GAN with Multivariate Disentangling for Controllable Hair Editing - Supplementary Material

Xuyang Guo[1,2], Meina Kan[1,2], Tianle Chen[1,2], and Shiguang Shan[1,2,3]

[1] Key Lab of Intelligent Information Processing of Chinese Academy of Sciences (CAS), Institute of Computing Technology, CAS, Beijing 100190, China
{xuyang.guo,tianle.chen}@vipl.ict.ac.cn, {kanmeina,sgshan}@ict.ac.cn
[2] University of Chinese Academy of Sciences, Beiijng 100049, China
[3] Peng Cheng Laboratory, Shenzhen, 518055, China

This supplementary material provides more details in six aspects: Section A gives more details about network architectures and the training procedure. Section B gives more details about how to automatically get color annotations. Section C presents details about exploring binary (i.e. weak) annotation to guide the semantic dimension disentangling for texture. Section D discusses the comparison between our method and other general GAN inversion-based editing methods. Section E shows details about testing of continuous editing ability. Section F shows more results about the disentangled dimension of color, texture, and shape, more comparisons with existing methods, and more visual results.

A Network Architectures and Training Procedure

Network Architectures for Hair Color and Texture Editing. The networks of color and texture have two parts, encoders (E_C, E_T) and decoder (D_X).

The encoder E_C of color is designed as a fully connection architecture with 3 hidden layers, as shown in Fig. 1. It takes hair feature X as input to get the latent representations Z_C. The encoder E_T of texture is designed as a fully connection architecture with 4 hidden layers, as shown in Fig. 2. It takes hair feature X as input to get the latent representations Z_T. A single decoder D_X is used to generate the edited hair feature $\hat{X}$ from edited latent representations $\hat{Z}_C, \hat{Z}_T$ of color and texture. The network architecture of the decoder D_X has 4 hidden layers, as shown in Fig. 3.

During optimization of the encoder and decoder, adversarial training with $\mathcal{L}^{real}$ is needed to ensure the reality of edited feature $\hat{X}$. So, a discriminator δ_X is introduced as described in Section 3.2 of our paper. The discriminator has a similar architecture with E_T and shares network weights as shown in Fig. 2 for saving parameters and better extracting common features.

Network Architecture for Hair Shape Editing. The networks for the shape editing are designed as convolution architecture considering that shape is position sensitive, consisting of the shape encoder and the shape adaptor. Since the input segmentation mask has no position information for convolution operation,

Algorithm 1: Training of CtrlHair

input : Images set $\Omega_I = \{I\}$ with its hair appearance features $\Omega_X = \{X\}$, color label $\Omega_{\tilde{Z}_C} = \{\tilde{Z}_C\}$ and portrait masks $\Omega_S = \{S\}$;
output: Encoders E_C, E_T, E_S, decoders D_X and shape adaptor Γ;
Initialize $E_C, E_T, E_S, D_X, \Gamma$ and discriminators $\delta_X, \delta_S, \delta_{Z_S}$;
while *not converge* **do**
`// training encoders and decoder for color and texture`
Sample a batch of latent representations $\hat{Z}_C, \hat{Z}_T \in \mathcal{N}(0, I)$;
Generate $\hat{X}$ using Eq. (4);
Sample a batch of appearance features $X \in \Omega_X$ with its color labels $\tilde{Z}_C \in \Omega_{\tilde{Z}_C}$;
Update E_C according to $\mathcal{L}_C^{dist}$, E_T according to $\mathcal{L}^{real}, \mathcal{L}_T^{rec}, \mathcal{L}_T^{dist}$, and D_X according to $\mathcal{L}^{real}, \mathcal{L}_C^{rec}, \mathcal{L}_T^{rec}, \mathcal{L}_T^{dist}$;
`// training encoder and adaptor for shape`
Sample a batch of masks from Ω_S and extract their hair masks S_H;
Sample another batch of masks from Ω_S and extract their face and background masks (S_F, S_B);
Generate adjusted portrait mask $\hat{S}_k$ from S_H, S_F, S_B using Eq. (2) and (5);
Update E_S according to $\mathcal{L}^{real}, \mathcal{L}_S^{rec}, \mathcal{L}_S^{dist}$, and Γ according to $\mathcal{L}^{real}, \mathcal{L}_S^{rec}$;
`// training discriminators`
Sample a batch of true appearance feature $X^r \in \Omega_X$, portrait mask $S^r \in \Omega_S$ and true latent representations $Z_S^r \in \mathcal{N}(0, I)$;
Update $\delta_X, \delta_S, \delta_{Z_S}$ according to their respective adversarial training losses;
end

inspired by NeRF [6], $1^{st} \sim 10^{th}$ order sine and cosine position embeddings are additionally concatenated for each input segmentation mask.

