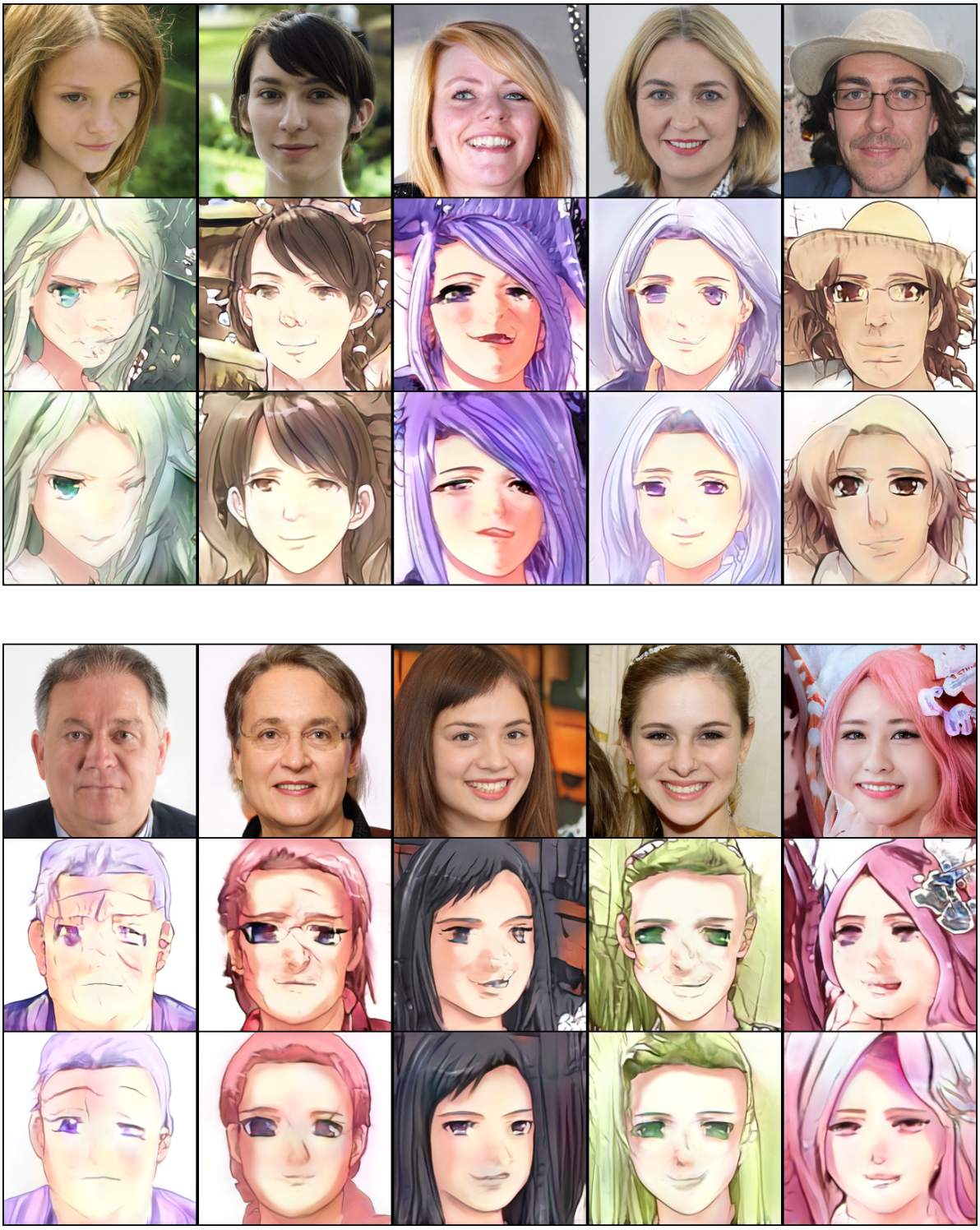

Figure B.3: Final examples part III. Every three images in a column is a group. In the first row is the input images. Second row the images after style transfer. The third row is our output with reverse generation.

