StyleSpace Analysis: Disentangled Controls for StyleGAN Image Generation (supplementary material)

Zongze Wu
Hebrew University
zongze.wu@mail.huji.ac.il

Dani Lischinski
Hebrew University
danix@cs.huji.ac.il

Eli Shechtman
Adobe Research
elishe@adobe.com

9. Structure of StyleGAN2 StyleSpace

To supplement the description of the different StyleGAN2 latent spaces in Section 3, here we describe the structure of the StyleSpace $\mathcal{S}$ in more detail. Every major layer (every resolution) of the StyleGAN2 generator (synthesis network) consists of two convolution layers for feature map synthesis and a single convolution layer that converts the second feature map into an RGB image (referred to as tRGB), as shown in Figure 9. Each of these three convolution layers is modulated by a vector of style parameters. We denote the three different vectors of style parameters as s_1, s_2, and s_{tRGB}. These are obtained from the intermediate latent vectors $w \in \mathcal{W}$ via three affine transformations, $w_1 \rightarrow s_1$, $w_2 \rightarrow s_2$, $w_2 \rightarrow s_{tRGB}$. In $\mathcal{W}$ space, w_1 and w_2 are the same vector, and it is the same vector for all layers. In $\mathcal{W}+$ space, w_1 and w_2 are two different vectors, and every major layer has its own pair (w_1, w_2). The length of all the w vectors is 512. The numbers of style parameters used by the different layers are listed in Table 2. Note that in 4x4 resolution, there is only s_1 and s_{tRGB}. The length of s is 512 from the early layers until layer 14. After that layer, the length decreases from 256 to 32. In total, for a 1024x1024 generator, there are 6048 style channels that control feature maps, and 3040 additional channels that control the tRGB blocks.

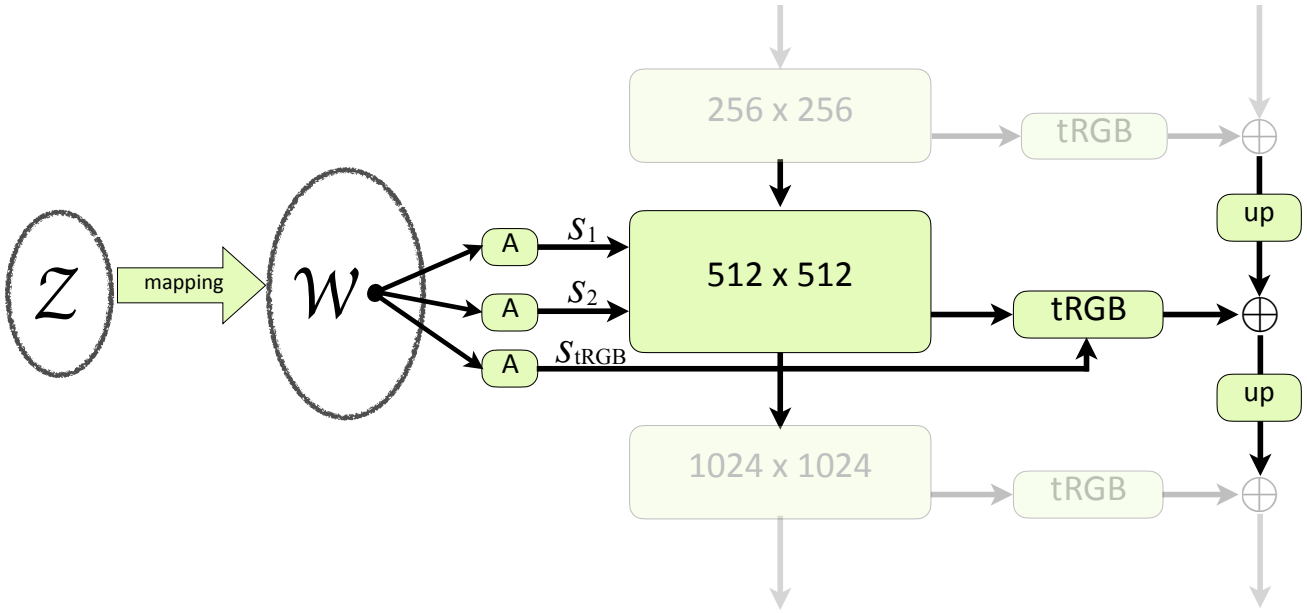


Figure 9. The internal structure of StyleSpace $\mathcal{S}$, shown for the 512×512 generator resolution.

$\mathcal{W}+$ layer index	$\mathcal{S}$ layer index	resolution	layer name	type	# channels
0	0	4×4	Conv	s_1	512
1	1	4×4	ToRGB	s_{tRGB}	512
2	2	8×8	Conv0_up	s_1	512
3	3	8×8	Conv1	s_2	512
3	4	8×8	ToRGB	s_{tRGB}	512
4	5	16×16	Conv0_up	s_1	512
5	6	16×16	Conv1	s_2	512
5	7	16×16	ToRGB	s_{tRGB}	512
6	8	32×32	Conv0_up	s_1	512
7	9	32×32	Conv1	s_2	512
7	10	32×32	ToRGB	s_{tRGB}	512
8	11	64×64	Conv0_up	s_1	512
9	12	64×64	Conv1	s_2	512
9	13	64×64	ToRGB	s_{tRGB}	512
10	14	128×128	Conv0_up	s_1	512
11	15	128×128	Conv1	s_2	256
11	16	128×128	ToRGB	s_{tRGB}	256
12	17	256×256	Conv0_up	s_1	256
13	18	256×256	Conv1	s_2	128
13	19	256×256	ToRGB	s_{tRGB}	128
14	20	512×512	Conv0_up	s_1	128
15	21	512×512	Conv1	s_2	64
15	22	512×512	ToRGB	s_{tRGB}	64
16	23	1024×1024	Conv0_up	s_1	64
17	24	1024×1024	Conv1	s_2	32
17	25	1024×1024	ToRGB	s_{tRGB}	32

Table 2. Breakdown of StyleSpace channels by generator layers.

10. Effect of style parameters in tRGB layers

To examine the function of style parameters that control the tRGB layers, we randomly generate a set of 500K style vectors $s \in \mathcal{S}$, and perturb their s_{tRGB} channels to manipulate the tRGB layers, $s_{new} = s_{original} + n\sigma(s)$. $\sigma(s)$ is the standard deviation of each channel of s over the generated set, used to normalize the amount of perturbation across different channels [3]. n is a vector of Gaussian noise, with mean 0 and standard deviation $\sigma(n)$, which indicates the manipulation strength. Below, we use $\sigma(n) = 15$.

As shown in Figure 10, manipulating the early (coarse) resolutions (0,1,2) mainly affects the center of the target object (better visible in faces than in cars), manipulating the middle resolutions (3,4,5) typically affects the entire target object, and manipulating the late (fine) resolution layers (6,7,8) affects the entire image. (The LSUN Car model reaches only up to 512×512, thus the fine resolution layers are (6,7)). The effect of the late (fine) resolution layers on the image is significantly stronger than that of the early and middle layers. The manipulations only affect color, without modifying shape or specific face or car related attributes.