The shape encoder E_S takes the above-mentioned input to obtain the shape latent representation Z_S, and the encoder's network architecture is shown in Fig. 4. Then, the shape adaptor Γ takes latent representation $\hat{Z}_S$ of edited shape, face mask S_F, and background mask S_B as input, to generate edited portrait mask $\hat{S}$. The shape adaptor Γ's network architecture is as shown in Fig. 7.

During training, adversarial training is employed to ensure the reality of edited portrait mask $\hat{S}$ by $\mathcal{L}^{real}$ and the standard multivariate gaussian distribution of encoded shape latent representation Z_S by $\mathcal{L}_S^{dist}$. So two discriminators δ_S and δ_{Z_S} are designed. The network architectures are shown in Fig. 5 and 6.

Training Details. We use Adam optimizer [5]. The batch size is set as 16. For loss weights, we set $\lambda^{real} = 1$, $\lambda_C^{rec} = \lambda_C^{dist} = 10^{-2}$, $\lambda_T^{rec} = 10^2$, $\lambda_T^{dist} = 10^{-2}$, $\lambda_S^{rec} = 10^2$, $\lambda_S^{dist} = 1$. For all encoders, decoders, and the shape adaptor, the learning rate is set as 2×10^{-4}. For all discriminators, the learning rate is set as 1×10^{-4}. The training procedure is shown in Algorithm 1. To achieve better results, the shape branch can also be trained separately, i.e., separating the steps related to shape editing for better controlling the network parameters.

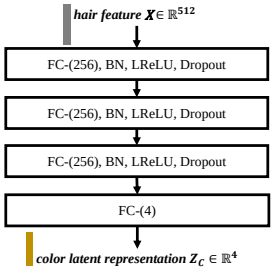


Fig. 1. Color encoder E_C

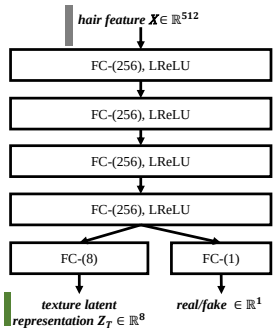
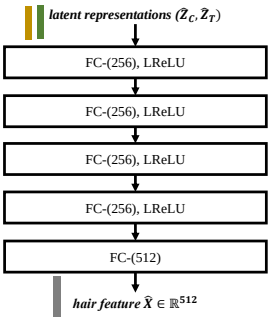


