Supplementary for What Decreases Editing Capability? Domain-Specific Hybrid Refinement for Improved GAN Inversion

A. Implementation Details

Architecture. In the Domain-Specific Segmentation module, we use FaRL [1] and SLIC [2] as the parsing model and superpixel algorithm. We use 1024×1024 StyleGAN2 as face generator, which is trained on FFHQ dataset.

Hyper-parameters. In Domain-Specific Segmentation module, we set two thresholds $\tau_1 = 0.7$ and $\tau_2 = 0.8$, according to parsing results, respectively. We set a higher threshold for eyes, nose, and mouse areas. For In Hybrid Modulation Refinement, coarse inversion optimizes initial latent codes for 40 steps. We add the additional modulation feature at 11th layer amount 18 layers of 1024×1024 StyleGAN2. Adam optimizers with no weight decay and betas$(0.9, 0.999)$ are used for weight and feature modulation, and learning rates are 0.0015 and 0.09, respectively. The feature is optimized in 100 steps and weight is only optimized in 50 steps for better editing results.

B. Runtime Analysis

One main concern of optimization-based methods is time-consuming. Previous methods (i.e., PTI and SAM) cost more than one minute per image, which is hardly applied. We next demonstrate that DHR dramatically outperforms all existing methods in terms of speed.

Settings. As PTI [3], SAM [4] and DHR are optimization-based refinement inversion methods and gain better inversion results, we compare their runtime and PSNR using 100 randomly selected images from CelebA-HQ test set. All experiments are conducted on a single NVIDIA RTX 3090 GPU and i9-10900X CPU (superpixel algorithm using CPU). For PTI and SAM, we change the optimization steps to get each data point, while the optimization process for pivotal latent codes in PTI is fixed. For DHR, we fix the coarse inversion steps (i.e., 40 steps), and change the feature modulation steps while weight modulation steps are set to half. The result is shown in Figure 1.

Results. Compared to previous methods, DHR accelerates the refinement process significantly and attains higher performance (almost 33dB of PSNR). Although the overly long modulation process may cause editing capability degradation, we experimentally find that 200/100 steps (20.1s with 29.19dB PSNR) for feature/weight modulation gain reasonable results. **Our method achieves previous SOTA using only 7.5s, compared to 62.3s for PTI, and gains extroadinary results in 14s**.

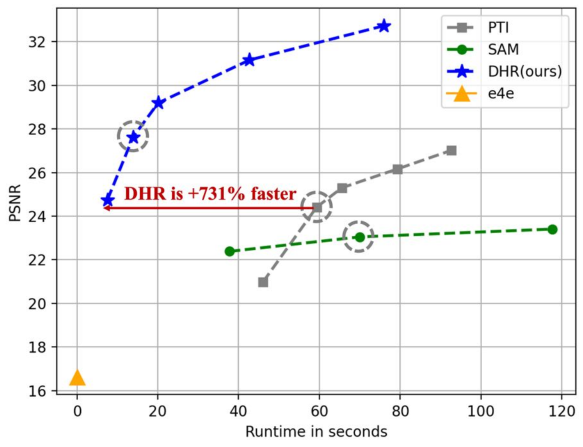

Figure 1. **Runtime Analysis.** We compare three optimization-based refinement methods using 100 CelebA-HQ test images. The default configuration of each method is marked by a dashed box. Our method greatly accelerates the refinement optimization process and gains much higher performance. **DHR only cost 7.5s to achieve the previous SOTA.**

C. Effects of Domain-Specific Segmentation

We illustrate the difference between Domain-Specific Segmentation (DSS) and invertibility segmenter in SAM [4] in Figure 3. As invertibility segmenter consists of invertibility predictor and pretrained segmentation model, it has two primary differences from DSS:

Semantic prior plays a significant role in invertibility prediction. Since there is no semantic information, simple color block areas are easily misclassified, as can be seen in Figure 2. The characters have high invertibility in $\mathcal{W}^+$ space. Our DSS draws on the semantic face prior from pretrained GAN, having better ability to distinguish "easy-to-