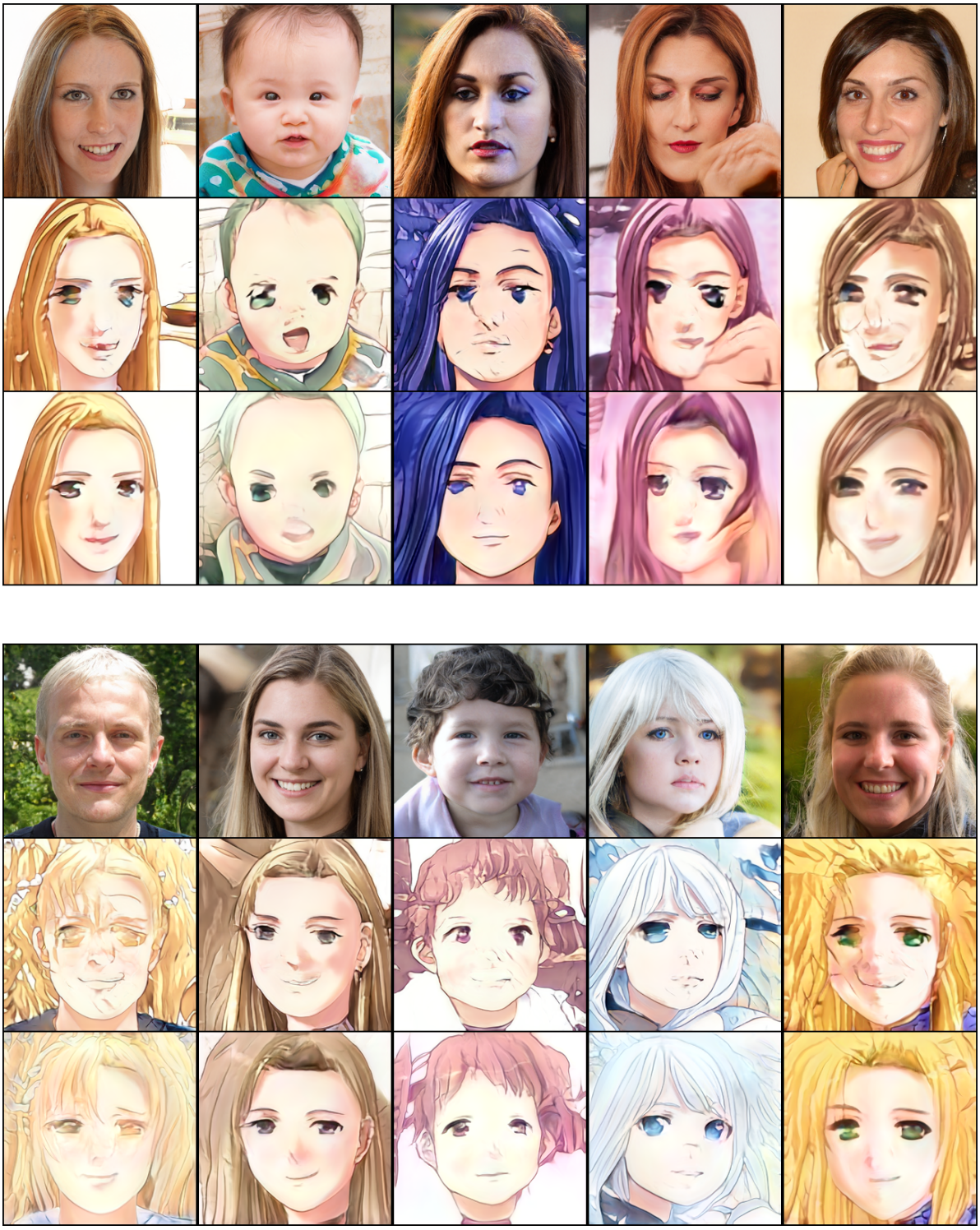

Figure B.4: Final examples part IV. Every three images in a column is a group. In the first row is the input images. Second row the images after style transfer. The third row is our output with reverse generation.