Fig. 2. The shared texture encoder E_T and the feature discriminator δ_X



Network architecture of the feature decoder D_X

Fig. 3. Feature decoder D_X for color and texture

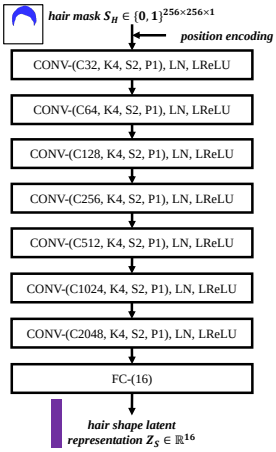


Fig. 4. Shape encoder E_S

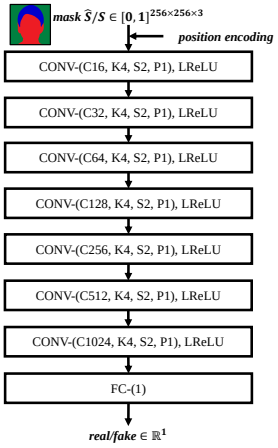


Fig. 5. Mask discriminator δ_S

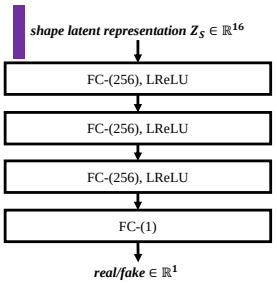


Fig. 6. Discriminator δ_{Z_S} of shape latent representation

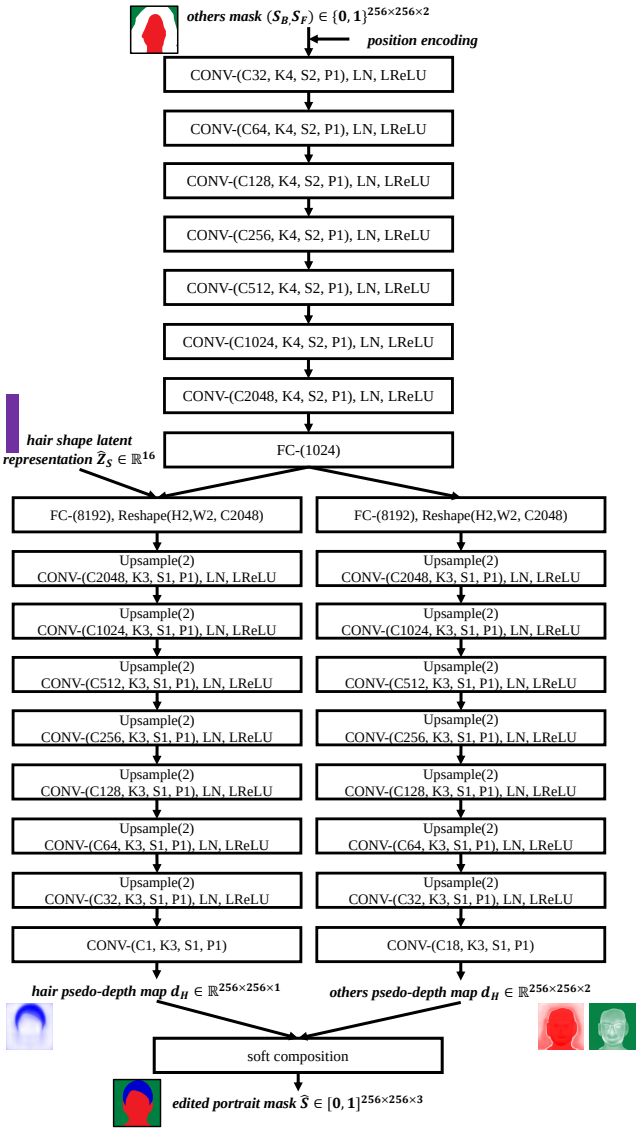


Fig. 7. Shape adaptor Γ

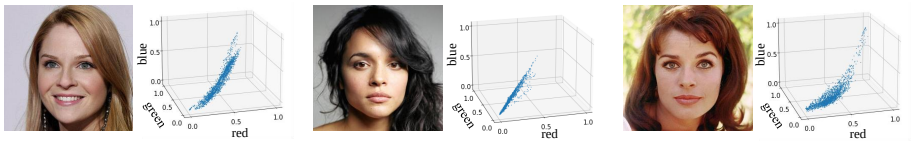


Fig. 8. The scatter plot of hair pixels for each image

B Details about Automatic Hair Color Annotation

In Section 3.2 of our paper, we briefly describe that the latent representation of hair color is designed as a 4-dimensional vector Z_C, indicating the major HSV values and color variance. They are automatically calculated as below, with no need for manual labeling.

For *HSV color values*, as denoted in our paper, the mean of pixels in the hair region is used. Besides mean, we also try median and mode, but their effects are slightly worse than the mean. For *color variance*, we observe that for most images, all pixels of hair region are roughly distributed on a one-dimensional manifold in RGB space for a single portrait as shown in Fig. 8. Based on this observation, for each image, we employ PCA to calculate its first principal component (containing about 97.0% energy of hair color pixels on average) and take the variance corresponding to the first principal component as the color variance.

C Details about Hair Texture Editing guided by Binary Annotation

As denoted in Section 3.2 of our paper, unsupervised training of hair texture disentangling and editing is able to achieve continuous and fine editing of different dimensions of the texture. However, each dimension of the learned latent representation Z_T does not necessarily correspond to an explicit semantic meaning. This is because the texture hardly obtains continuous label for supervised training, which is quite different from the color that is easy to obtain a continuous label.

However, sometimes binary annotation (which is discrete and weak) can be obtained, such as wavy or straight. With this kind of weak label, our method is also easily compatible for better editing by introducing an additional binary classification loss like that in [4], detailed in the following.