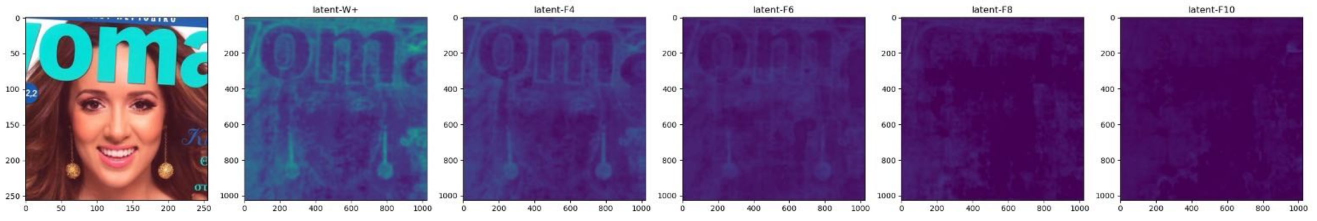

Figure 2. **Invertibility prediction in SAM [4].**

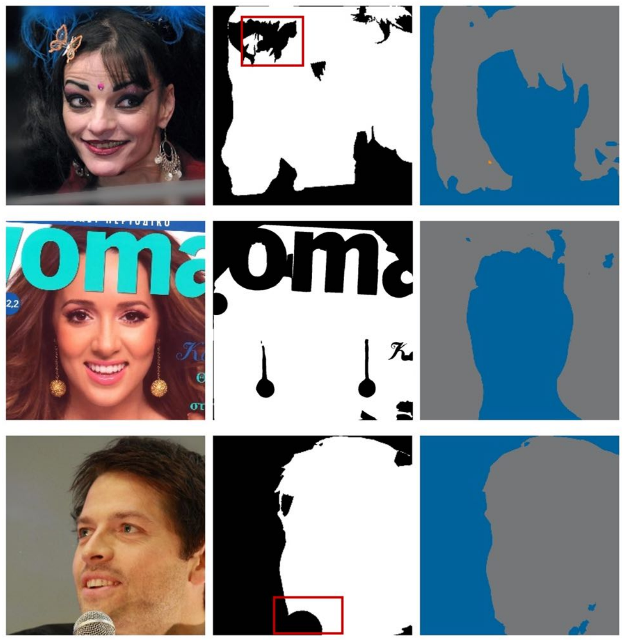

Figure 3. **Comparison between Domain-Specific Segmentation and Invertibility Segmenter in SAM [4].**

invert" (i.e., *in-domain*) and "hard-to-invert" (i.e., *out-of-domain*).

General segmentation model can not segment complex area. Another main difference is that SAM uses general segmentation models to split areas, which can only distinguish pre-defined categories. In real-world images, there are a large number of complex areas, such as hairpins, occlusion, and microphones. We address it by using the superpixel algorithm, which is required to partition image into different areas and is category-agnostic.

This also verifies the idea of "partition and binarize". We use superpixel to split tens of areas and coarsely inverse to binarize each of them. Hence, DDS is robust to real-world images and attains promising segmentation results.

D. Additional Results

We further demonstrate additional inversion and editing results:

- Figure 4: Comparison of "Smile" Editing.
- Figure 5: Comparison of "Young" Editing.
- Figure 6: Comparison of "Lipstick" Editing.
- Figure 7: Results of "Eye Closure" Editing.
- Figure 8: Results of "Wrinkles" Editing.
- Figure 9: Results of "Eyebrow Thickness" Editing.

