Surrogate Gradient Field for Latent Space Manipulation
Supplementary Material

Minjun Li* Yanghua Jin* Huachun Zhu
Preferred Networks
{minjunli, jinyh, zhu}@preferred.jp

1. Simple Optimization on Latent Code

In this section, we demonstrate that simple latent code optimization fails to alter the attributes of a given image. As the most straightforward method to manipulate the latent space, latent code optimization first calculates the difference between the current attributes and some desired attributes, and then backpropagates the error to the latent vector. As we update the latent vector in each step to minimize the difference, we expect that the optimized result should possess the desired attributes.

Specifically, we use the following setting for latent code optimization. Given original latent vector z_0, its corresponding attributes c_0 and the target attributes c_1. We set the initial value $z = z_0$, and optimize z via back-propagation to get the result $z_1 = \arg\min_z ||C(G(z)) - c_1||_2$. We use the Adam [4] optimizer with learning rate set to 0.0002.

As shown in Figure 1, latent code optimization unfortunately does not modify the image as expected. We hypothesize that the high degree of non-convexity of the composite function $C \circ G$ leads to this weird behavior. As a result, gradient-based optimization easily gets stuck in local optima. A good example of such a local optimum is the face image in the middle of Figure 1 which C classifies as a female image.

Another limitation of latent code optimization is that we need to back-propagate $C \circ G$, which may not be possible. For example, in our Flower-Caption experiments, C is an image captioner followed by a sentence embedding network. The step of beam search in the caption generation makes it difficult to back-propagate through C.

Our method does not suffer from the above two limitations. We construct a surrogate gradient field of $C \circ G$ to avoid local minima. Also, we evaluate $C \circ G$ only in the forward direction, thus avoiding the need to back-propagate $C \circ G$.

2. Implementation Details of F

The auxiliary mapping F consists of N layers of conditional linear block, as shown in Figure 2. AdaIN represents adaptive normalization introduced in [1]. We add a LeakyReLU operation after each AdaIN operation. The dimension of both hidden features and output features are 512. The length of each latent vector z is 512 in all experiments, while the length of condition c depends on the experimental settings. The length of condition c is 48 in FFHQ-Attributes and CelebAHQ-Attributes experiments, 120 in Anime-KeypointsAttr (70 for facial landmarks and 50 for facial attributes), and 768 in Flowers-Caption.

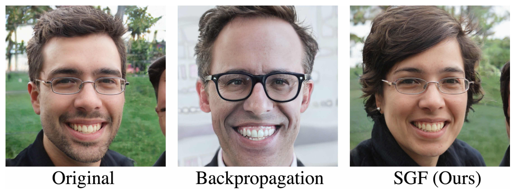


Figure 1: **Latent code optimization fails to change the gender of the input. Left:** The original face as input. **Middle:** The result of latent code optimization. We notice that the classifier predicts this face as female. **Right:** The result of our method.

3. Evaluation Details

Table 1 shows the full attributes list for our facial attributes predictor. The attributes from No.0 to No.18 are trained using Azure Face API predicted images. The attributes from No.19 to No.47 are trained using CelebA [5] dataset. We use all attributes in the experiments on FFHQ and CelebA datasets.

As we mentioned in the main paper, every manipulation methods have its specific way to control the manipulation intensity, *e.g.* adjusting the length of the vector to apply to control the intensity in InterfaceGAN. Users are required to fine-tune the strength of movement along the manipulation path. We find that some methods tend to over-modify the image when increasing strength related hyper-parameters, which results in more entangled outputs. These make it difficult to align the magnitude of editing for each algorithm to

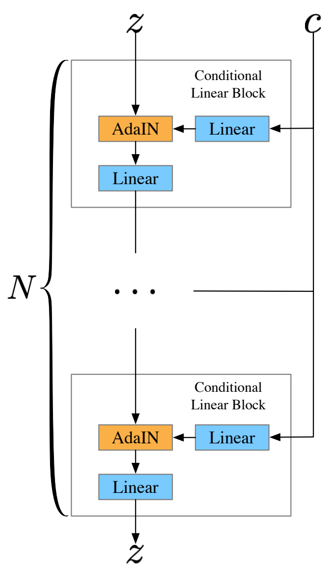


Figure 2: **The network architecture of the auxiliary mapping network F.** We use $N = 6$ blocks for experiments on Z-Space, $N = 15$ blocks for experiments on W-Space. Each conditional linear block contains a fully-connected layer, followed by a AdaIN and a LeakyReLU activation. c is used to calculate the parameters of AdaIN.

make them fairly comparable. Thus we design the Manipulation Disentanglement Score as a strength-agnostic metric for comparing the disentanglement of image manipulation algorithms. Table 2 shows detailed evaluation results of Manipulation Disentanglement Score on "gender" attribute under the FFHQ-Attributes settings.

4. Additional Results on FFHQ-Attributes

Figure 5 shows the additional results of attributes manipulation in FFHQ-Attributes dataset. The first two rows show results of editing facial orientation by using different values of "yaw". The second two rows show continuous editing results of "age". Finally, the last two rows are results of sequential editing using SGF. In Figure 6, we also show some samples used in our user study.

5. Additional Results on CelebAHQ-Attributes

The MDCs of CelebAHQ-Attributes dataset are shown in Figure 7(a). Green circles highlight the image that has the highest harmonic mean of accuracy and disentanglement along the curve.