Taking the curliness factor as an example, a small number of samples are binarily labeled, including 1180 images labeled as wavy and 727 images labeled as straight. Specifically, one dimension in the texture latent representation Z_T is used to represent curliness, denoted as Z_T^{cur}, and the rest 7 dimensions of Z_T remain unchanged, i.e. still in an unsupervised manner. In addition to adopting the constraints indicated in Eq. (9), (10), (11) of our paper for texture, additionally, a binary classification loss is introduced for curliness.

Firstly, a binary label $\hat{Y}$ is assigned for the edited latent representation $\hat{Z}_T^{cur}$, i.e. $\hat{Y} = \mathbb{I}(\hat{Z}_T^{cur} \geqslant 0)$. Then, the decoded feature $\hat{X}$ is input to a pre-trained binary classifier P to get its prediction. After that, the classification loss between the estimated prediction and the binary label is calculated as follows:

$$\min_{D_X} \mathcal{L}_{T,cur}^{rec} = \mathbb{E}_{\hat{Z}_C, \hat{Z}_T \sim \mathcal{N}(0,I)} \left[-\hat{Y} \log P(\hat{X}) \right], \tag{1}$$

where P is the pre-trained binary classifier for curliness Y from the hair feature X trained by using truly labeled data. This loss can enforce that the change of latent representation is consistent with the binary label, which roughly guides the learning of continuous editing ability for curliness.

D Comparison with GAN Inversion-based Editing Methods

General GAN inversion-based editing methods, such as [1,2,3,7], also have the ability of continuous editing of portrait images and show compelling results. However, these methods are not quite the same as ours in terms of tasks. Usually for hair editing, only those editing that can manipulate the semantic variation of hair such as hair shape, structure, etc., are considered meaningful. The general GAN inversion-based editing methods can only achieve continuous editing, but not meaningful editing. BarberShop [10] and LOHO [8], also as GAN inversion-based methods, are designed specifically for hair, which is meaningful but loses continuity as discussed in the text of our paper. In one word, on premising meaningful editing, our method is continuous while existing methods are not.

E Details about Testing of Continuous Editing Ability

In section 4.3 and Fig. 8 of our paper, we test the ability of continuous editing, specifically, taking saturation of color, length of shape, and volume of shape as three examples. In testing, rough true values of the three attributes are used as criteria for evaluation. Here we introduce how these values are calculated in detail.

Saturation of Color: The mean of the saturation of all pixels in the hair region is directly used to represent the saturation of hair color.

Length of Shape: The length is mainly reflected in the vertical distance of the hair shape. In order to estimate the length and avoid being too sensitive to outliers, we use the standard deviation of the vertical axis coordinates of all pixels in hair region to reflect the length of hair, i.e. the larger the deviation is, the longer the hair is.

Volume of Shape: The hair volume reflects the amount of hair, which is reflected by the ratio of hair area and the whole image to roughly approximate hair volume.

The calculation of the three criteria is not exactly the same as the ground truth, but it is certainly proportional to the ground truth. So the relationship

Table 1. Summary of attributes and fine-grained factors of our method

Attributes	Color	Texture	Shape
Dimensions	4	8	16
Fine-grained Factors	Hue Saturation Brightness Variance	Curliness Smoothness Hair-strand-thickness	Length Volume Bangs Direction

between the above criteria and the corresponding latent representation in Fig. 8 of our paper can show whether continuous editing ability is correctly achieved.

F More Visual Results

In total, our method can edit 3 attributes (i.e. color, texture, and shape) of hair, with 28 controllable dimensions. Among them, 11 dimensions are observed to be with obvious semantics as shown in Table 1. Besides, more visual results are shown for a better understanding of our method.

Fig. 9 shows a group of editing examples by using CtrlHair's Demo. These results illustrate that our method is convenient and friendly for user interaction by sliding bars.

Fig. 10 shows more comparison results with existing methods on hairstyle transfer with a reference image. From the figure, the effect of our method on color and texture is comparable to other methods. While our method is obviously superior to MichiGAN [9] and LOHO [8] in shape transferring.