References

[1] Yinglin Zheng, Hao Yang, Ting Zhang, Jianmin Bao, Dongdong Chen, Yangyu Huang, Lu Yuan, Dong Chen, Ming Zeng, and Fang Wen. General facial representation learning in a visual-linguistic manner. *arXiv preprint arXiv:2112.03109*, 2021. 1

[2] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2274–2282, 2012. 1

[3] Daniel Roich, Ron Mokady, Amit H Bermano, and Daniel Cohen-Or. Pivotal tuning for latent-based editing of real images. *ACM Trans. Graph.*, 2021. 1

[4] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11399–11409, 2022. 1, 2

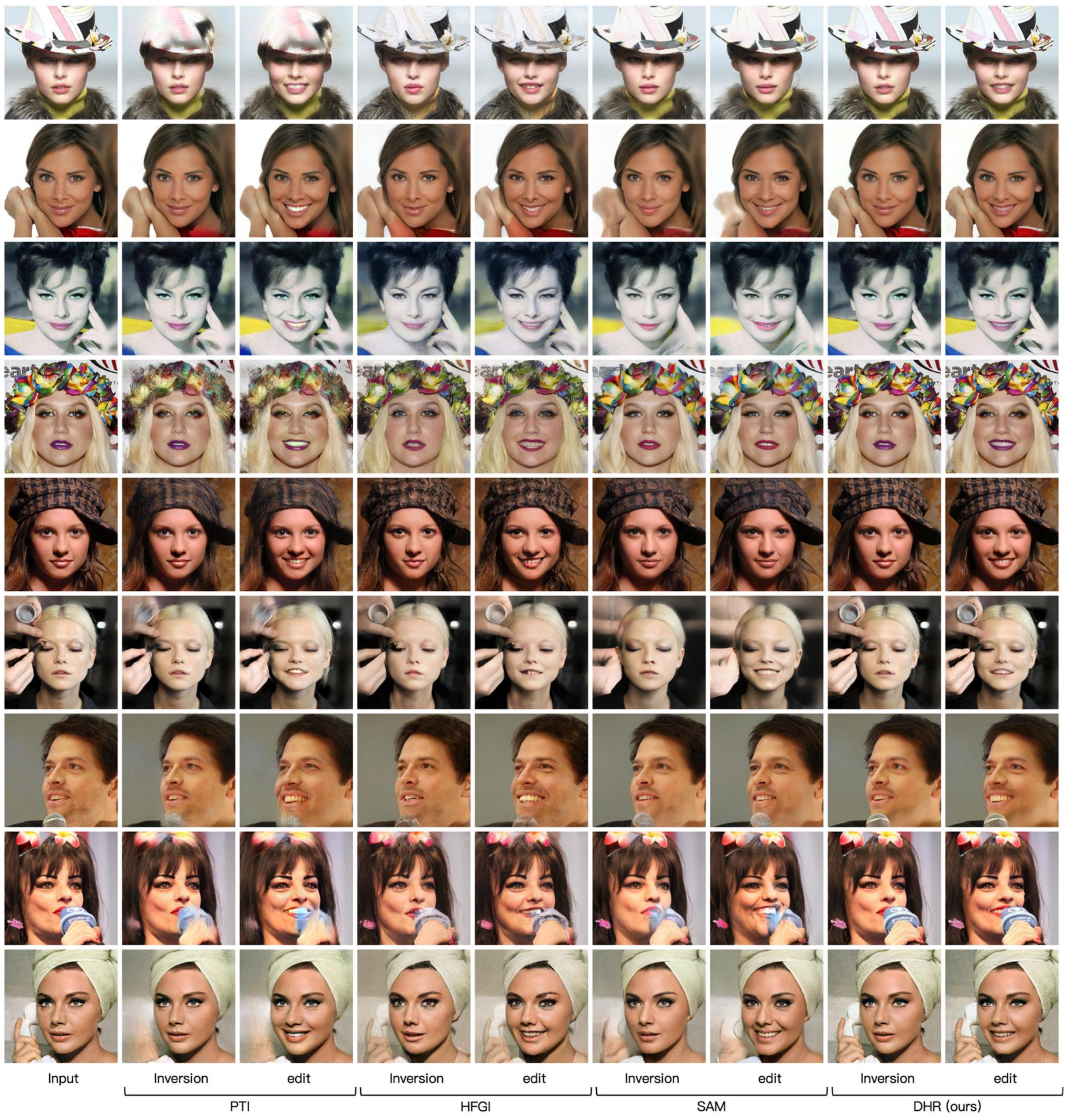

Figure 4. **Comparison of the editing result with the previous methods in "Smile".**

