Supplementary Material: Out-of-domain GAN inversion via Invertibility Decomposition for Photo-Realistic Human Face Manipulation

1. Summary

In this supplementary document, we will give more details about our model architecture, loss function, and training strategy in Sec. 2.1, 2.2, and 2.3, respectively. Also, we conduct more experiments and display more visual results.

2. Implementation Details

2.1. Model Architecture

In our main paper, we introduce our face inversion framework with invertibility decomposition. Our model consists of three parts: the encoder, the generator, and the spatial alignment and masking module (SAMM).

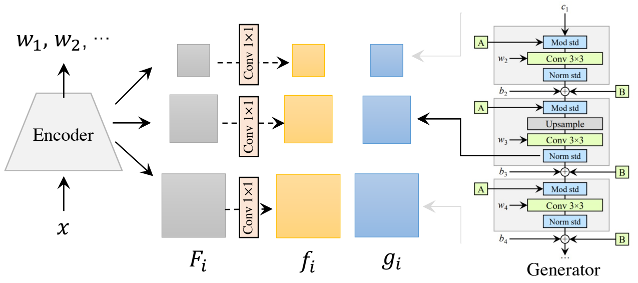


Figure 1: The encoder [15] and the generator [8] architecture, where the "Conv 1×1" denotes the vanilla 2D convolution module and the kernel size is 1.

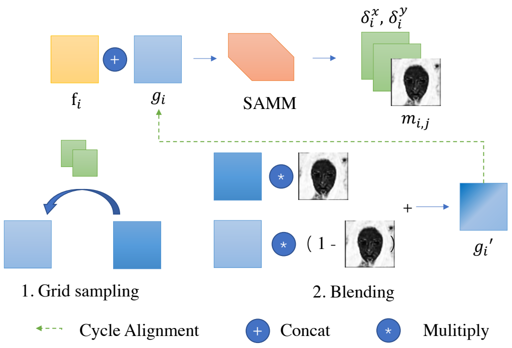


Figure 2: The spatial alignment and masking module.

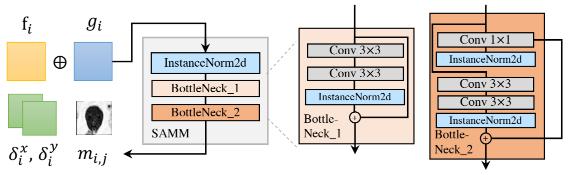


Figure 3: The network architecture of SAMM.

Encoder and generator. We adopt the official PyTorch implementation of the e4e [15] and ReStyle [1] encoder with their official pre-trained checkpoint. The encoders are trained with a StyleGAN2 [8] generator pre-trained on the FFHQ [7] dataset. We especially choose the ReStyle encoder which shares the same architecture as the e4e encoder for convenience. We fix all the parameters of the pre-trained encoder and generator during our experiment. In Fig. 1, we visualize the encoder and generator architecture. Because the intermediate feature maps extracted by the encoder do not match the shape of generated features g_i, we also train 1×1 convolution layers at each resolution to produce f_i from F_i. Especially, we obtain g_i after the style modulated convolution [8] but before the noise injection from each upsampling convolution block in the StyleGAN2 generator. In this paper, we select the feature maps at multiple resolutions from the encoder and generator for optical flow computation and invertibility mask prediction. For the ReStyle encoder which iteratively encodes the images to latent codes, we only extract the feature maps in the last iteration. The shapes of the feature maps are shown in Tab. 1.

Layer i	F_i	f_i, g_i
1	$32 \times 32 \times 256$	$32 \times 32 \times 512$
2	$64 \times 64 \times 128$	$64 \times 64 \times 512$
3	$128 \times 128 \times 64$	$128 \times 128 \times 256$
4	$256 \times 256 \times 64$	$256 \times 256 \times 128$

Table 1: The shapes of feature maps in our experiments.

Spatial Alignment and Masking Module (SAMM). In our main paper, we introduce the spatial alignment and masking module, which estimates the optical flow and mask from $f_i \oplus g_i$, where $\oplus$ is the concatenation operator. In Fig. 3, we illustrate the network architecture of our SAMM module. Especially, for each layer in Tab. 1, we train a correspond SAMM. The SAMM consists of the InstanceNorm (IN) module [16] and the BottleNeck modules [13]. For convenience, we define the following terms:

- Conv(c_{in}, c_{out}, k) denotes the vanilla 2D convolution layer with stride 1, where k is the kernel size.
- BottleNeck_1(c_{in}) denotes the layer with a residual connection between the input and Conv$(c_{in}, c_{in}, 3)$-PReLU-Conv$(c_{in}, c_{in}, 3)$-IN layer, as visualized in Fig. 3.
- BottleNeck_2(c_{in}, c_{out}) denotes the layer with a residual connection between the Conv$(c_{in}, c_{out}, 1)$-IN layer and Conv$(c_{in}, c_{out}, 3)$-PReLU-Conv$(c_{out}, c_{out}, 3)$-IN layer, as visualized in Fig. 3.