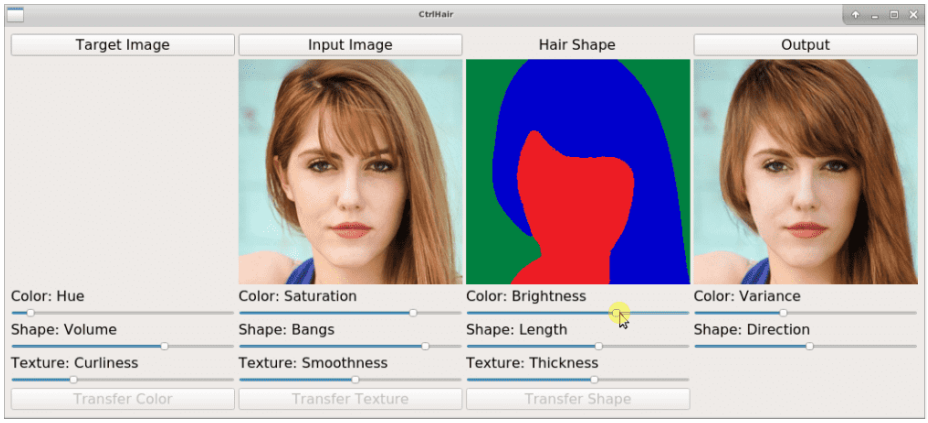
Fig. 11 shows more editing of fine factors of all attributes simultaneously and separately from our CtrlHair. Fig. 12∼22 show more editing results of each fine factor of attributes. These figures illustrate the ability of fine and continuous editing of our method.

Finally, Fig. 23 shows the result of hair editing with a wide variety for a given person, which illustrates the arbitrariness of style achieved by our method.

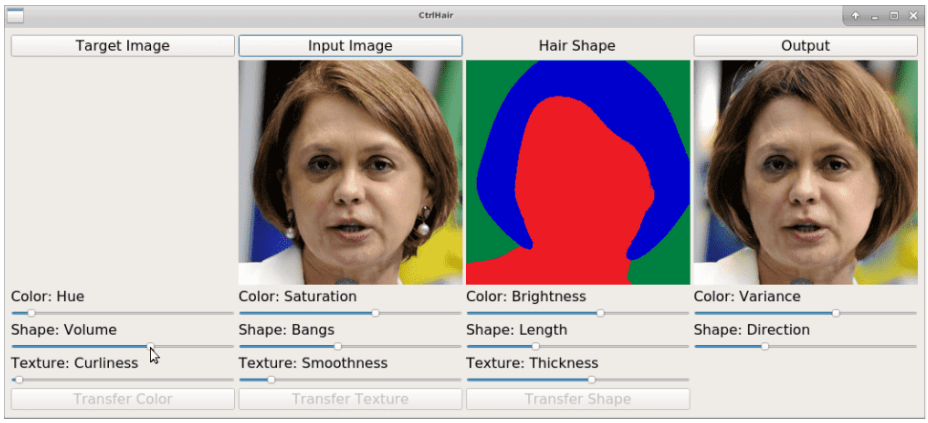
References

1. Abdal, R., Qin, Y., Wonka, P.: Image2stylegan: How to embed images into the stylegan latent space? In: IEEE/CVF International Conference on Computer Vision (ICCV) (2019)
2. Abdal, R., Qin, Y., Wonka, P.: Image2stylegan++: How to edit the embedded images? In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2020)
3. Alaluf, Y., Patashnik, O., Cohen-Or, D.: Restyle: A residual-based stylegan encoder via iterative refinement. In: IEEE/CVF International Conference on Computer Vision (ICCV) (2021)
4. Kaneko, T., Hiramatsu, K., Kashino, K.: Generative adversarial image synthesis with decision tree latent controller. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)