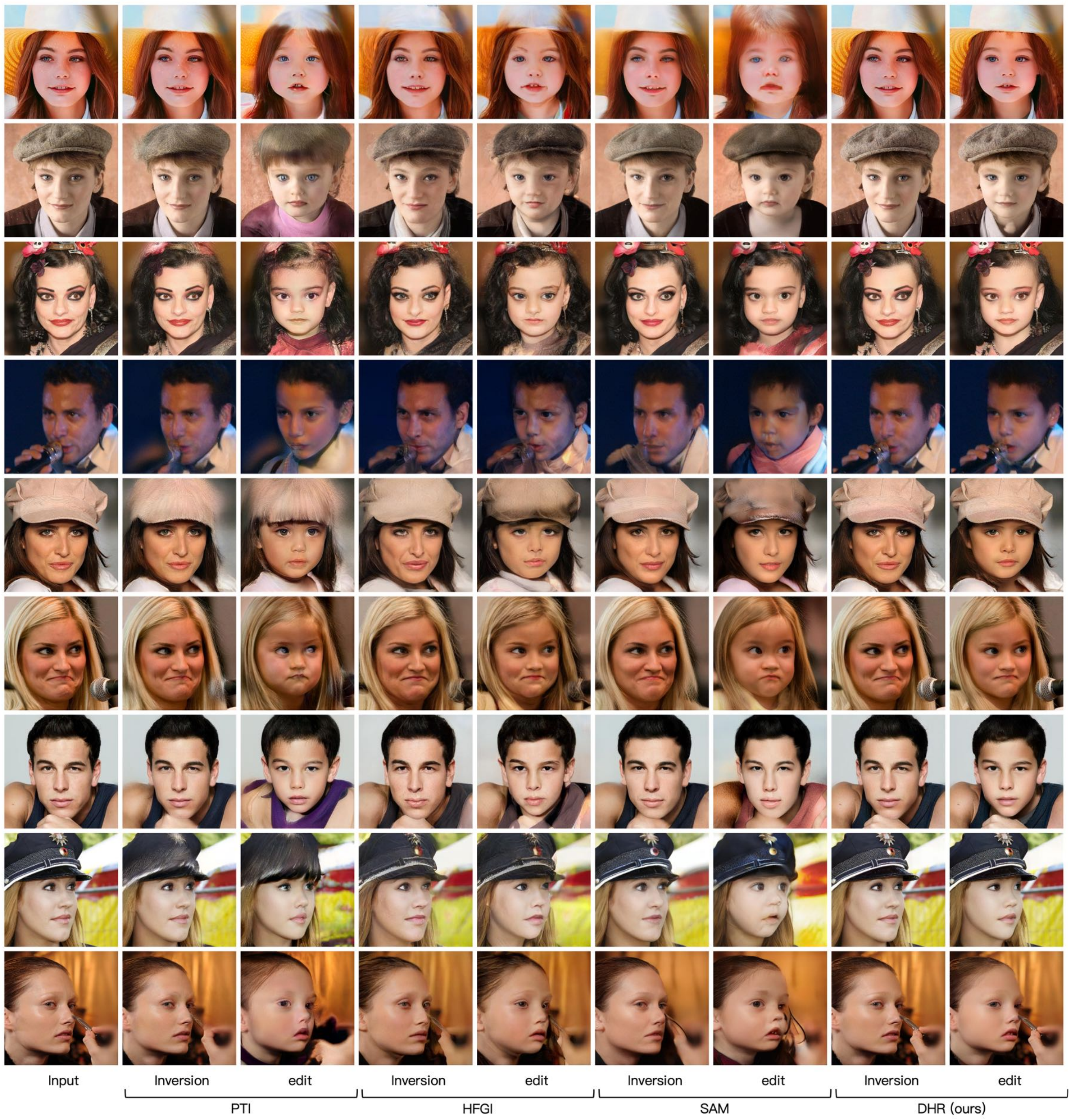

Figure 5. **Comparison of the editing result with the previous methods in "Young".**