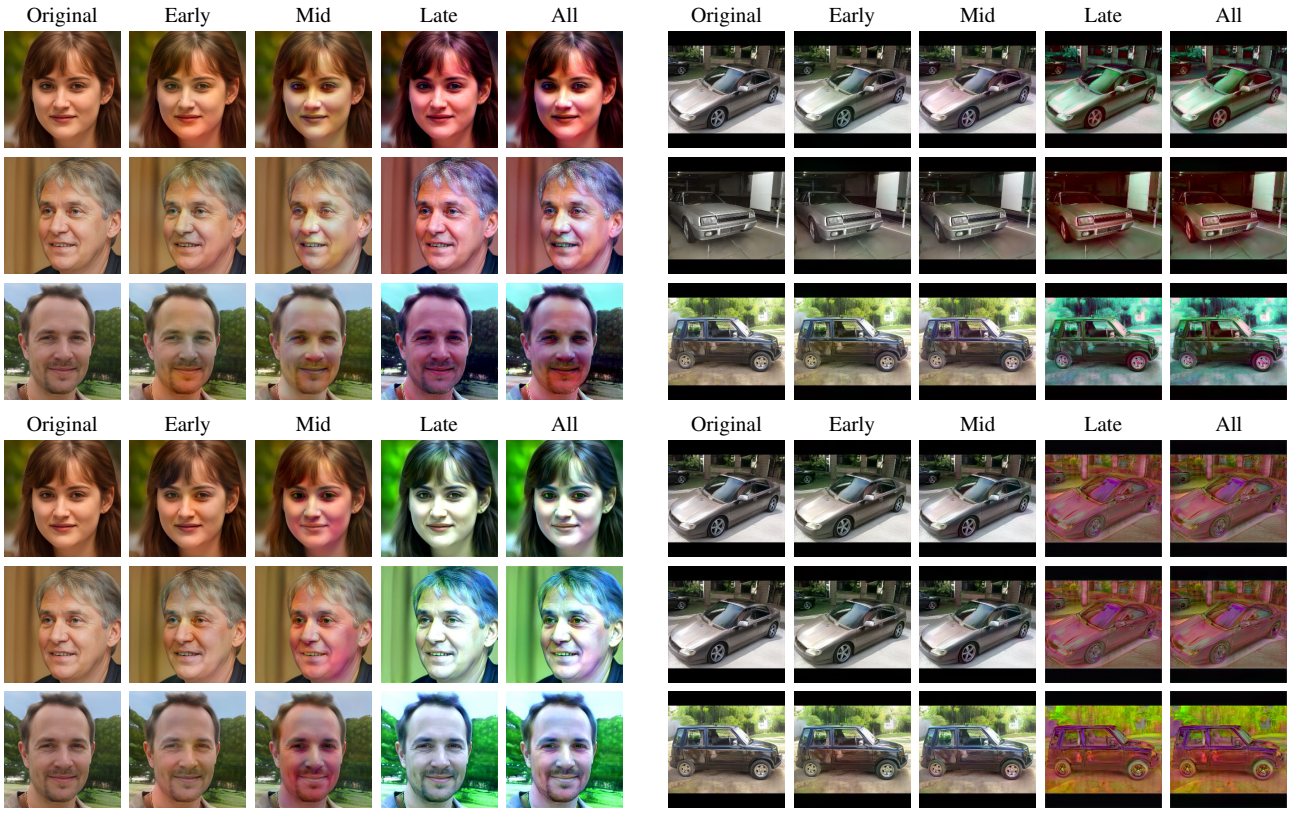


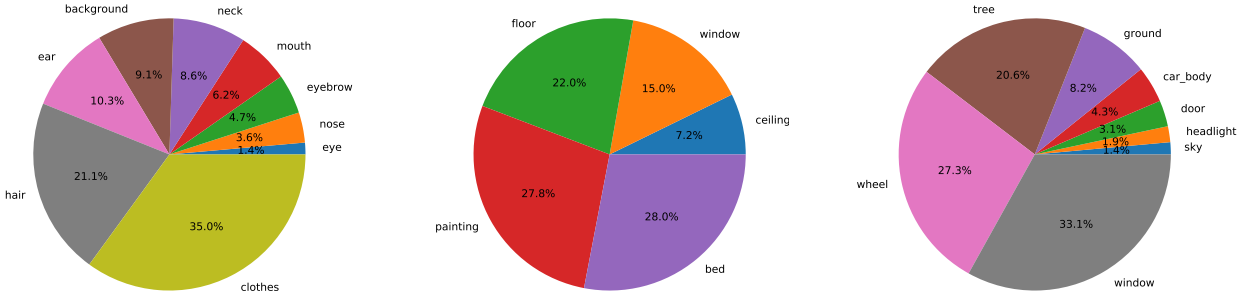
Figure 10. Manipulation by perturbing the s_{tRGB} channels in early resolution layers (0,1,2), middle layers (3,4,5) and late layers (6,7,8). Each of the four blocks above uses the same noise vector n.

11. Locally-active style channels

The number of locally-active channels that we found using the method in Section 4 for each of the three models we experimented with is summarized in Table 3. The breakdown of these localized controls across different semantic regions is plotted in Figure 11. Not all of the detected controls correspond to semantically meaningful manipulations. While there is no way to objectively determine which manipulations are meaningful, in Table 4 we report the number of manipulations that were (subjectively) determined as meaningful by the authors, among the most highly localized controls.

	FFHQ	**LSUN Bedroom**	**LSUN Car**
Num. locally-active channels	1871	421	913
Total num. of feature map style channels	6048	5376	5952
Percent of locally-active channels	30.9%	7.8%	15.3%

Table 3. Number of locally-active channels detected in different StyleGAN2 models.



(a) FFHQ (b) LSUN Bedroom (c) LSUN Car

Figure 11. Breakdown of detected localized controls across different semantic regions.

	Top 5	Top 10	Top 20
Eyebrows	5	10	19
Hair	5	9	17
Nose	4	7	13
Mouth	4	7	11
Clothes	5	6	9
Neck	2	4	7
Eye	4	5	6
Ear	3	4	6
Background	5	10	15

Table 4. Number of meaningful controls among the top $k = 5, 10, 20$ most locally-active channels (those with the highest overlap coefficient, as defined by equation (1) in the main paper) in each semantic area (for the FFHQ model). Note that this count is subjective and may contain channels that control similar things (for example, size of lips).

Finally, Figure 12 demonstrates (in addition to Figure 1) the high degree of disentanglement of the localized controls that our method detects. Even pairs of controls that affect the same semantic region, typically do so in an independent manner.