We summerize the architecture of SAMM for each layer i in Tab. 2.

Layer i	SAMM architecture
1	IN-BottleNeck_1(1024)-BottleNeck_2(1024, 3)
2	IN-BottleNeck_1(1024)-BottleNeck_2(1024, 3)
3	IN-BottleNeck_1(512)-BottleNeck_2(512, 3)
4	IN-BottleNeck_1(256)-BottleNeck_2(256, 3)

Table 2: The architecture of SAMM in our experiments.

2.2. Loss Function

In our main paper, we introduce our loss functions, including the reconstruction loss L_{rec}, the adversarial loss L_{adv}, and the mask regularization loss L_{mask}. Here, we will introduce more details about the perceptual loss L_{per} in L_{rec} and the L_{adv}.

Perceptual loss. Our perceptual loss L_{per} contains the feature loss L_{feat} and the style loss L_{style}. With a VGG19 [14] feature extractor $\Phi(x)$, the feature loss and style loss are calculated by:

$$L_{feat}(x, \hat{x}) = \sum_i \lambda_i \mathbb{E}(||\Phi_i(x) - \Phi_i(\hat{x})||), \qquad (1)$$

$$L_{style}(x, \hat{x}) = \sum_i \lambda_i \mathbb{E}(||\mathrm{Gram}(\Phi_i(x)) - \mathrm{Gram}(\Phi_i(\hat{x}))||), \quad (2)$$

where $\mathrm{Gram}(\cdot)$ is the Gram matrix [6] of the input tensor, $\Phi_i(\cdot)$ means extracting the i-th layer's feature from VGG, and $\mathbb{E}$ is the mean operation. Following [18], we use the {"conv_1", ..., "conv_5"} features before the activation for loss calculation and λ_i is the weight of the loss on different layers. We have:

$$L_{per}(x, \hat{x}) = L_{feat}(x, \hat{x}) + \lambda_{style} L_{style}(x, \hat{x}), \qquad (3)$$

and λ_{style} is the style loss weight which we set to 50 in most experiments.

Adversarial loss. In our training stage, we also involve the adversarial loss [8] with a discriminator D:

$$L_{adv}(\hat{x}) = -\mathbb{E}[\mathrm{softplus}(D(\hat{x}))]. \qquad (4)$$

And the discriminator D is trained to minimize the discriminator loss L_D:

$$L_D(x, \hat{x}) = \mathbb{E}[log(D(\hat{x}))] + \mathbb{E}[log(1 - D(x))]. \qquad (5)$$

2.3. Training Details

In our experiments, our main model is trained on the FFHQ [7] dataset with a single Nvidia RTX 3090 GPU. We adopt the Adam [9] optimizer to optimize the 1×1 convolution layers in the encoder, the SAMM, and the discriminator. We set the learning rate to 2×1.0^{-5} for all modules, the batch size to 2, and train our model for 40,000 iterations.

Progressive training. As we train multiple SAMM at different resolutions and the generator generates features progressively. We adopt a progressive training strategy for SAMM, i.e., we enable the SAMM of {2, 3, 4} layer (Tab. 2) at {2,000, 6,000, 10,000} iteration, respectively.

3. Experiments

3.1. Mask Regularization Loss

To examine the effectiveness of our mask regularization loss L_{mask}, we train models with different ϕ_{area}. In Fig. 5, we visualize the inversion results and their corresponding masks. The experiment shows that with decreasing ϕ_{area}, the prediction of the Out-Of-Domain (OOD) area becomes more conservative. On the other hand, when ϕ_{area} goes too large, the eyes and mouth area on the face are also detected as the OOD area, which is not ideal for downstream tasks such as attribute manipulation. In this paper, we set $\phi_{area} = 0.3$ for $32^2, 64^2$ masks and $\phi_{area} = 0.25$ for $128^2, 256^2$ masks to balance the editability and the fidelity of our inversion framework.

3.2. Ghosting Artifact

In our main paper, we demonstrate that our spatial alignment on the generated features mitigates the geometrical misalignment between the input image and the generated result at the blending stage. Consequently, our model trained with spatial alignment predicts more accurate invertibility masks and produces artifact-free blending results. In this section, we further investigate the influence of spatial alignment at different resolutions. As is shown in Fig. 4, when we progressively enable the spatial alignment in more layers in our model, the ghosting artifacts are gradually removed from the blending results. That spatial alignment is important in our framework for high-fidelity GAN inversion for face images.