We show more comparisons in Figure 7(b) to illustrate the effect of different hyper-parameters on the results of each method. For SGF, the hyper-parameter refers to the max step number n, while for InterfaceGAN it is the magnitude of displacement in the direction of condition. Both can be interpreted as the process of increasing the intensity of manipulation. Green boxes highlight the results that use the corresponding highlighted hyper-parameters in Figure 7(a). As we can see, similar to the results in FFHQ-Attributes, our method shows higher disentanglement on each attribute, changing the target attribute while keeping other attributes intact during the manipulation process.

Figure 7(c) shows the results of continuous attribute adjustment. The first row shows results of gradual adjustment

Table 1: **Attributes list for our facial attribute predictor.**

No.	Attribute
0	Age
1	Gender
2	Smile
3	Glasses
4	Bald
5	Head_Roll
6	Head_Yaw
7	Head_Pitch
8	Beard
9	Moustache
10	Sideburns
11	Happiness
12	Neutral
13	Brown_Hair
14	Black_Hair
15	Blond_Hair
16	Red_Hair
17	Gray_Hair
18	Other_Hair_Colors
19	5_o_Clock_Shadow
20	Arched_Eyebrows
21	Attractive
22	Bags_Under_Eyes
23	Bangs
24	Big_Lips
25	Big_Nose
26	Blurry
27	Bushy_Eyebrows
28	Chubby
29	Double_Chin
30	Goatee
31	Heavy_Makeup
32	High_Cheekbones
33	Mouth_Slightly_Open
34	Narrow_Eyes
35	No_Beard
36	Oval_Face
37	Pale_Skin
38	Pointy_Nose
39	Receding_Hairline
40	Rosy_Cheeks
41	Straight_Hair
42	Wavy_Hair
43	Wearing_Earrings
44	Wearing_Hat
45	Wearing_Lipstick
46	Wearing_Necklace
47	Wearing_Necktie

Table 2: **Calculation of the MDS of "gender" attribute on FFHQ-Attribute dataset.** For each method, we increase the manipulation strength (total iteration n for SGF, vector length μ for InterfaceGAN) until the manipulation accuracy reaches 1 or accumulated MDS starts to decrease, and choose the maximum accumulated MDS as the final MDS.

	Manipulation		**Accumulated**	**Harmonic Means of**
Method	**Acc.**	**Disent.**	**MDS**	**Acc. & Disent.**
SGF , $n = 5$	0.18	0.986	0.179	0.304
SGF , $n = 10$	0.48	0.915	0.464	0.630
SGF , $n = 15$	0.79	0.890	0.744	0.837
SGF , $n = 20$	0.93	0.872	0.867	0.900
SGF , $n = 25$	0.99	0.859	**0.919**	**0.920**
SGF , $n = 30$	0.98	0.842	0.910	0.906
InterfaceGAN , $\mu = 0.25$	0.13	0.993	0.129	0.230
InterfaceGAN , $\mu = 0.5$	0.32	0.942	0.312	0.478
InterfaceGAN , $\mu = 0.75$	0.41	0.883	0.394	0.560
InterfaceGAN , $\mu = 1.0$	0.55	0.822	0.513	0.659
InterfaceGAN , $\mu = 2.0$	0.85	0.612	0.728	**0.712**
InterfaceGAN , $\mu = 3.0$	0.99	0.469	0.804	0.636
InterfaceGAN , $\mu = 4.0$	1.00	0.398	**0.808**	0.569

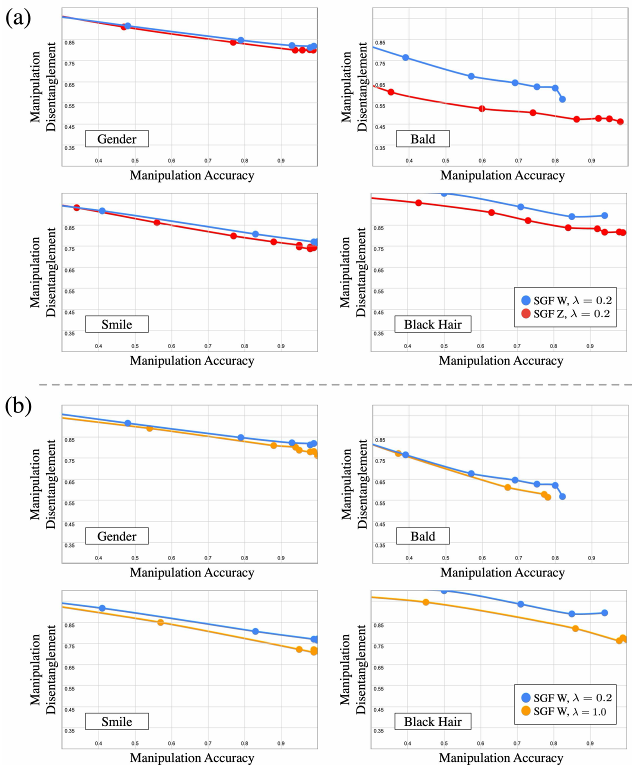


Figure 3: **The MDS of ablation study on our method in the FFHQ-Attributes dataset.** (a) The MDC of our ablation study in different latent spaces. (b) The MDC of our ablation study using different step size λ.

of the yaw attribute, and the following rows show sequential adjustments of several attributes. For each row, we observe smooth translation from the original image to the target image, which facilitates an overall more realistic editing sequence.