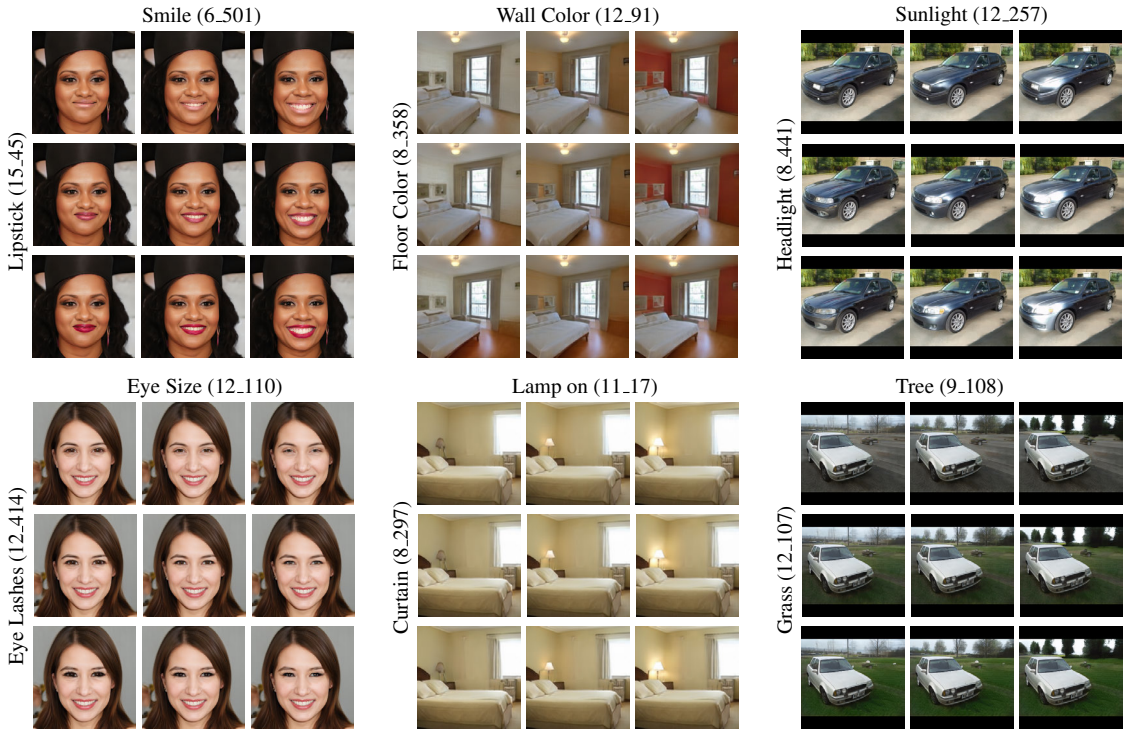


Figure 12. Disentanglement in style space, demonstrated using three different datasets (FFHQ, LSUN Bedroom, LSUN Car). Each of the six groups above shows two manipulations that occur independently inside the same image. The indices of the manipulated layer and channel are indicated in parentheses.

12. Attribute-specific channels

Starting from the 40 attributes from CelebA [4], we first remove inactivated, ambiguous and neutral attributes. Inactivated attributes (defined in Section 3) are those that are not well represented in the generated image distribution. Ambiguous attributes are highly subjective. Neutral attributes are those in between more extreme states, or attributes that are highly common across the dataset. For example, "mouth slightly open" is between an open mouth and a closed one. Another example of a neutral attribute is "no beard", since most of the faces in FFHQ don't have a beard. The attributes that were found inactivated, ambiguous, or neutral, and were removed from further consideration are listed in Table 5.

status	type	# attributes	list of attributes
removed	inactivated	9	blurry, narrow eyes, necklace, oval face, rosy cheeks, pointy noise, bald, mustache, pale skin
	ambiguous	2	attractive, heavy make up
	neutral	3	no beard, five-o-clock shadow, mouth slightly open
annotated	one or more disentangled single-channel controls found	16	gender, smiling, lipstick, eyeglasses, bangs, wavy hair, earrings, black hair, blond hair, sideburns, goatee, receding hairline, gray hair, suit (tie), double chin, hat
	no disentangled single-channel controls found	10	bags under eyes, big nose, high cheekbones, young, arched eyebrows, brown hair, big lips, bushy eyebrows, chubby, straight hair

Table 5. For 40 CelebA attributes, we first remove inactivated (9), ambiguous (2) or neutral (3) attributes. Our method is able to detect one or more disentangled single-channel controls for 16 out of the 26 remaining attributes.

To find attribute-specific controls we apply our method described in Section 5 on the remaining 26 attributes. We found that 16 out of the 26 remaining attributes are controllable by one or more single style channels, in a disentangled manner. Our method was not able to identify any *disentangled* single-channel controls for the other 10 attributes. All of the above attributes are listed in Table 5. The attributes for which no disentangled single-channel controls were found indeed appear to be correlated with other visual attributes (in FFHQ). For example, "bags under eyes" is correlated with eye size, "big lips" is correlated with skin color, and "high cheekbones" is correlated with smiling.

While our method could not find a disentangled single-channel control for the "young" attribute, it was able to find such controls for wrinkles, eyeglasses, and gray hair. Because all of these attributes are correlated with age, the "young" attribute can only be controlled by manipulating multiple style channels, rather than a single one.

Note that although the attribute-specific detection method of Section 5 could not detect a single-channel control for either "arched eyebrows" or "bushy eyebrows", our locally-active detection method in Section 4 was able to find disentangled controls for these attributes: (9,30) for arched eyebrows, and (12,325) for bushy eyebrows. Thus, these could be considered as failure cases for our attribute-specific detection method.

Table 6 lists the various attributes and the single-channel controls that were detected for them. For each control we list the layer and channel number, as well as its rank by the detection method of Section 5.

region	attribute	(layer,channel,rank)	related attributes
hair	black hair	(12,479,1)	different hair color, lighting
	blond hair	(12,479,1) (12,266,3)	gender, other hair color and style
	gray hair	(11,286,1)	glasses, gender, wrinkle and beard
	wavy hair	(6,500,1) (8,128,2) (5,92,3) (6,394,7) (6,323,28)	hair style, gender
	bangs	(3,259,1) (6,285,2) (5,414,3) (6,128,4) (9,295,8) (6,322,9) (6,487,11) (6,504,14)	hair style
	receding hairline	(5,414,1) (6,322,2) (6,497,3) (6,504,8)	hair style
mouth	smiling	(6,501,1)	size of face or eye
	lipstick	(15,45,1)	gender, face expression
beard	sideburns	(12,237,2)	other type of beard, gender
	goatee	(9,421,1)	other type of beard, gender
chin	double chin	(9,132,1)	size of neck, wrinkle
ear	earrings (entangled with gender)	(8,81,1)	gender, face shape
eye	glasses	(3,288,1) (2,175,3) (3,120,4) (2,97,6)	gender, wrinkle and beard
clothes	suit (tie)	(9,441,1) (8,292,2) (11,358,3) (6,223,11)	cloth style
hat	hat size	(5,200,7)	nothing change
overall	gender	(9,6,1)	beard,hair style

Table 6. List of attributes and the single-channel controls that were detected for them. The indices of layers and channels start from 0, while ranks start from 1.

13. Attribute Dependency

To compare the disentanglement of different image manipulation methods, we propose a general disentanglement metric for real images, which we refer to as *Attribute Dependency* (AD). Attribute Dependency measures the degree to which manipulation along a certain direction induces changes in other attributes, as measured by classifiers for those attributes. Below we share our insights regarding AD, and its implementation details. Next, we use AD to show our image manipulation method is more disentangled that two other methods (GANSpace [3], InterfaceGan [8]) in Figure 15 and Figure 16. Additionally, we further show in Figure 21 that our method changes face identity less than GANSpace and InterFaceGAN.