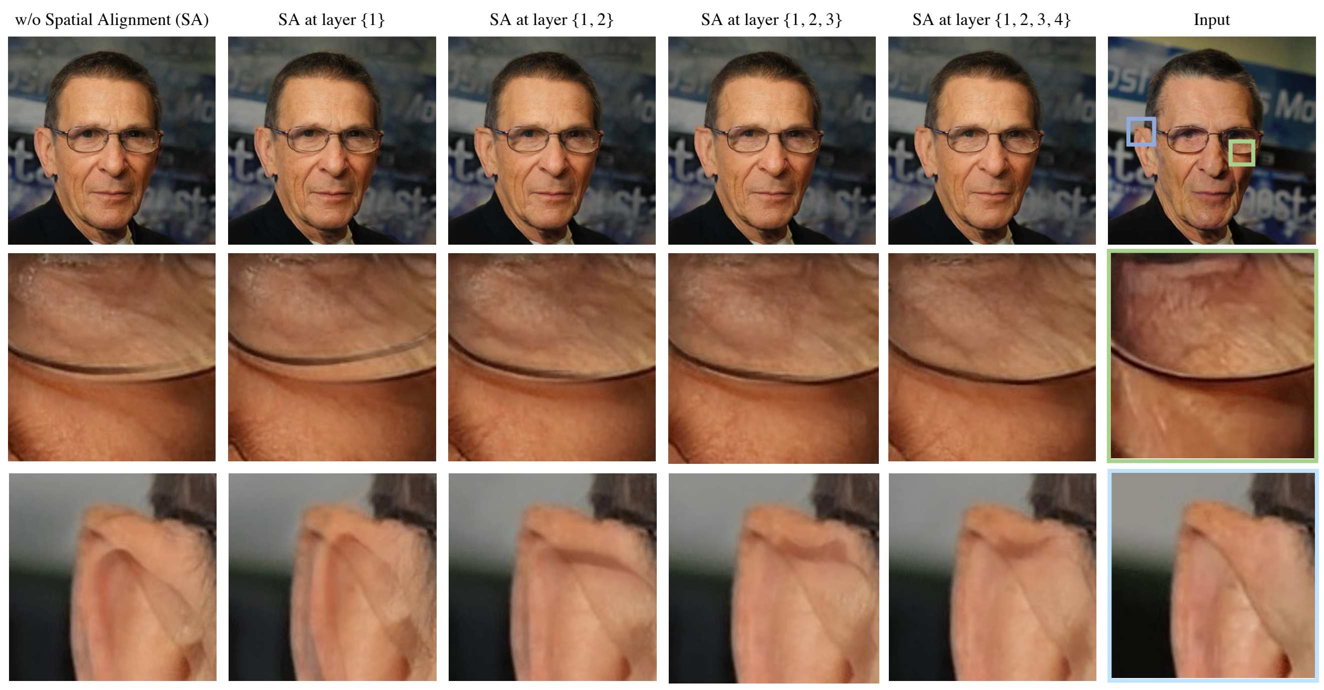


Figure 4: **Ablation study on the Spatial Alignment (SA) layers.** We progressively enable the spatial alignment from the low-resolution layer to the high-resolution layer in our model and blend the generated image with the input image with the same invertibility masks. As shown in the results, as we enable more SA layers, the ghosting artifacts in the inversion result decrease significantly.

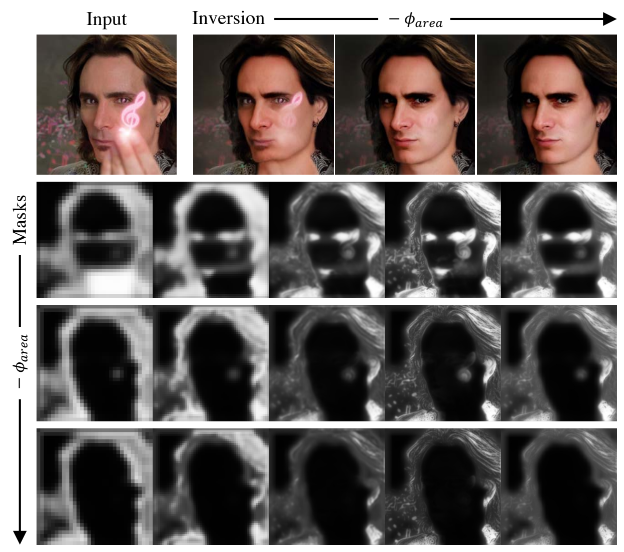
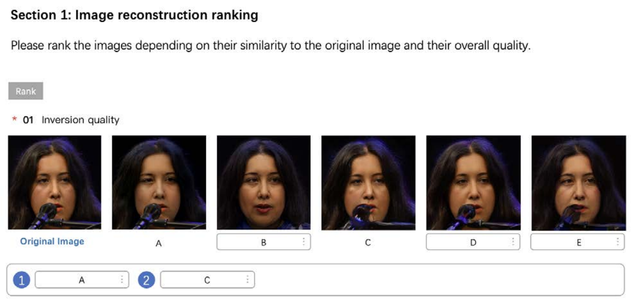