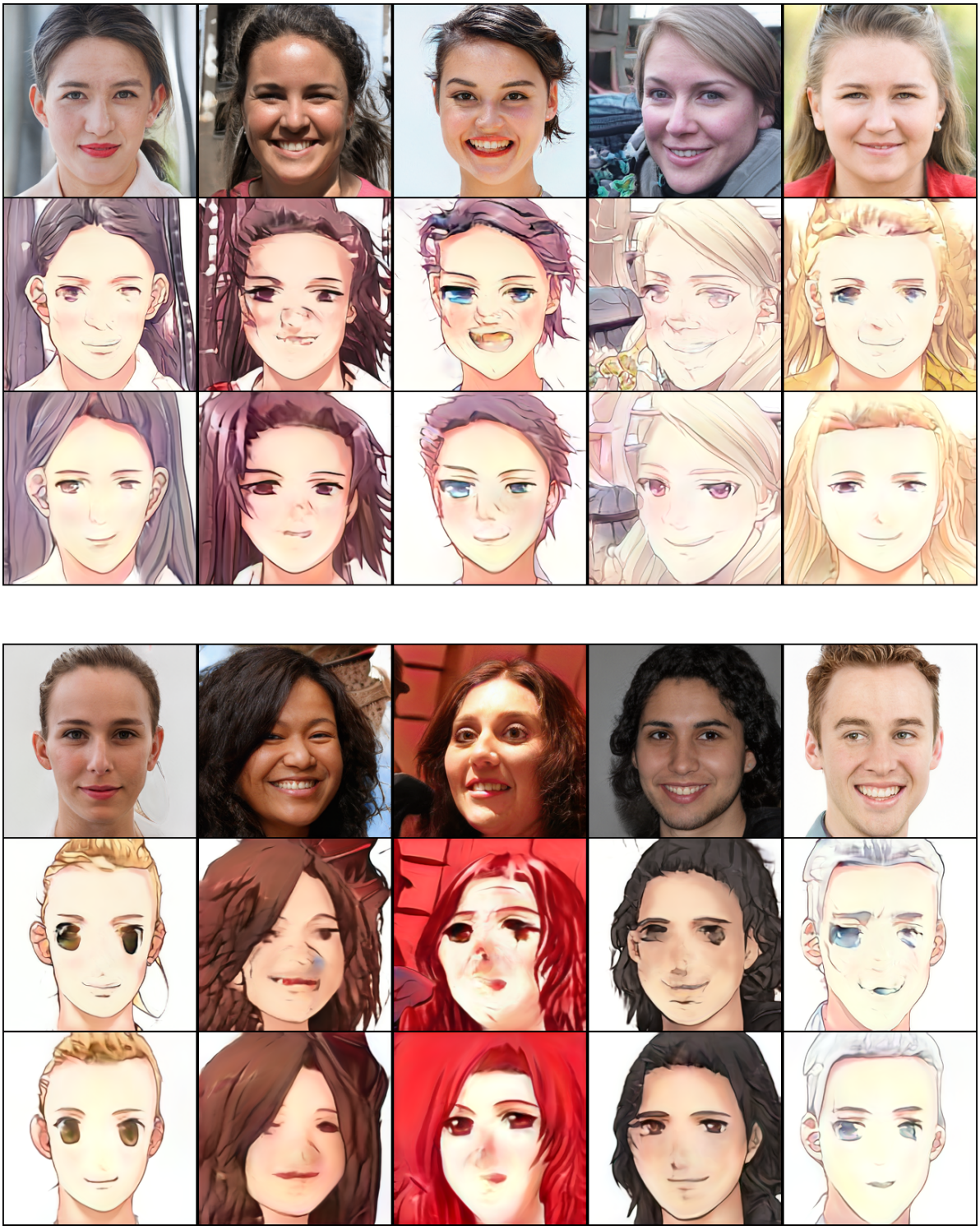

Figure B.5: Final examples V. Every three images in a column is a group. In the first row is the input images. Second row the images after style transfer. The third row is our output with reverse generation.