For a given target attribute t, we measure AD as follows. First, we sample a set of images without the target attribute t (e.g., without gray hair), and manipulate them towards the target attribute, by a certain amount measured by the change in the logit outcome Δl_t of a classifier pretrained to detect attribute t. Next, we measure the change of logit Δl_i between the original images and the manipulated ones for other attributes $\forall i \in \mathcal{A} \backslash t$, where $\mathcal{A}$ is the set of all attributes. Each change is normalized by $\sigma(l_i)$, the standard deviation of the logit value for attribute i over a large set of generated images. We measure mean-AD, defined as $E(\frac{1}{k}\sum_{i \in \mathcal{A}\backslash t}(\frac{\Delta l_i}{\sigma(l_i)}))$, where $k = |\mathcal{A}| - 1$. Similarly, we measure max-AD, defined as $E(\max_{i \in \mathcal{A}\backslash t}(\frac{\Delta l_i}{\sigma(l_i)}))$.

13.1. Insights

To measure how much a specific attribute $i \in \mathcal{A} \backslash t$ has changed, we use a pretrained classifier for that attribute. Under normal operating mode, a binary classifier outputs a logit $l_i \in [-\infty, +\infty]$, which is then converted to a probability value in $[0, 1]$, with positive logit values yielding probabilities higher than 0.5, and negative logit values yielding probabilities lower than 0.5. However, classifiers trained on real data may be affected by entanglement present in the training data, and they are often unable to detect the presence or absence of an attribute in a disentangled manner. For example, a female face with lipstick will typically cause the classifier to output a negative logit value (indicating the presence of a lipstick), but the classifier might output a positive logit value given a face of a male with lipstick. Similarly, a gray hair classifier will output a negative logit value for a male with gray hair, but might output a positive logit value for a female with gray hair. This is demonstrated in Figures 13 and 14.

Thus, when attempting to measure the magnitude of change of an attribute, we choose not to consider the classifier's logit sign or value; rather, we find that the change in the logit value, Δl, appears to be better correlated with an image space change in the attribute. We use the change in the logit, rather than the change in the probability because of the saturating effect of the sigmoid non-linearity that is used to convert logits to probabilities. For example, the probability produced by a lipstick classifier for a female wearing a lighter lipstick and a stronger lipstick is going to be nearly the same, while this is not the case for the logit values (see the last row of Figure 13).

Another insight is that if the manipulation strength is too high, the generated images will be unrealistic, and classifiers will give unexpected predictions. It is crucial not to use too high manipulation strengths to make sure the logits are meaningful. If the generated images are realistic, the logit is nearly a monotonic function of the strength of target attribute. And we consider the manipulation strength is a monotonic function of strength of target attribute. Therefore, we can control the amount of manipulation (measured by Δl) by searching for the corresponding manipulation strength through bisection method.

Our final insight is that the classifiers are not immune to noise. When provided with the same images with only slight texture changes in hair and background, the classifiers are supposed to output the same logits. In practice, however, the logits are slightly different. Thus, when comparing the effect of different manipulation methods on various attributes, it is necessary to ensure that the differences in the measurements are caused by the inherent differences between the methods, rather than by noise in the classifier outputs.

13.2. Implementation

We randomly generate 500K images, as our image bank, and annotate each image with 31 active attributes, same as was done in Section 3, where a negative logit corresponds to presence of the target attribute in an image. Let $\sigma(l)$ denote the standard deviation of logits over the entire image bank. For each target attribute (for example, gray hair), we rank its logit from negative to positive, and take images with 50-75% quantile as manipulation candidates, since they exhibit little to mild presence of the target attribute (without much gray hair). We don't take images with the most positive logit (75-100% quantile) since they are less likely to result in a realistic manipulation. Candidates are manipulated toward strong attribute presence (adding gray hair, more negative logit). We set the $\Delta l_t = r\sigma(l_t)$, where $r \in \{0.25, 0.5, 0.75, 1\}$. r should not be too large to make sure that most of the manipulated images are still realistic. Then we use the bisection method to find the manipulation strength m that can generate an image with final logit $|(l_t^{final} - (l_t^{initial} - \Delta l_t))| < r_{tolerance}\sigma(l_t)$ with $r_{tolerance} = 5\%$, and ignore images that don't converge after 20 iterations. We manually set the maximum manipulation

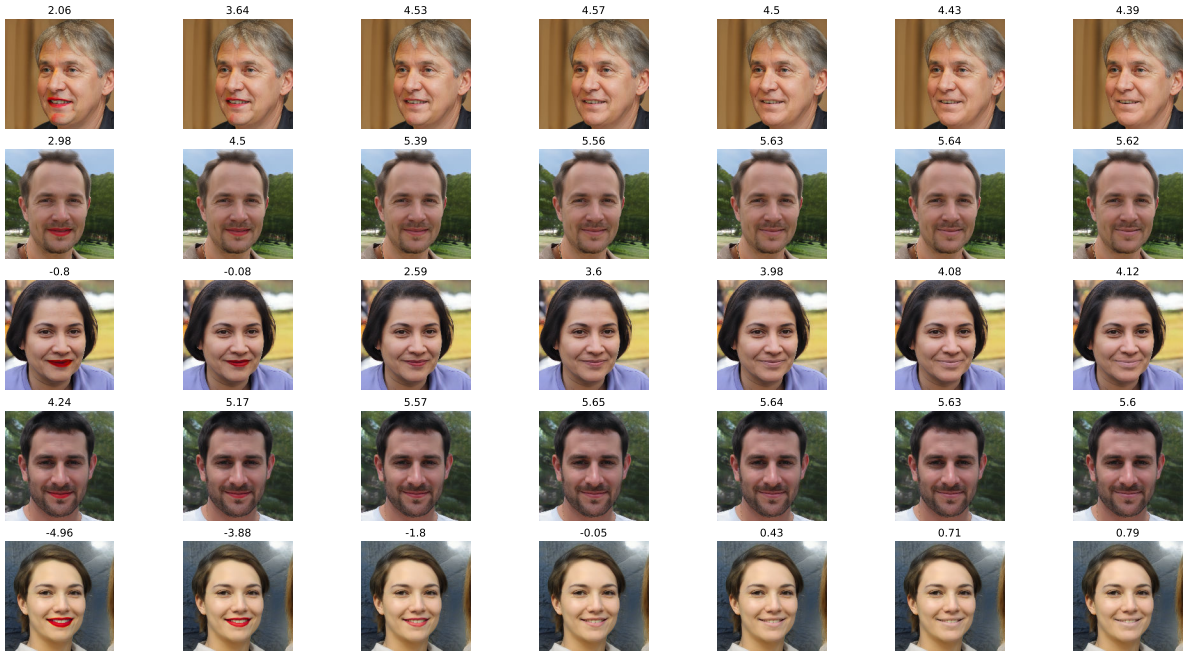


Figure 13. Logits of lipstick classifier. The strength of attribute is reduced from left to right. The classifier logit is on top of each image, increasing from left to right. Note that the logit sign is not aligned with the presence of the attribute: there are images with strong lipstick, but a positive logit.