Figure 5: **Ablation of Mask Regularization.** With increasing strength of the mask regularization (denoted as $-\phi_{area}$), the area of predicted out-of-domain regions (i.e., white areas in the masks) goes smaller.


Section 1: Image reconstruction ranking

Please rank the images depending on their similarity to the original image and their overall quality.

Rank

* 01 Inversion quality

Figure 6: A screenshot of our user study questionnaire for inversion quality ranking. The questionnaire of attribute editing ranking follows the same structure.

4. User Study

In this paper, we conduct a user study to research the user's preferences on the GAN inversion and attribute manipulated results of our model and the baselines. In the user study, we ask the participants to compare the input image with the results of different methods for specific tasks such as inversion, age editing, etc. We provide 5 sets of images for inversion quality ranking and 6 sets of images for editing quality ranking. We collect 50 questionnaires (Fig. 6) from the internet and summarize the user's preferences as shown in Tab. 2 in our main paper.

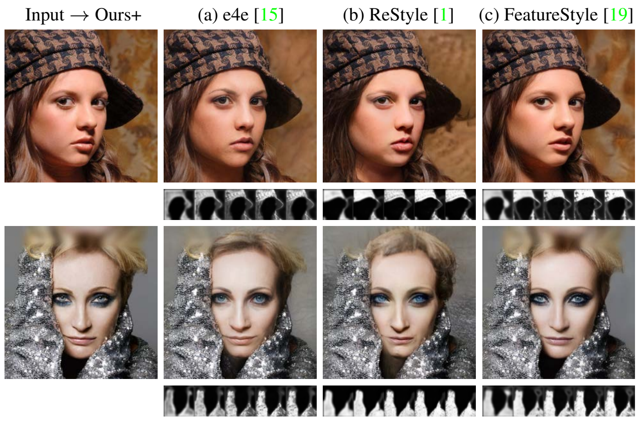


Figure 7: **Our GAN inversion with different encoders.** We implement our framework with different pre-trained encoders for StyleGAN2 [8] generator. We show the predicted out-of-domain area under the GAN inversion results.

Method	PSNR↑	SSIM↑	LPIPS↓	FID↓
FeatureStyle [19]	25.24	0.717	0.188	16.86
Ours$_{FeatureStyle}$, N=2	**28.46**	**0.837**	**0.152**	**13.39**

Table 3: Quantitative evaluation of GAN inversion quality on the first 1,000 images in the CelebAHQ-Mask [11] testing dataset.

5. GAN Invertibility

In our main paper, we solve the GAN invertibility from the perspective of spatial decomposition (the decomposition of In-Domain (ID) and OOD areas). In contrast, previous works mainly focus on translating the images to the GAN latent manifold to build the ID regions. We add more discussions for these works that are not mentioned in the main paper. Zhu et al. [20] proposed the encoder for ID inversion, where they trained a domain-guided encoder with reconstruction loss and adversarial loss. Some following works [13, 15] improved the ID inversion by training the encoder progressively and applying adversarial supervision on the predicted latent vectors. Our work is built on the existing image-to-latent encoders, and we expect the encoder encodes the images into ID latent vectors as accurately as possible. With the latent predicted, our framework can detect the OOD areas in the input images, then skip the inversion in such areas.

Our SAMM as a general plugin. In our main paper, we introduce our SAMM module trained with the pre-trained e4e [15], Restyle [1] encoders and the StyleGAN2 [8] generator. Conceptually, our framework could be used as a general plug-in for image-to-latent encoders which share a similar GAN inversion architecture.

For example, our framework could also be extended with the pre-trained FeatureStyle [19] encoder. We demonstrate the inversion results and predicted OOD masks of our framework using different encoders in Fig. 7. Our inversion results consistently maintain high fidelity with different encoder architectures. We also evaluate the inversion quality of our framework with the pre-trained FeatureStyle encoder in Tab. 3. Our framework improves the inversion quality of all pre-trained encoders significantly by decomposing the OOD area and aligning the ID area to the ground truth. The "Ours$_{FeatureStyle}$" model performs best among all the other settings in the inversion quality, however, the FeatureStyle [19] encoder works on an extended latent space, which damages its editability. Thus, we choose "Ours$_{e4e}$" as our default model for most experiments. We will also explore the possibility of adapting our framework to more encoders in future work.