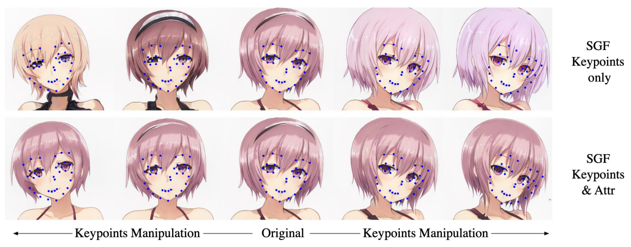


Figure 4: **Comparison on keypoint manipulation results on Anime-Keypoints and Anime-KeypointsAttr.** The first row shows the manipulation results of SGF conditioned on keypoints only. The second row shows the results conditioned on both keypoints and attributes. Additional controllablilty on attributes (e.g. hair color) ensures that they are consistent when keypoints change.

6. Additional Results on Anime-KeypointsAttr

We found that only using keypoints as the control condition could cause undesired changes in results, as shown in the first row in Figure 4. Thus the keypoints prediction is concatenated with the first 50 attributes from illustration2vec [6] classifier as the final **KeypointsAttr** condition for keypoints manipulation experiments. Since the predicted attributes in KeypointsAttr contain hair color information, adding these attributes to training conditions can alleviate the undesired changes in results. For the F trained with both keypoints and attributes (the second row in Figure 4), our method successfully maintains the hair color unchanged after the manipulation, indicating that adding conditioning variables can encourage our method to perform better disentanglement.

Figure 8 shows additional manipulation results in Anime-KeypointsAttr dataset. We use our method to edit the head poses and zoom levels of generated anime faces (columns two through five). In addition, we show some sequential editing of zoom level and head poses (last two columns).

7. Additional Results on Flower-Caption

Figure 10(a) shows additional results of Flower-Caption experiments using the same configuration as that used in the paper. Note that unlike Anime-KeypointsAttr experiments, we do not have additional conditioning variables to keep the shape of the flower unchanged when editing the color. We limit latent space manipulation to apply on the top four layers for color manipulations, the bottom four layers for shape manipulations, while imposing no limitation for editing that involves changes of both color and shape.

Table 3: **MDS comparison on FFHQ for ablation study,** evaluated using different step size λ for SGF on Z-space and W-space of StyleGAN2.

Method	Gender	Bald	Smile	Black Hair	Overall
SGF Z, $\lambda = 1$	0.857	0.519	0.846	0.886	0.777
SGF Z, $\lambda = 0.2$	0.901	0.579	0.839	0.909	0.807
SGF Z, $\lambda = 0.02$	0.353	0.039	0.258	0.140	0.198
SGF W, $\lambda = 1$	0.889	0.587	0.859	0.911	0.812
SGF W, $\lambda = 0.2$	**0.919**	**0.590**	**0.884**	**0.955**	**0.837**
SGF W, $\lambda = 0.02$	0.209	0.125	0.649	0.240	0.306

Figure 10(b) shows the results using the same inputs as Figure 10(a) without limiting the latent space manipulations to apply on specific layers in StyleGAN2. We observe that conditioning on colors using all layers might result in unwanted shape changes, and vise versa.

8. Ablation Study

We conduct an ablation study on latent space of GANs and the step size in our algorithm. Figure 3(a) shows the MDC of our method in either Z-space or W-space of the StyleGAN model trained on the FFHQ-Attributes dataset. Given the same attribute to manipulate, our model performs better in the W-space of StyleGAN2 [3] than in the Z-space. This shows that our model can benefit from a more disentangled latent space [2]. Figure 3(b) shows the MDC of our method using different step size λ. Using a smaller step size improves accuracy, resulting in a better MDC shape. However, using step sizes that are too small leads to slow convergences. Here we omit the MDC of $\lambda = 0.02$ and below because our method under such settings reachs the maximum iteration with an accuracy lower than 0.3 most of the time.

In table 3, we report quantitative results of SGF under different hyper-parameter settings on FFHQ-Attributes dataset. We set the max iteration number $n = 50$ and examine performances under different step sizes λ. We find that decreasing the step size from 1 to 0.2 could result in a better performance. However, a further decrease to 0.02 significantly slows down the convergences, thus shows inferior results in MDS. We also test performances on Z-space, we observe performance drop for all step sizes comparing with W-space. This suggests that our method can benefit from using a better disentangled latent space.

9. Non-Linear Path of SGF

Compared with previous approaches, our SGF model has non-linear, position-variant properties when manipulating latent codes. Figure 9(a) shows the editing path of SGF and its linear interpolation on mouth keypoints editing. Specifically, the non-linear results are obtained by setting different iteration limit n on SGF, while the linear interpolation is interpolation on latent space from the starting w_0 to final w_0' calculated by SGF. We notice that the linear interpolation makes a close approximation to the original non-linear path.

Applying the "mouth open" direction $(w_0' - w_0)$ to another latent point also achieves similar results to the non-linear path from SGF, as shown in Figure 9(b). However, such transferability does not apply to every manipulation. As shown in Figure 9(c), linearly applying the "eyes closed" direction obtained from SGF to other latent points generates results inferior to the original non-linear results. Although both linear manipulation operations close the eyes of anime charactors, they also introduce unwanted mouth manipulation (row 3) and unnatural editing (row 4) in the final results.

Overall, we can simplify the control direction obtained by SGF to linear control without making significant sacrifices, and in some cases, such linear direction can be applied to other samples. To ensure making precise and disentangled modification on given results, however, needs to rely on the non-linear path of SGF.