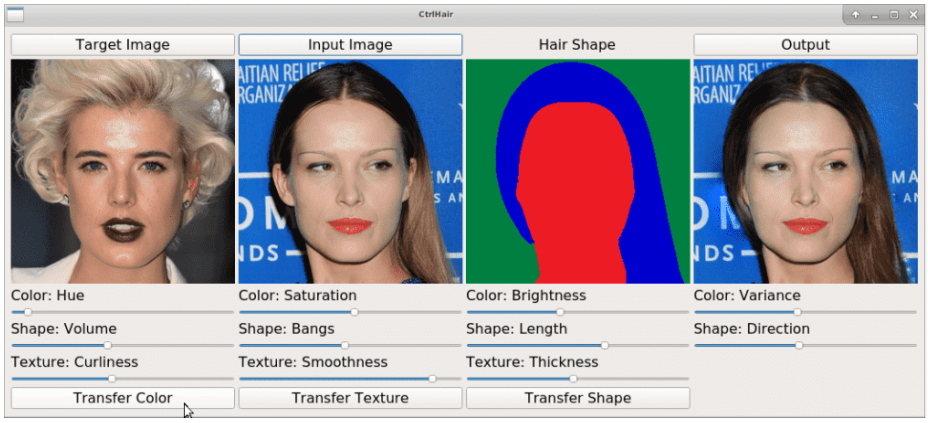
5. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
6. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In: European Conference on Computer Vision (ECCV) (2020)
7. Richardson, E., Alaluf, Y., Patashnik, O., Nitzan, Y., Azar, Y., Shapiro, S., Cohen-Or, D.: Encoding in style: a stylegan encoder for image-to-image translation. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2021)
8. Saha, R., Duke, B., Shkurti, F., Taylor, G.W., Aarabi, P.: Loho: Latent optimization of hairstyles via orthogonalization. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2021)
9. Tan, Z., Chai, M., Chen, D., Liao, J., Chu, Q., Yuan, L., Tulyakov, S., Yu, N.: Michigan: Multi-input-conditioned hair image generation for portrait editing. ACM Transactions on Graphics (TOG) **39**(4), 95 (2020)
10. Zhu, P., Abdal, R., Femiani, J., Wonka, P.: Barbershop: Gan-based image compositing using segmentation masks. ACM Transactions on Graphics (TOG) **40**(6), 215:1–215:13 (2021)



(a) Editing color



(b) Editing shape and texture



(c) Hybrid Editing of transfering and fine-tuning with sliding bars

Fig. 9. A group of editing examples using CtrlHair's Demo with User Interface. Please click on the images to open the animation

Fig. 10. Comparison with existing methods on hairstyle transfer with a reference image

Fig. 11. Editing fine factors of attributes simultaneously and separately. For a given portrait in the upper left corner of each subfigure, CtrlHair can edit the hair by sliding a set of bars