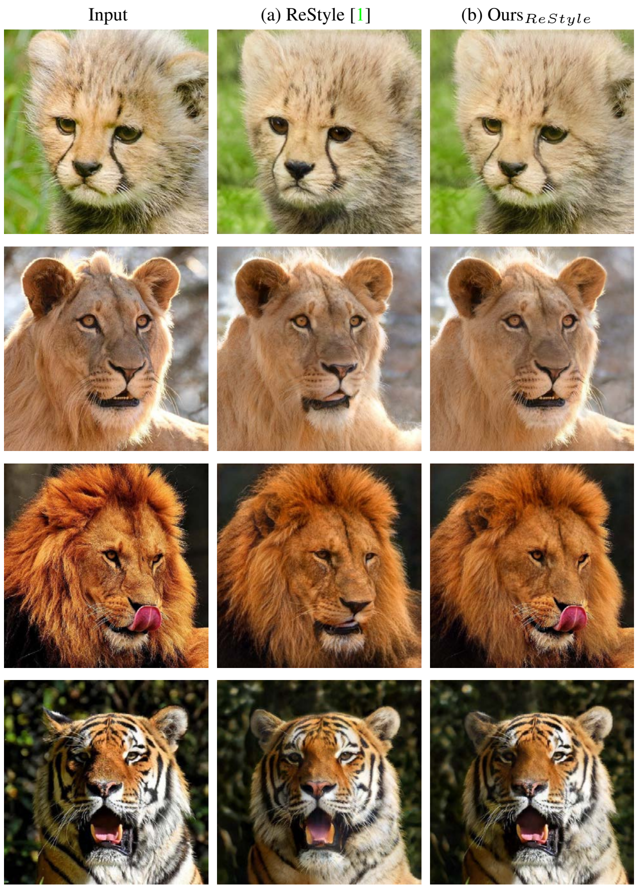


Figure 8: **GAN inversion on AFHQ [3] wild animal dataset.** (a) Results with pre-trained ReStyle [1] encoder. (b) Results of our framework. Overall, our framework helps better reconstruct the mouths and eyes of the input images.

Domain generalization. While our primary focus is on out-of-distribution (OOD) GAN inversion for human faces, we also investigate the domain generalization capability of our proposed framework on the Stanford Cars [10] dataset in Sec. 4.2 of our main paper. Additionally, we extend our work to the AFHQ [3] dataset for wild animal GAN inversion, as a complementary evaluation. The results shown in Fig. 8 demonstrate that our framework is also effective in improving the inversion quality of the pre-trained encoder (ReStyle [1] in this specific case) on wild animal images. Nevertheless, our current framework doesn't always work well on tasks that introduce large-scale geometry changes in the output (e.g., pose or shape editing). We aim to extend our SAMM with pose awareness to support such editing in our future study.

6. More Visual Results

In this section, we display more visual results of our framework in face image inversion and attribute manipulation results on both testing datasets and real-world images.

Face inversion. We visualize our face inversion results against baseline methods in Fig. 9.

Face editing. We visualize more face editing results with our method in Fig. 10. In Fig. 11, we apply our approach on style transfer [5] and attribute manipulation on real-world images.