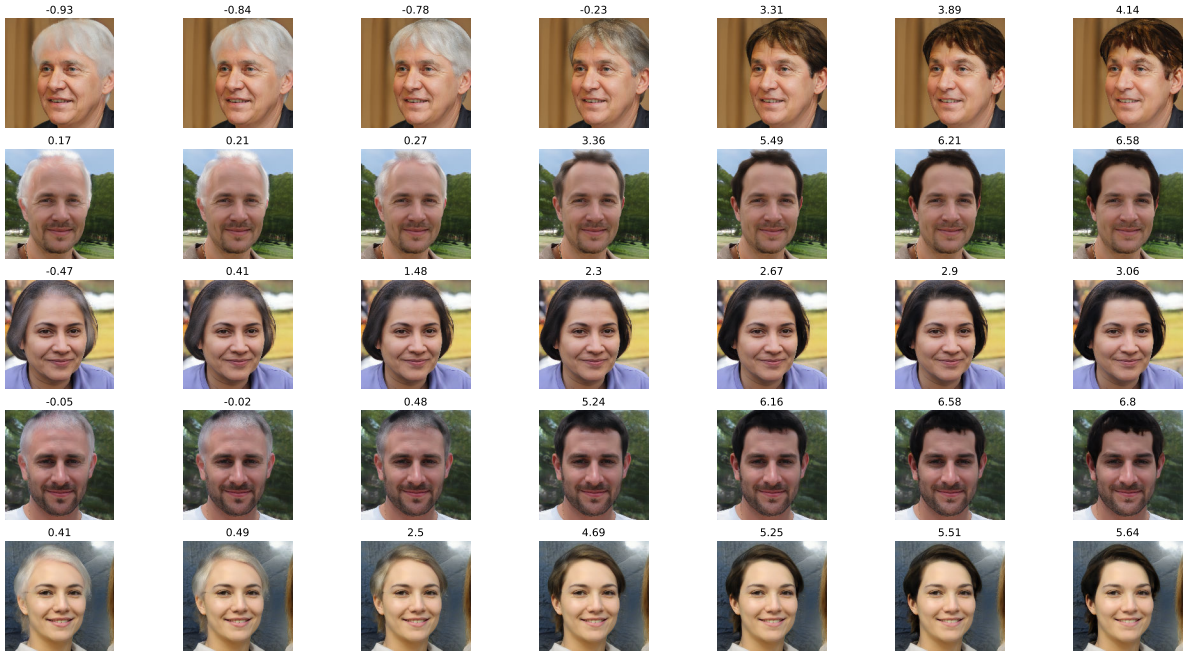


Figure 14. Logits of hair greyness classifier. The strength of attribute is reduced from left to right. The classifier logit is on top of each image, increasing from left to right. Note that the logit sign is not aligned with the presence of the attribute: there are images with strong hair greyness, but a positive logit.

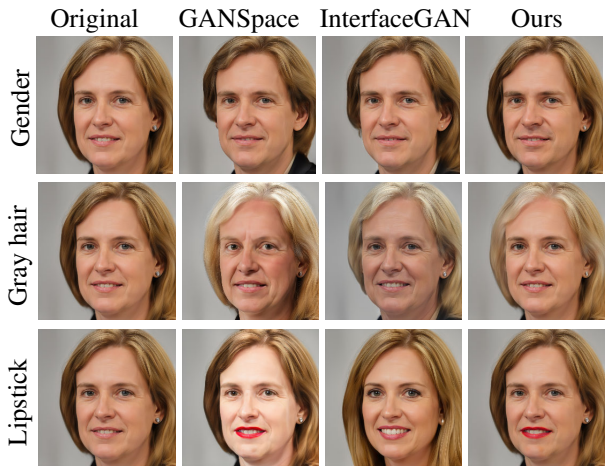


Figure 15. Comparison with state-of-the-art methods with amount of manipulation $\Delta l_t = 1.5\sigma(l_t)$. We deliberately choose a strong manipulation (1.5 instead of 1) to emphasize the differences.

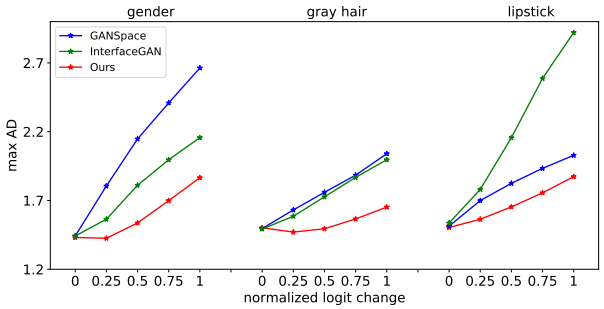


Figure 16. Max-AD vs. the degree of target attribute manipulation ($\Delta l_t/\sigma(l_t)$). Lower max-AD indicates better disentanglement.

strength m_{max} such that almost all manipulated images with manipulation strength m_{max} strongly exhibit the target attribute, but still look realistic. m_{max} is used to initiate the bisection method.

Next, we add a control group with $r = 0$ to the experiment, such that $\Delta l_t = 0\sigma(l_t)$. This group is used to represent the inherent noise of classifiers. The input images are copies of the original ones, obtained by keeping the latent code s unchanged, but changing the noise inputs at different layers. They are essentially identical to the original images, with subtle differences in hair, skin, and background.

Finally, we use the same 3K images with identifiable m for all candidate manipulation methods to calculate mean-AD and max-AD. The mean-AD for the three methods (GANSpace, InterFaceGAN, and ours) for three attributes (gender, gray hair, and lipstick) are plotted in Figure 8, and the max-AD in Figure 16. Figures 6 and 15 show a qualitative comparison between the manipulations produced by the three methods. Note that, like in Figure 6, the *Lipstick* manipulation by InterFaceGAN significantly changes the identity of the person, and the *Gray hair* manipulation adds wrinkles. GANSpace manipulations also exhibit some entanglement (*Lipstick* affects face lightness, *Gray hair* ages the rest of the face). In contrast, our approach appears to affect only the target attribute. Our *Gender* manipulation, for example, does not affect the hair style, and minimally changes the face, yet the gender unmistakably changes.

13.3. Identity change

In addition to AD, we use another metric (identity change) to compare our method against GANSpace and InterFaceGAN. Specifically, we use FaceNet [7], which is a standard network for measuring identity change. We use the official implementation, which first detects faces, then crops faces out, obtains an embedding from the last layer, and calculates the Euclidean norm between the embedding of the original images and that of the manipulated ones. As shown in Figure 21, manipulations done with our method change the identity less than manipulations of GANSpace or InterFaceGAN.

14. Manipulation of real images

To manipulate real images, it is necessary to first invert them into latent codes. Through latent optimization [5], we observe that the reconstruction quality is the highest when optimizing in $\mathcal{S}$, followed by $\mathcal{W}+$, and is the lowest for $\mathcal{W}$, as demonstrated in Figure 17. However, the naturalness of subsequent manipulation is the best when the latent optimization is done in $\mathcal{W}$, followed by $\mathcal{W}+$, and the worst for $\mathcal{S}$, as shown in Figure 18. Through training a latent embedding encoder and using the embedding produced by the encoder as the initial point for a few iterations of latent optimization, we obtain both good reconstruction and natural manipulation. We demonstrate manipulation of real images in Figure 19 (for real images from the FFHQ dataset) and in Figure 20 for images from the CelebA-HQ [6] dataset.

15. Additional Comparisons

We compare our method to two other image manipulation methods, Image2stylegan++ [1] and StyleFlow [2].