Table 4: **Running time of SGF on different datasets (in sec.)**. The running time of SGF depends on the re-estimation step $c_i = C(G(z_i))$, using a larger generator and condition predictor would result in a longer running time.

Method	SGF	SGF (fast ver.)
FFHQ-Attributes	2.89	0.304
Anime-KeypointsAttr	6.54	0.332
Flower-Captions	11.73	0.366

10. Running Time of SGF

Table 4 shows the running time (in second) of SGF on different settings, averaged from 100 samples. The running time of SGF (*i.e.*, running time of Algorithm 1) largely depends on the running time of inferencing condition of new latent code in each step $c^{(i)} = C(G(z^{(i)}))$. We observe faster running time in FFHQ-Attributes experiments, which uses a facial attributes classifier that has a much simpler structure compared to the keypoint attribute predictor in Anime-KeypointsAttr. The C in Flower-Captions consists of an image captioner and a sentence embedding encoder, making the overall running time much longer than other settings.

A trick to save time when running Algorithm 1 is to replace the inference step $c^{(i)} = C(G(z^{(i)}))$ by $c^{(i)} = i\delta_c$ after the first inference. As a result, the Algorithm 1 runs at approximately a constant 0.4 second, shown as SGF (fast ver.) in Table 4. This trick would make some scarification on manipulation performance due to the estimation error of Auxiliary Mapping F.

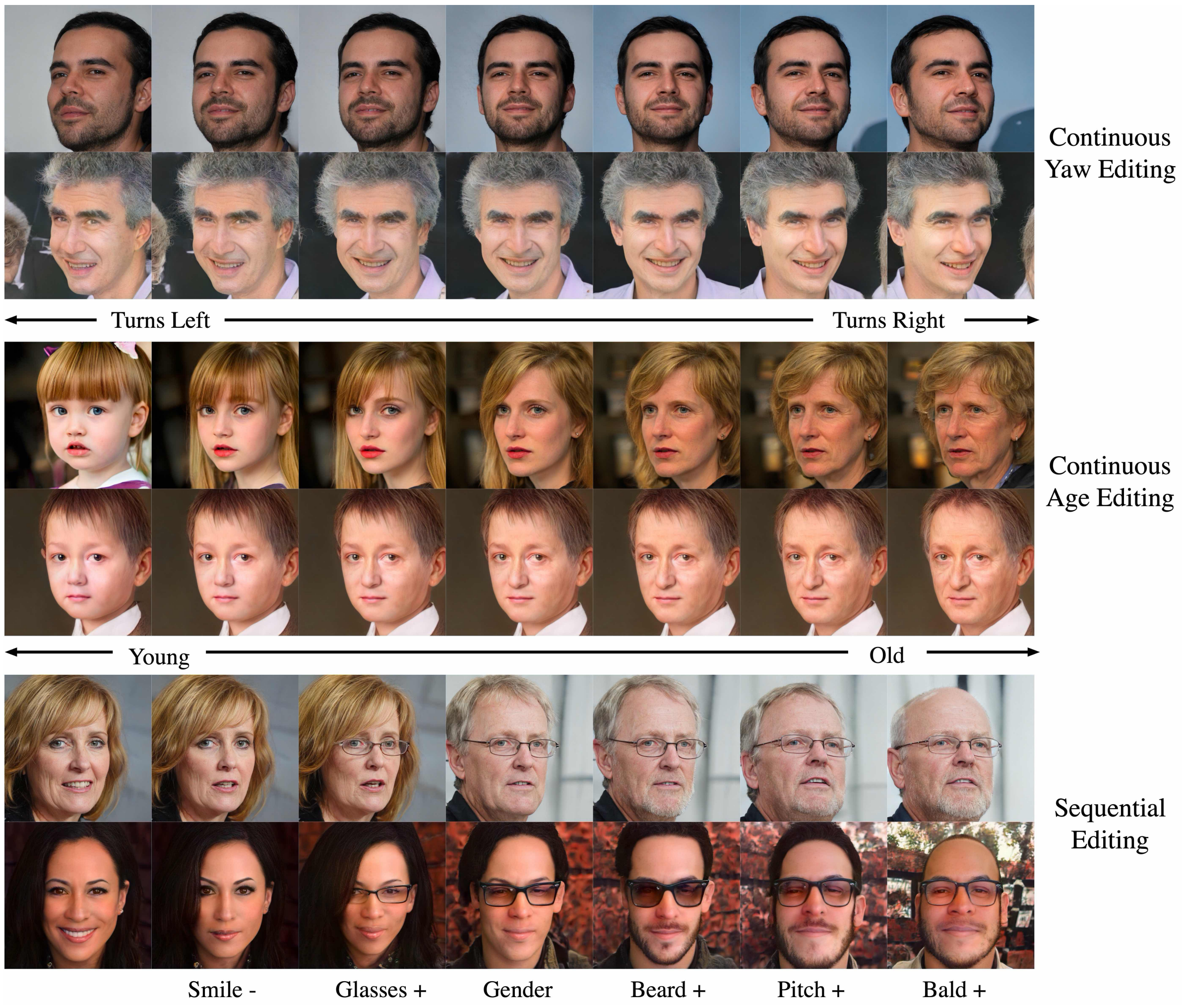


Figure 5: **Additional results on the FFHQ-Attributes dataset.** We edit face images generated by GANs in odd rows and edit real-life images projected to the latent space of GANs in even rows.