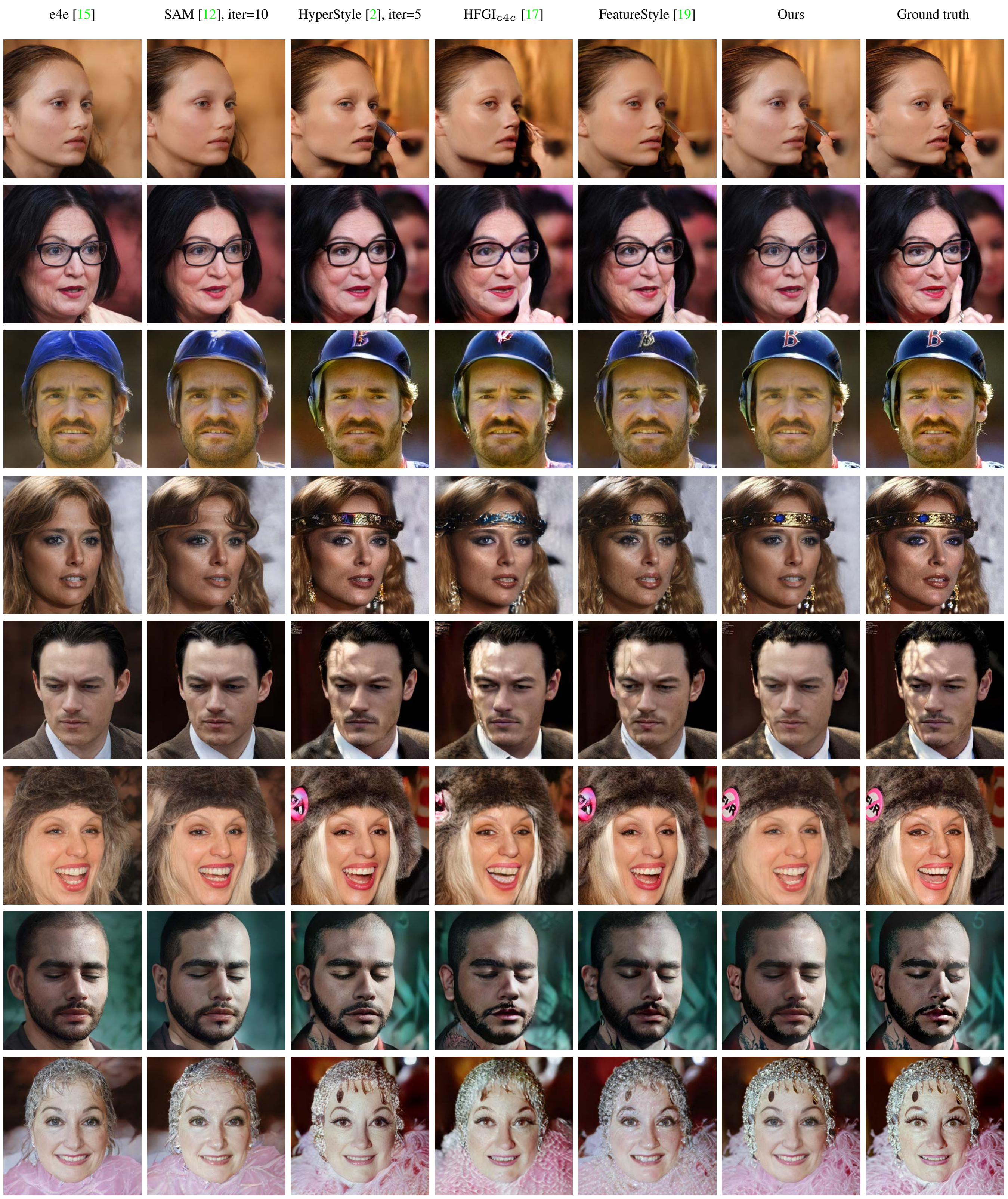


Figure 9: Comparison of GAN inversion quality on CelebAMask-HQ [11] testing dataset. Please zoom in for details.