Fig. 12. Continuous editing of hue of color

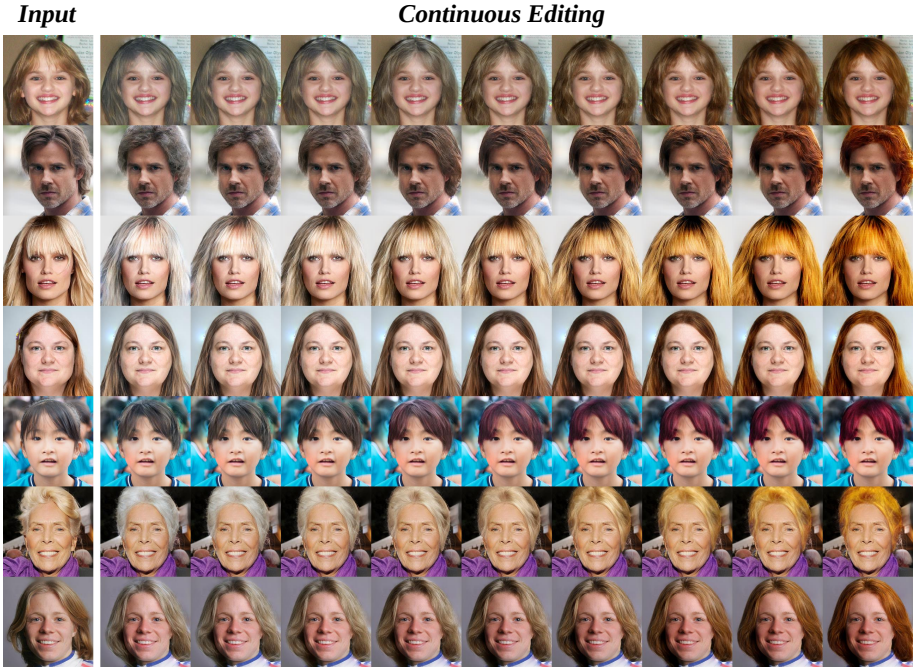


Fig. 13. Continuous editing of saturation of color

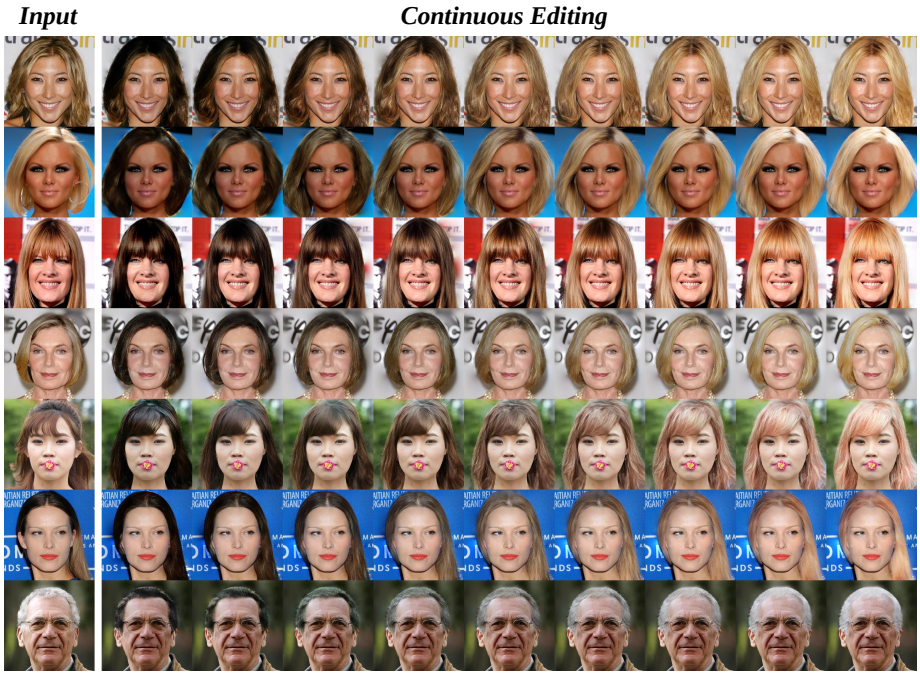


Fig. 14. Continuous editing of brightness of color

Fig. 15. Continuous editing of variance of color

Fig. 16. Continuous editing of curliness of texture

Fig. 17. Continuous editing of smoothness of texture

Fig. 18. Continuous editing of hair-strand-thickness of texture