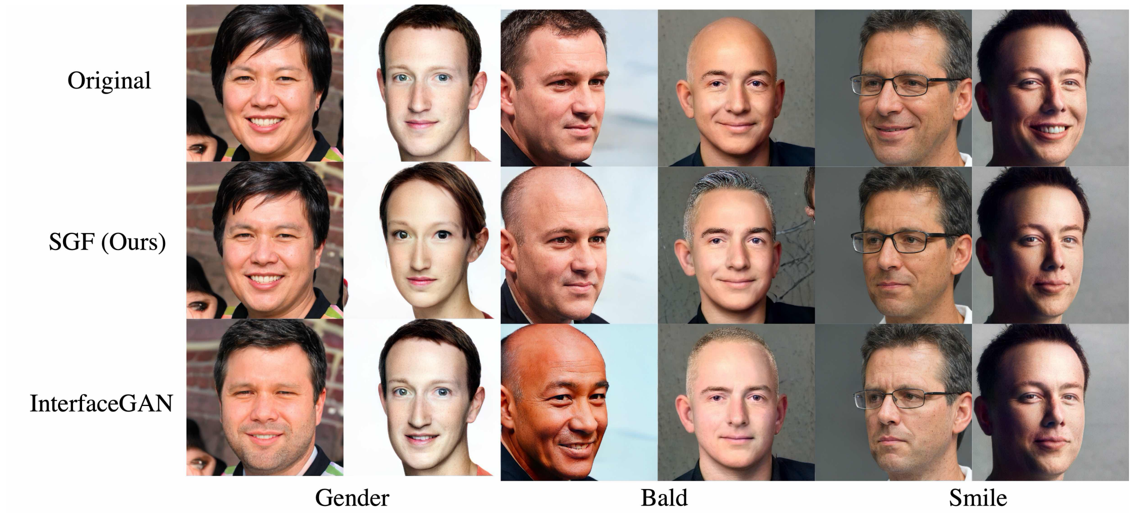
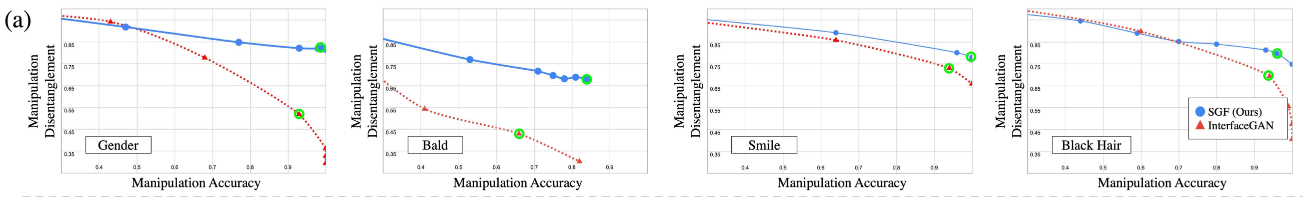
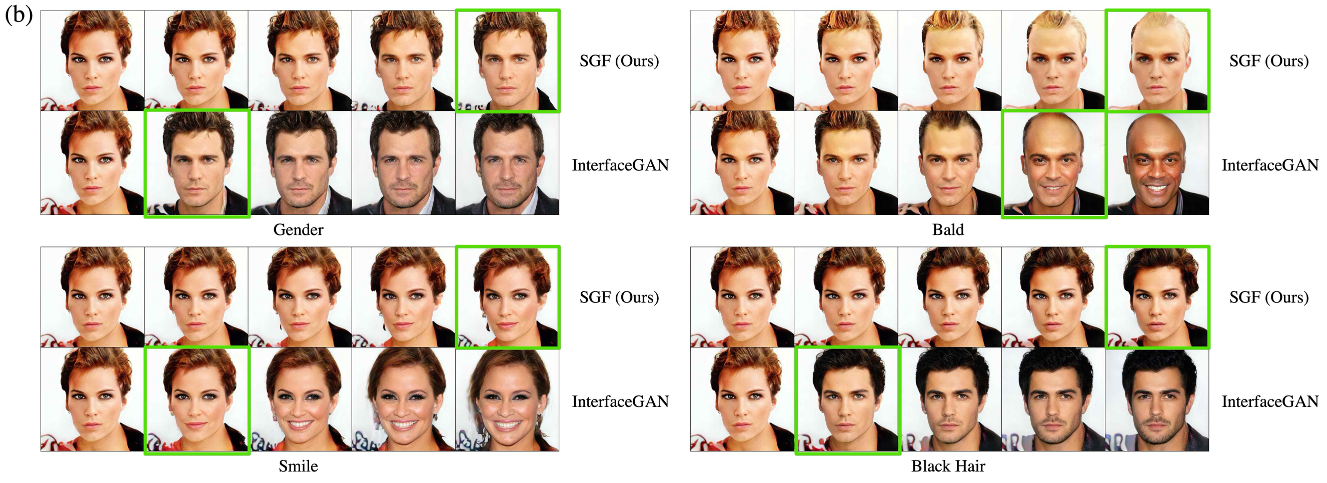


Figure 6: **Part of the samples used in the user study.** The original images in odd columns are generated by GANs while those in even columns are real-life images projected to the latent space of GANs.