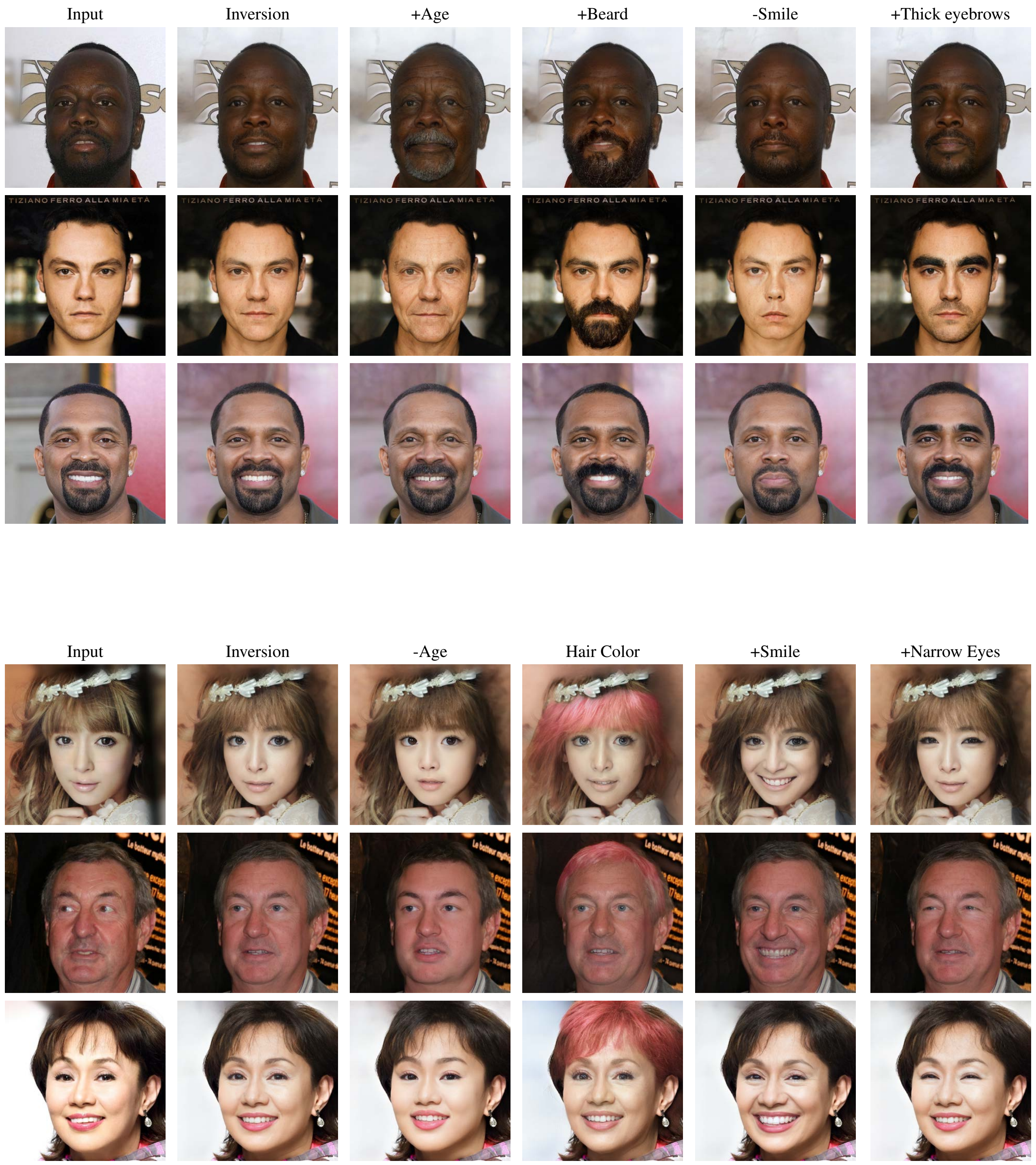


Figure 10: Face manipulation results of our framework on CelebAMask-HQ [11] dataset, please zoom in for detail.