Figure 22 shows a comparison with Image2stylegan++ [1], where users can edit an image by drawing strokes over it, and the annotated image is then inverted into the StyleGAN latent space, which is supposed to result in a realistic rendering. For example, in order to change the hair color, the user must place strokes of the desired color in the hair region. As demonstrated in Figure 22, the method isn't necessarily successful in producing a realistic edit in this manner, while our generates more realistic results. It should also be pointed out that more complex edits, such as changing gender or adding wrinkles is difficult, if not impossible, to convey by such user-drawn strokes.

Figure 23 shows a comparison with StyleFlow [2], where it may be seen that our method and StyleFlow are bot able to generate convincing results of comparable realism. However, StyleFlow simultaneously uses several attribute classifiers and regressors (from the Microsoft face API), and is thus able to manipulate a limited set of attributes. In contrast, our method only requires one target attribute.

References

[1] Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2stylegan++: How to edit the embedded images? In *Proc. CVPR*, pages 8296–8305, 2020.

[2] Rameen Abdal, Peihao Zhu, Niloy Mitra, and Peter Wonka. Styleflow: Attribute-conditioned exploration of stylegan-generated images using conditional continuous normalizing flows. *arXiv e-prints*, pages arXiv–2008, 2020.

[3] Erik Härkönen, Aaron Hertzmann, Jaakko Lehtinen, and Sylvain Paris. GANSpace: discovering interpretable GAN controls. *arXiv preprint arXiv:2004.02546*, 2020.

[4] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proc. CVPR*, pages 4401–4410, 2019.

[5] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of StyleGAN. In *Proc. CVPR*, pages 8110–8119, 2020.

[6] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pages 3730–3738, 2015.

[7] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[8] Yujun Shen, Ceyuan Yang, Xiaoou Tang, and Bolei Zhou. InterFaceGAN: interpreting the disentangled face representation learned by GANs. *arXiv:2005.09635*, 2020.

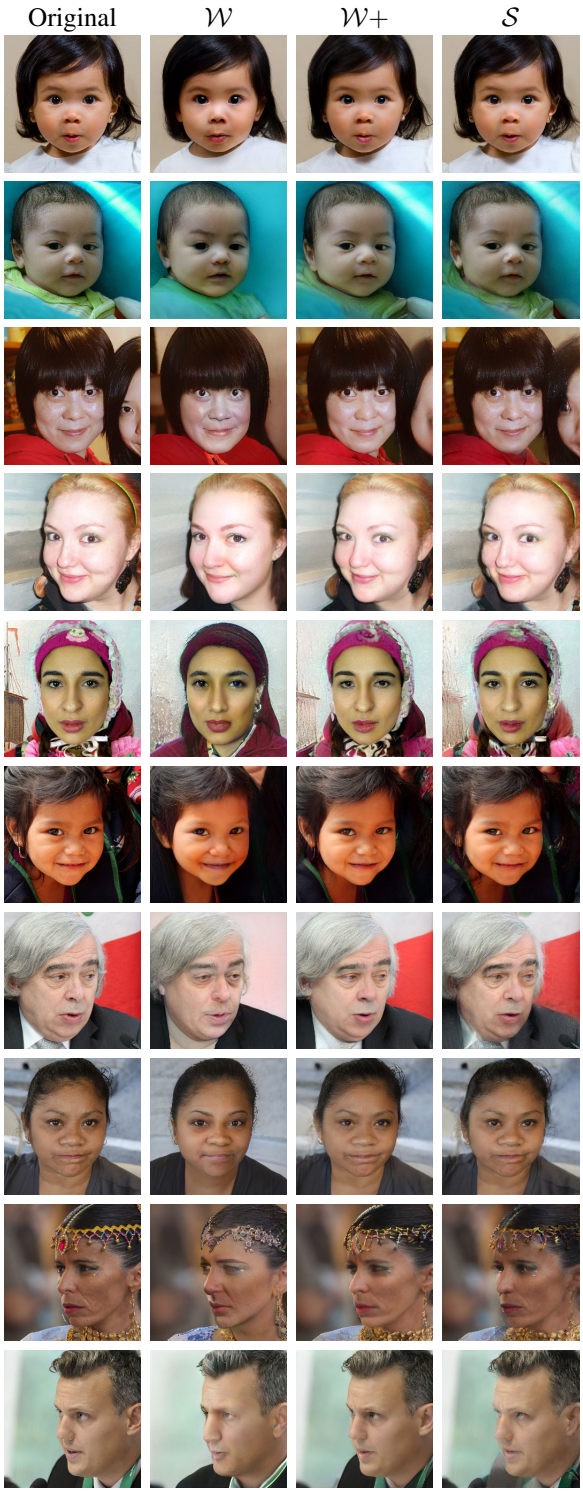


Figure 17. Inversion via latent optimization in $\mathcal{W}$, $\mathcal{W}+$, $\mathcal{S}$. It may be easily seen that the reconstruction is least accurate for $\mathcal{W}$, more accurate for $\mathcal{W}+$, and is best for $\mathcal{S}$

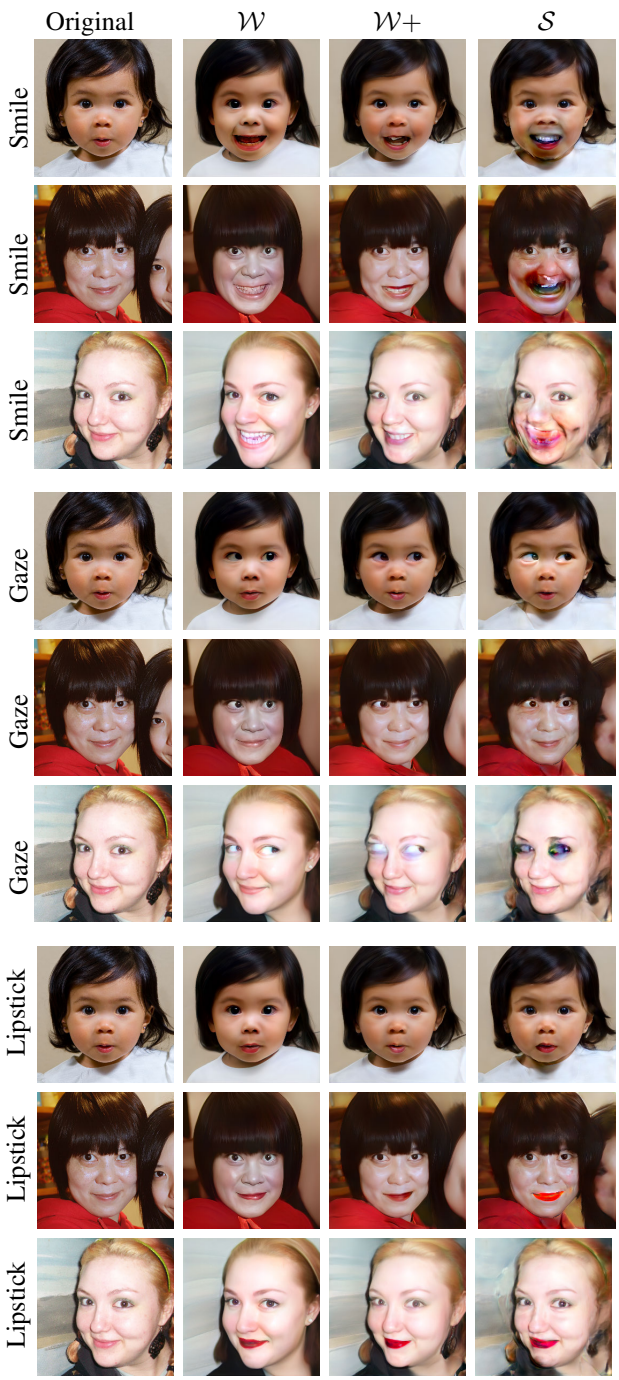


Figure 18. Manipulation of style codes obtained by latent optimization in $\mathcal{W}$, $\mathcal{W}+$, and $\mathcal{S}$ spaces. The exact same manipulation is applied in each row. It may be seen that manipulation of codes optimized in $\mathcal{S}$ produces significant artifacts, while manipulation on codes optimized in $\mathcal{W}$ produce more realistic results.