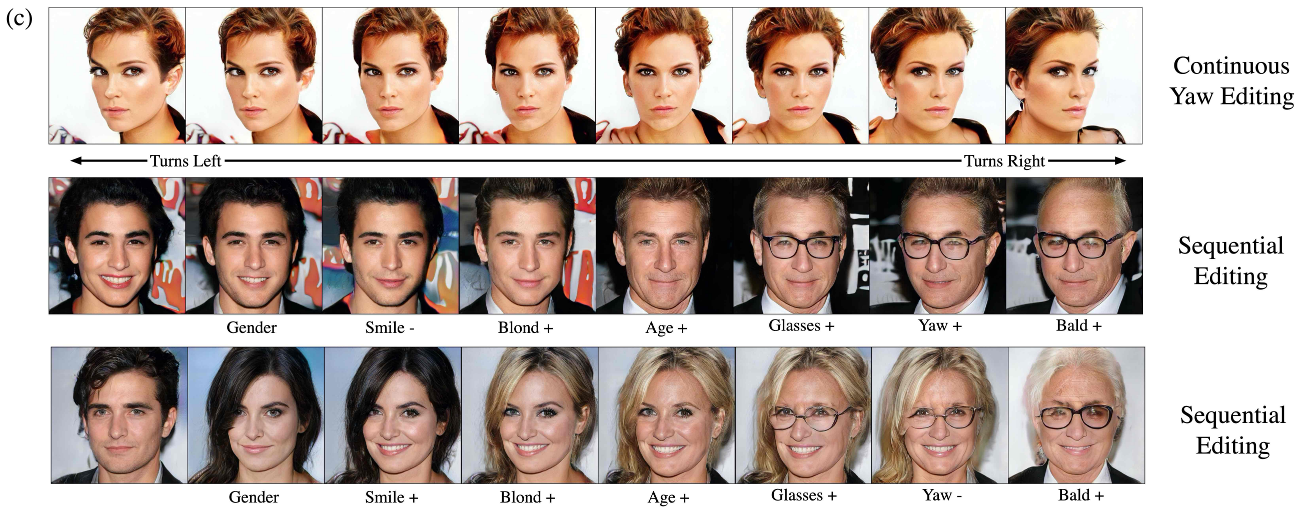


Figure 7: **Facial attribute editing results in the CelebAHQ-Attributes dataset.** (a) The MDC of InterfaceGAN and our method on several attributes. Green circles highlight the image that has the highest harmonic mean of accuracy and disentanglement along the curve. (b) Manipulation process of InterfaceGAN and our method. Green boxes highlight the images with the highest harmonic mean of accuracy and disentanglement during the manipulation. (c) More results of sequential editing using our method.