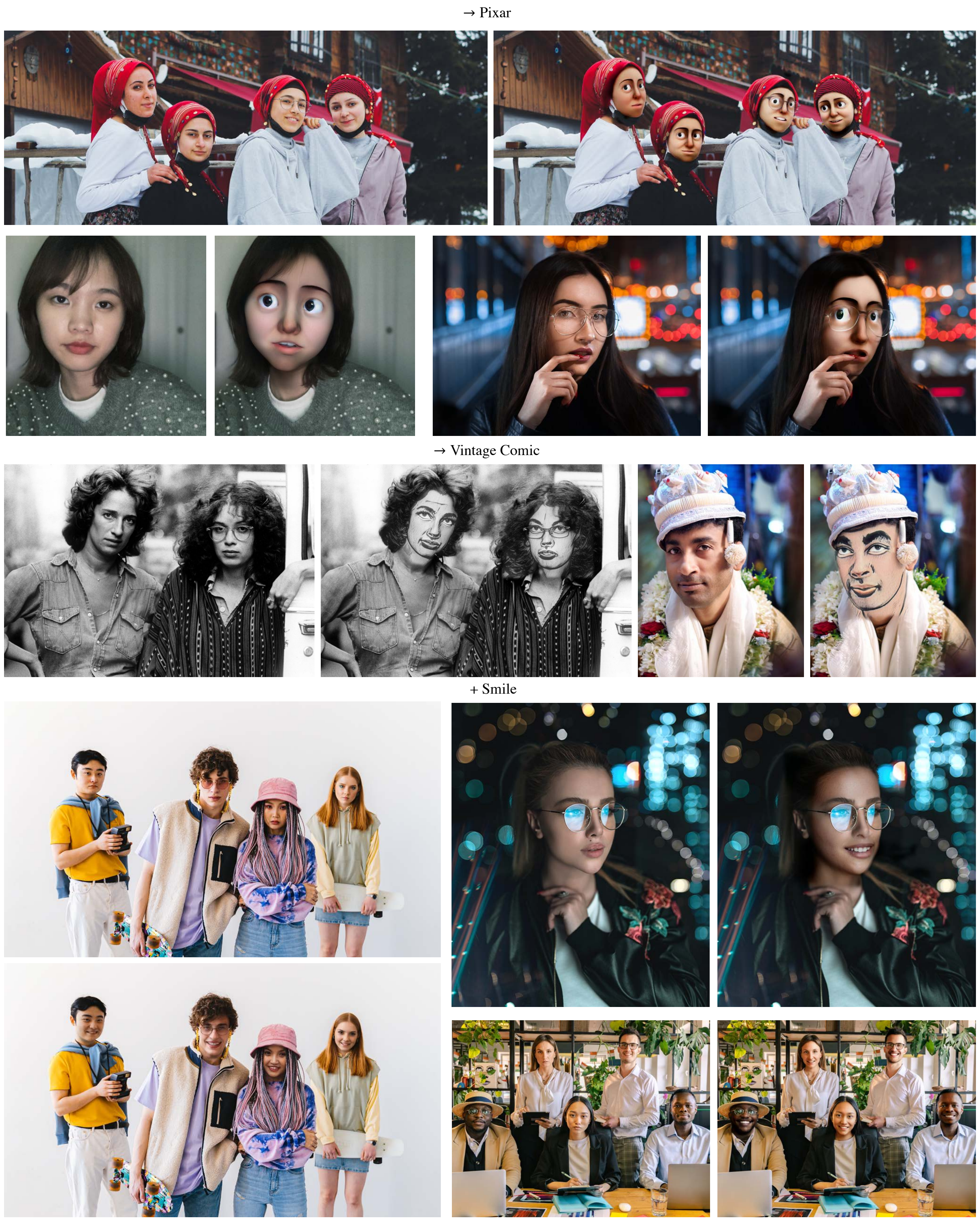


Figure 11: Face manipulation results on real-world images. We detect faces in the image with RetinaFace [4] model. For face style transfer ("Pixar" and "Vintage Comic"), we adopt the pre-trained generators from StyleGAN-NADA [5] without retraining our SAMM layers. Please zoom in for details.

References

[1] Yuval Alaluf, Or Patashnik, and Daniel Cohen-Or. Restyle: A residual-based stylegan encoder via iterative refinement. In *ICCV*, 2021. 1, 4, 5

[2] Yuval Alaluf, Omer Tov, Ron Mokady, Rinon Gal, and Amit H. Bermano. Hyperstyle: Stylegan inversion with hypernetworks for real image editing. In *CVPR*, 2021. 6

[3] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. In *CVPR*, 2020. 4, 5

[4] Jiankang Deng, Jia Guo, Evangelos Ververas, Irene Kotsia, and Stefanos Zafeiriou. Retinaface: Single-shot multi-level face localisation in the wild. In *CVPR*, 2020. 8

[5] Rinon Gal, Or Patashnik, Haggai Maron, Gal Chechik, and Daniel Cohen-Or. Stylegan-nada: Clip-guided domain adaptation of image generators, 2021. 5, 8

[6] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *CVPR*, 2016. 2

[7] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019. 1, 2

[8] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *CVPR*, 2020. 1, 2, 4

[9] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 2

[10] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *3dRR-13*, 2013. 4

[11] Cheng-Han Lee, Ziwei Liu, Lingyun Wu, and Ping Luo. Maskgan: Towards diverse and interactive facial image manipulation. In *CVPR*, 2020. 4, 6, 7

[12] Gaurav Parmar, Yijun Li, Jingwan Lu, Richard Zhang, Jun-Yan Zhu, and Krishna Kumar Singh. Spatially-adaptive multilayer selection for gan inversion and editing. In *CVPR*, 2022. 6

[13] Elad Richardson, Yuval Alaluf, Or Patashnik, Yotam Nitzan, Yaniv Azar, Stav Shapiro, and Daniel Cohen-Or. Encoding in style: a stylegan encoder for image-to-image translation. In *CVPR*, 2021. 2, 4

[14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv*, 2014. 2

[15] Omer Tov, Yuval Alaluf, Yotam Nitzan, Or Patashnik, and Daniel Cohen-Or. Designing an encoder for stylegan image manipulation. *ACM TOG*, 2021. 1, 4, 6

[16] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv*, 2016. 2

[17] Tengfei Wang, Yong Zhang, Yanbo Fan, Jue Wang, and Qifeng Chen. High-fidelity gan inversion for image attribute editing. In *CVPR*, 2021. 6

[18] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy. Esrgan: Enhanced super-resolution generative adversarial networks. In *ECCVW*, 2018. 2

[19] Xu Yao, Alasdair Newson, Yann Gousseau, and Pierre Hellier. A style-based gan encoder for high fidelity reconstruction of images and videos. In *ECCV*, 2022. 4, 6

[20] Jiapeng Zhu, Yujun Shen, Deli Zhao, and Bolei Zhou. In-domain gan inversion for real image editing. In *ECCV*, pages 592–608. Springer, 2020. 4