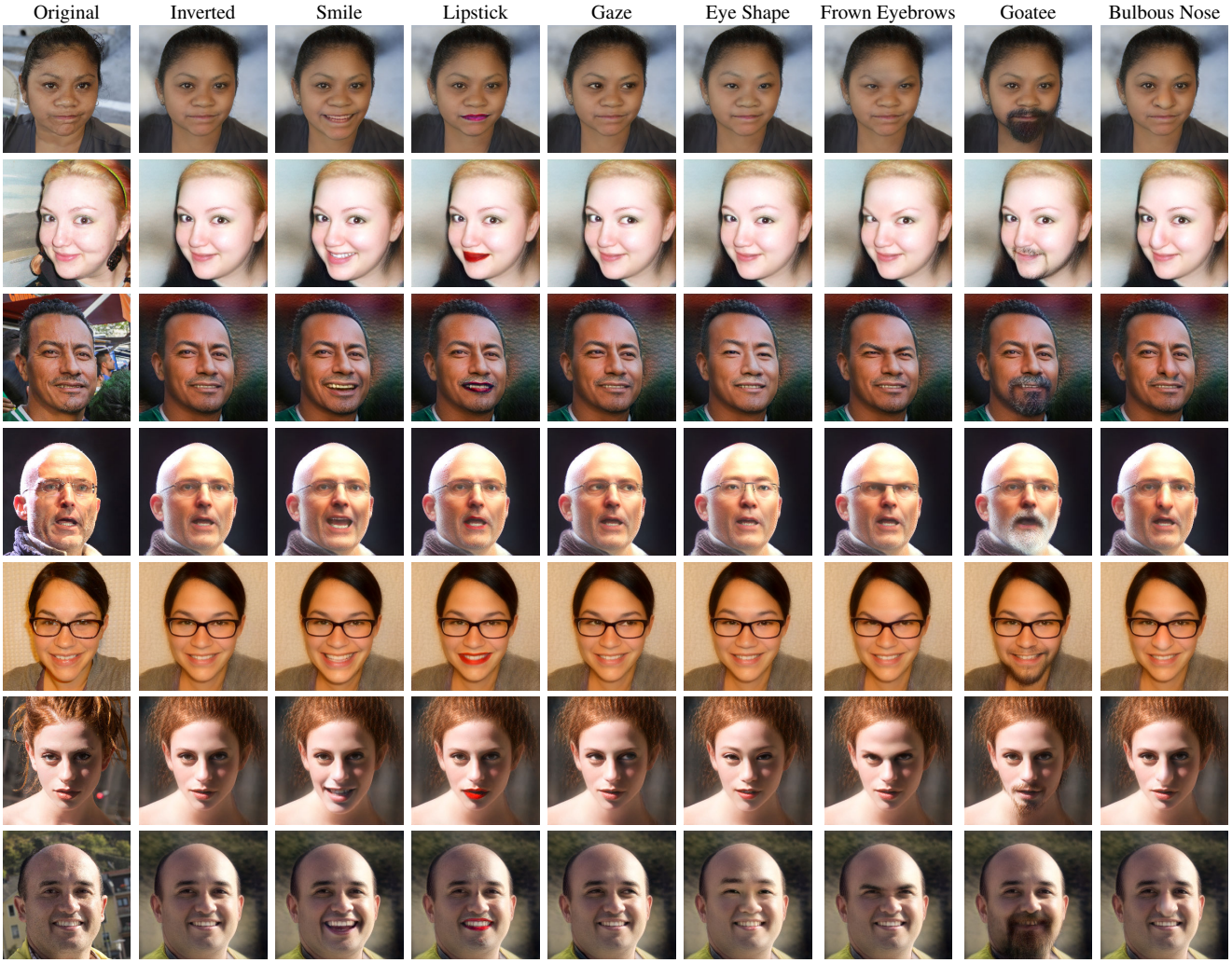


Figure 19. Manipulation of real images using encoder-based inversion. Original images are from FFHQ, and were not part of the encoder's training set.