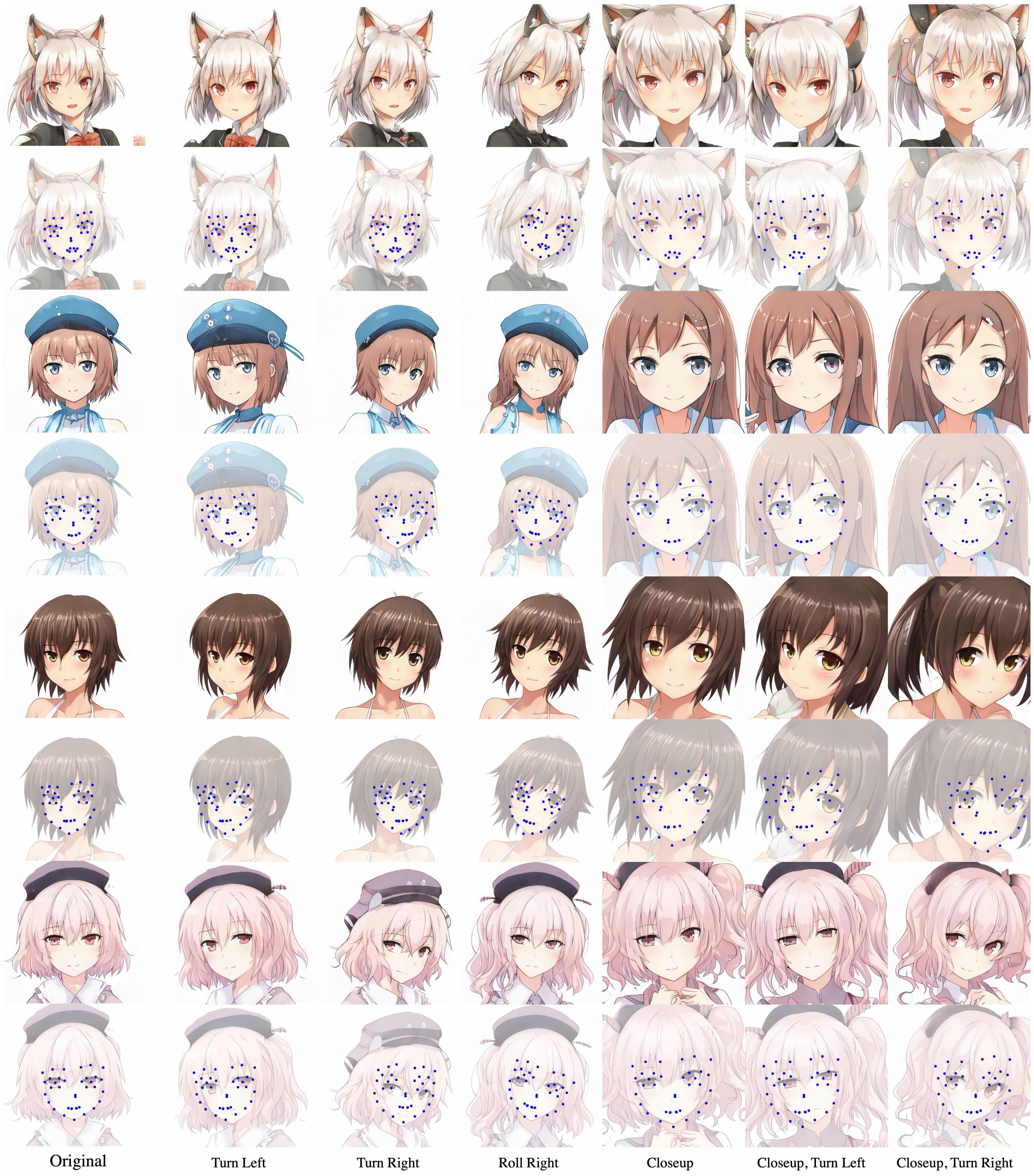


Figure 8: **Keypoints editing on the Anime-KeypointsAttr dataset.** Odd rows show the results of our keypoints manipulation and even rows are the corresponding target keypoint conditions.

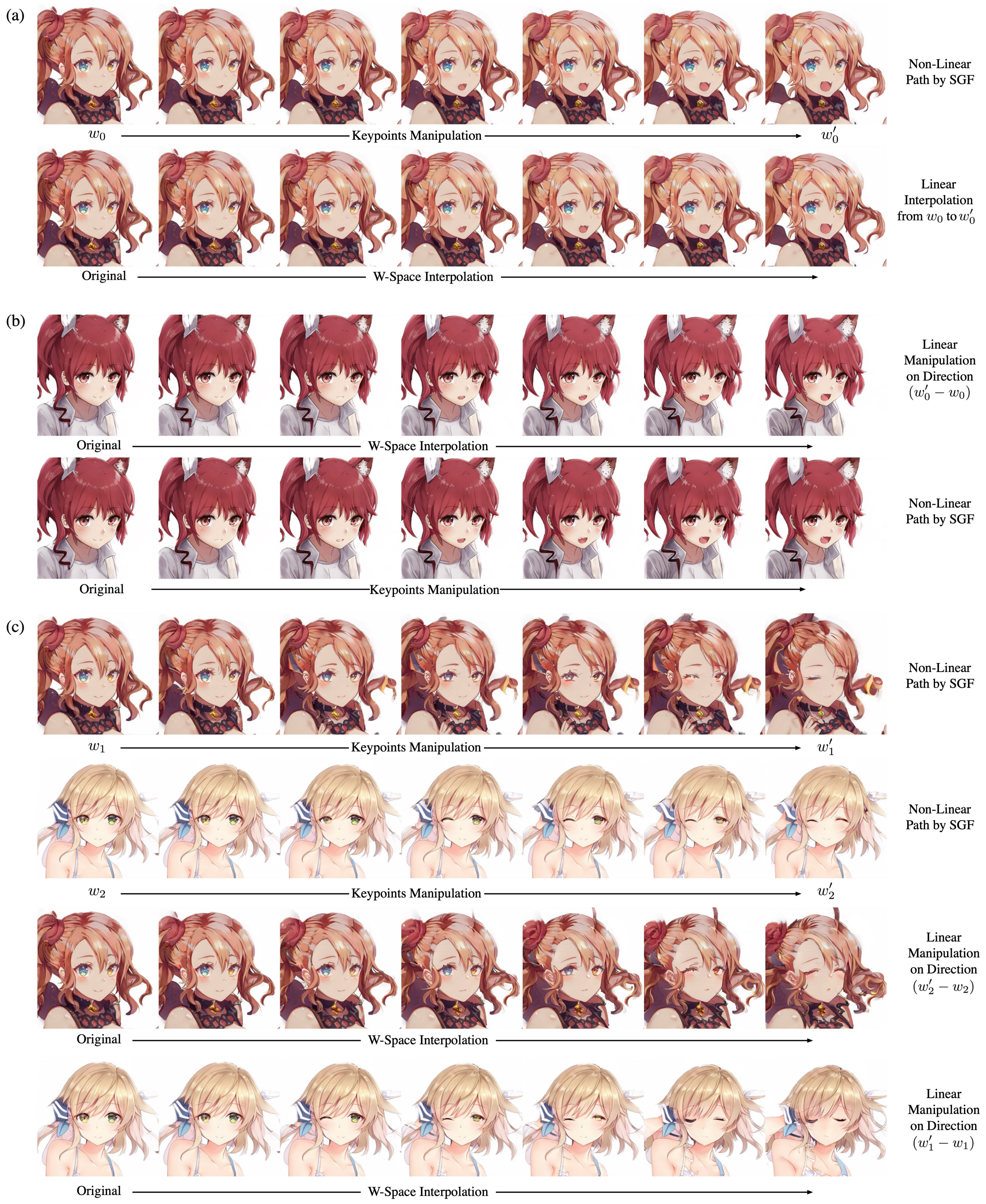


Figure 9: **Comparing the non-linear path of SGF and its linear interpolations on Anime-KeypointsAttr dataset.** (a) Comparing the non-linear path of SGF and its linear interpolations. (b) Using the manipulation direction in (a) to control another latent sample. (c) Use SGF to get the "eye closed" directions, and then exchange the manipulation direction of two latent samples. Refer to Section 9 for the details.