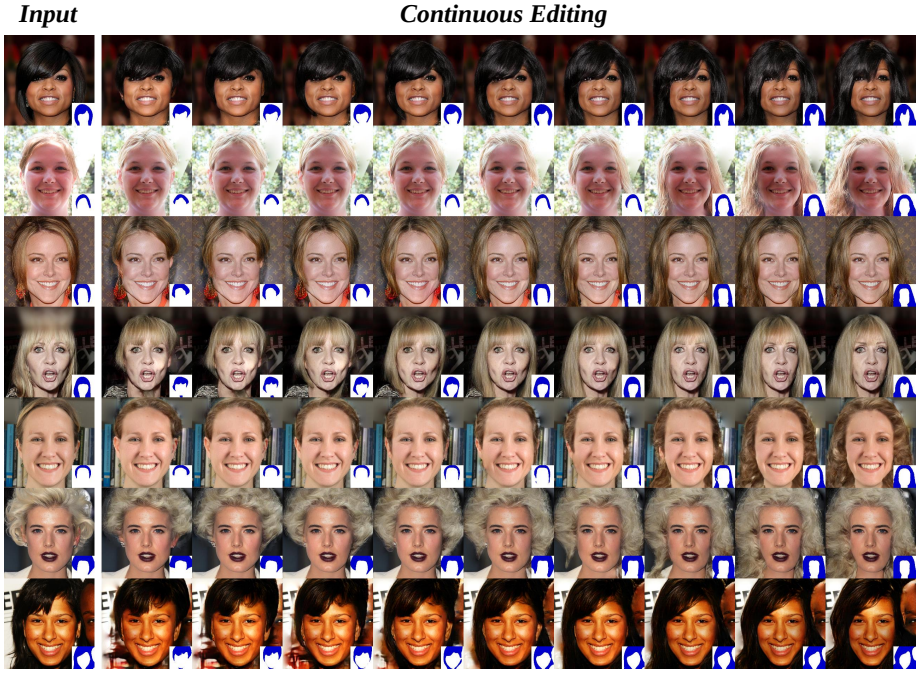


Fig. 19. Continuous editing of length of shape

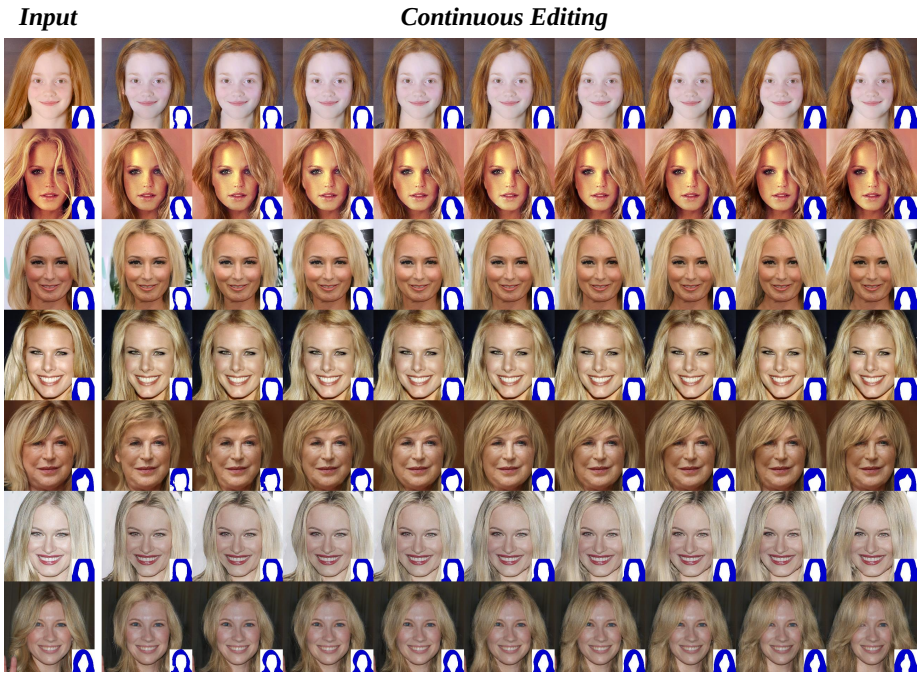


Fig. 20. Continuous editing of volume of shape

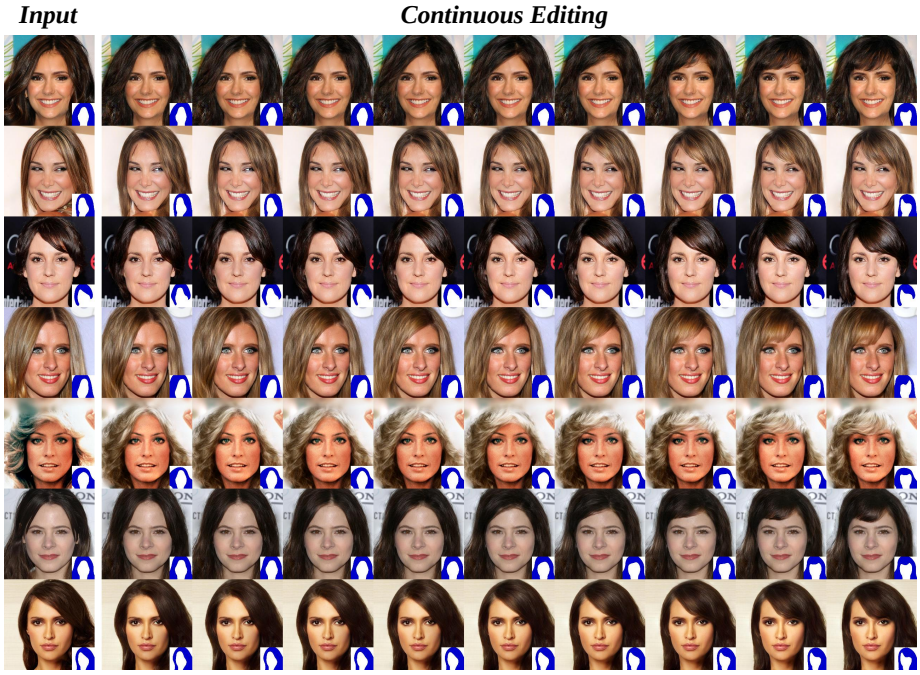


Fig. 21. Continuous editing of bangs of shape

Fig. 22. Continuous editing of direction of shape

Fig. 23. A wide variety of hair is obtained for the person in the red box