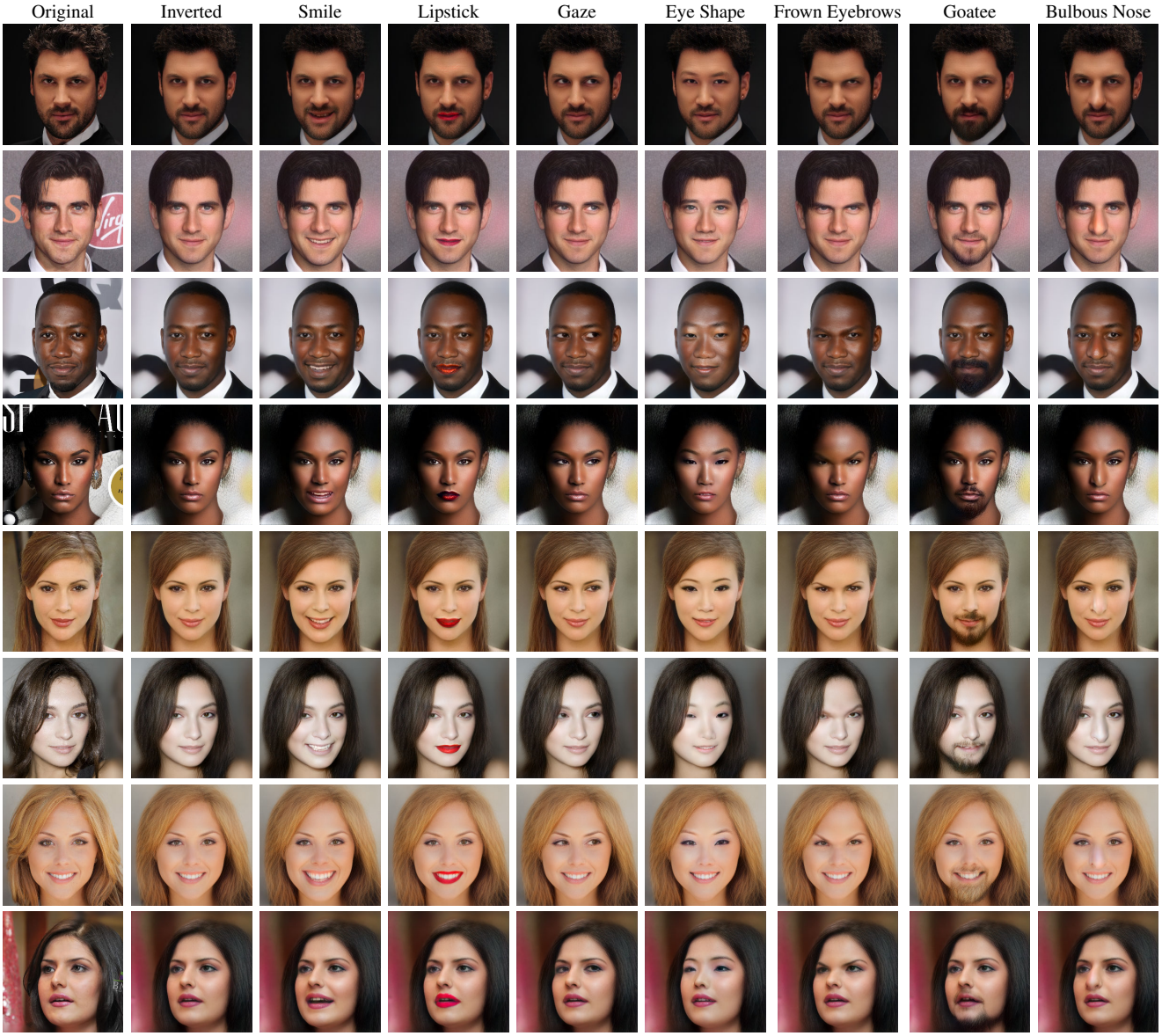


Figure 20. Manipulation of real images using encoder-based inversion. Original images are from CelebA-HQ, which were not part of the encoder's training set, and not part of the GAN training set (the StyleGAN2 model was trained on the FFHQ dataset).

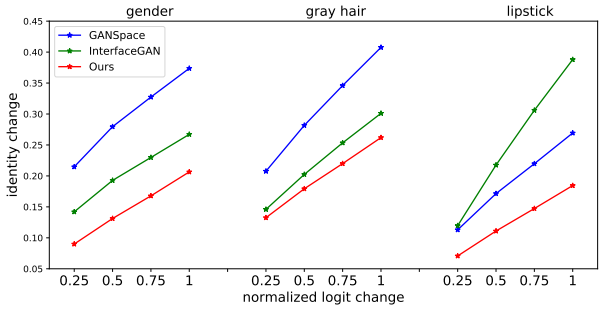


Figure 21. Identity change vs. the degree of target attribute manipulation ($\Delta l_t / \sigma(l_t)$). Lower identity changes indicates that the manipulation is better disentanglement from the identity.

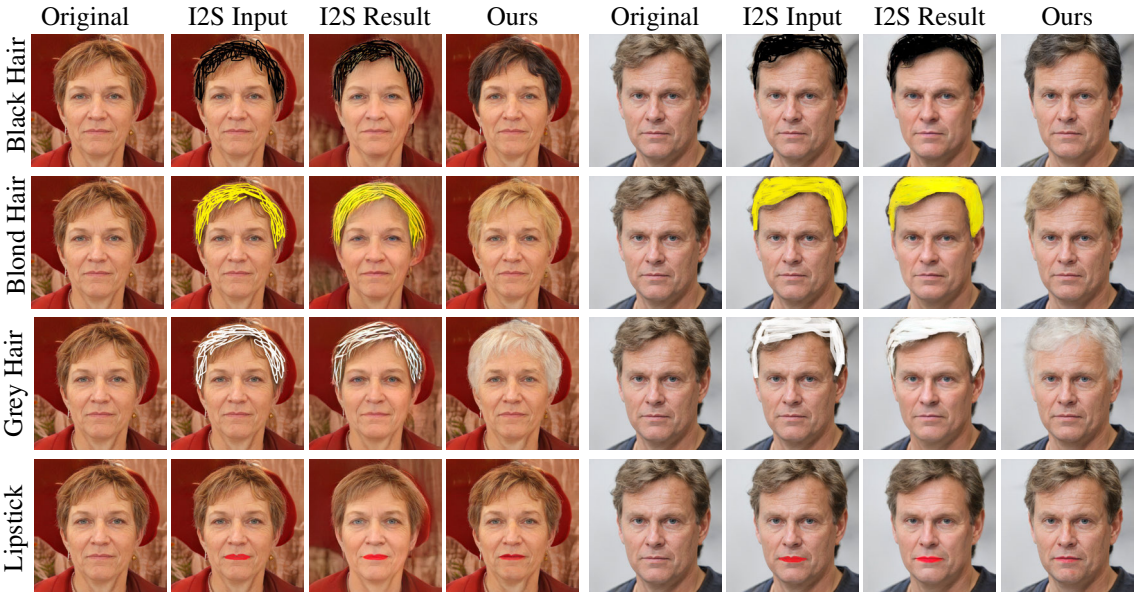


Figure 22. Comparison between Image2stylegan++ [1] and our method. Our method typically succeeds in achieving more natural and realistic results.

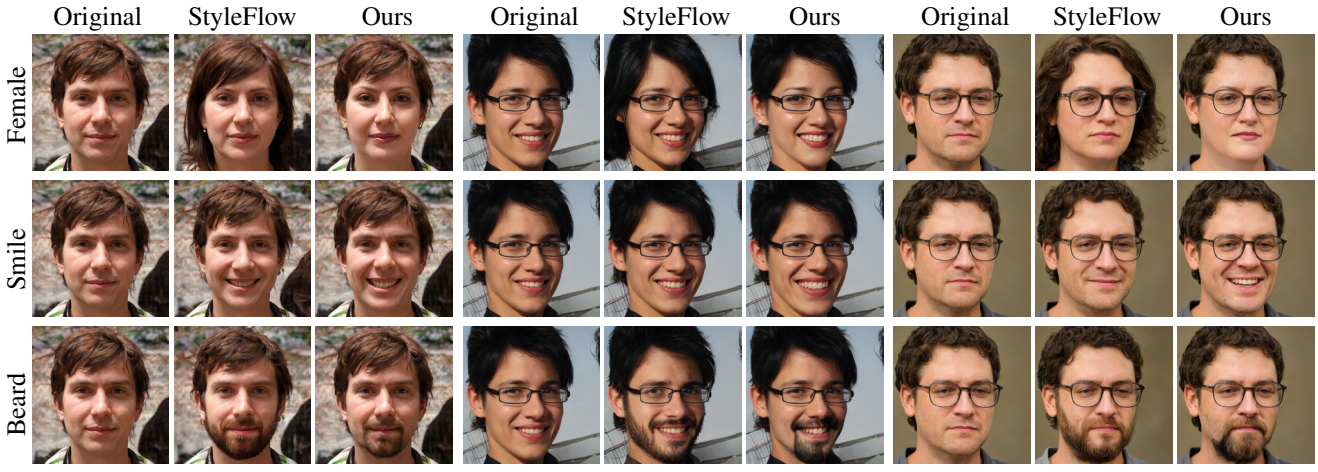


Figure 23. Comparison between StyleFlow [2] and our method. Our method is typically more fine-grained, for example, it can change gender without changing hair style, while StyleFlow usually changes the length of hair. In some edits, such as adding a smile, our method introduces a more pronounced change, compared to StyleFlow. In other edits, such as adding a beard, both methods achieve similar results.