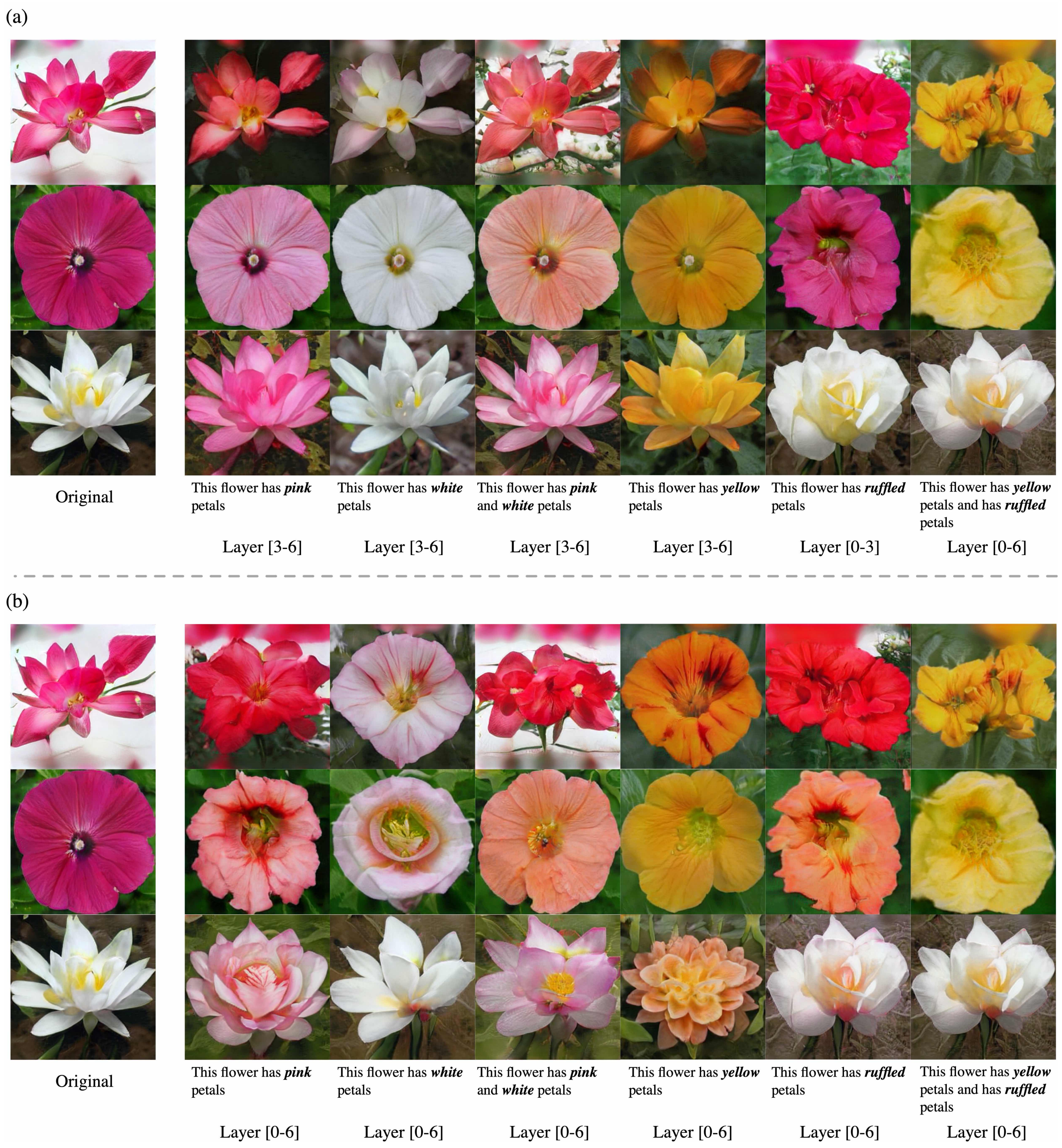

(a)
Original
This flower has ***pink*** petals
This flower has ***white*** petals
This flower has ***pink*** and ***white*** petals
This flower has ***yellow*** petals
This flower has ***ruffled*** petals
This flower has ***yellow*** petals and has ***ruffled*** petals
Layer [3-6]
Layer [3-6]
Layer [3-6]
Layer [3-6]
Layer [0-3]
Layer [0-6]
(b)
Original
This flower has ***pink*** petals
This flower has ***white*** petals
This flower has ***pink*** and ***white*** petals
This flower has ***yellow*** petals
This flower has ***ruffled*** petals
This flower has ***yellow*** petals and has ***ruffled*** petals
Layer [0-6]
Layer [0-6]
Layer [0-6]
Layer [0-6]
Layer [0-6]
Layer [0-6]

Figure 10: **Manipulation by captions in Flowers-Caption dataset.** (a) Manipulation results on Flower-Caption dataset. We limit the latent space manipulation to certain layers when editing either color or shapes, and apply to all layers when conditioning on both color and shape. The layers used for each manipulation are noted at the bottom of each column. (b) Manipulation results without layer-wise manipulation in (a).

References

[1] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of ICCV*, 2017. 1

[2] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of CVPR*, 2019. 4

[3] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of CVPR*, 2020. 4

[4] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of ICLR*, 2015. 1

[5] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of ICCV*, 2015. 1

[6] Masaki Saito and Yusuke Matsui. Illustration2vec: a semantic vector representation of illustrations. In *SIGGRAPH Asia 2015 Technical Briefs*. 2015. 3