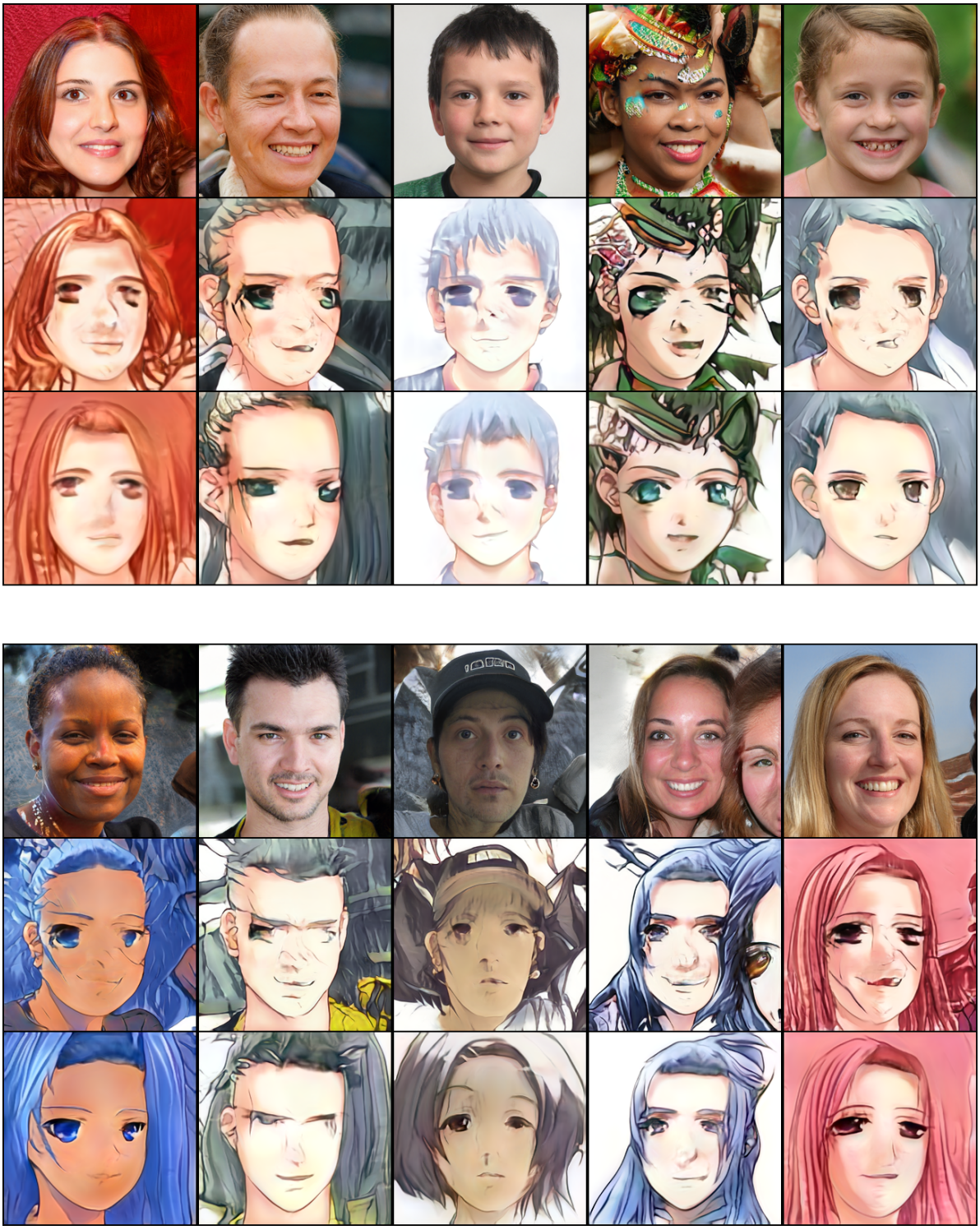

Figure B.6: Final examples VI. Every three images in a column is a group. In the first row is the input images. Second row the images after style transfer. The third row is our output with reverse generation.