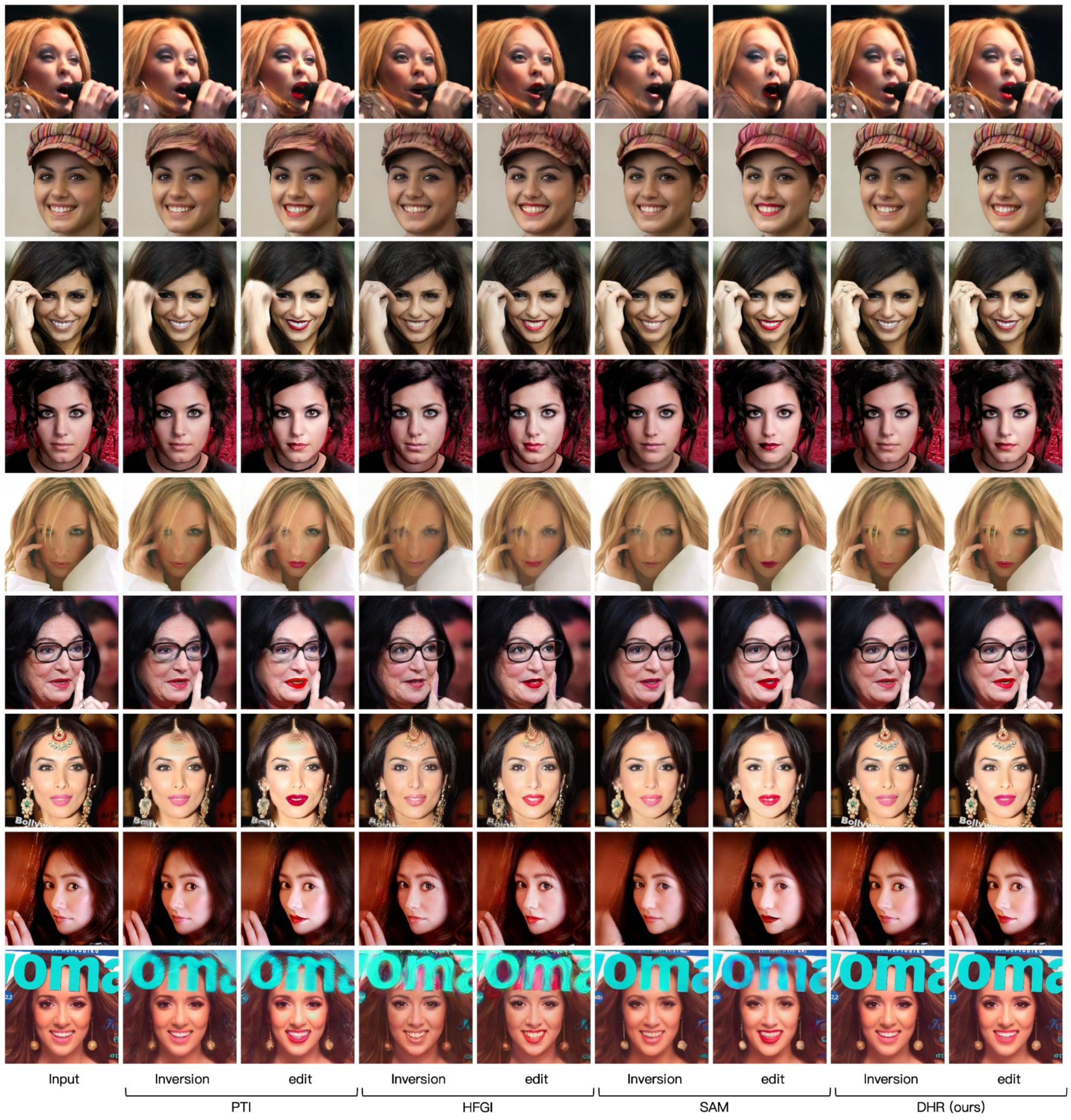

Figure 6. **Comparison of the editing result with the previous methods in “Lipstick”.**

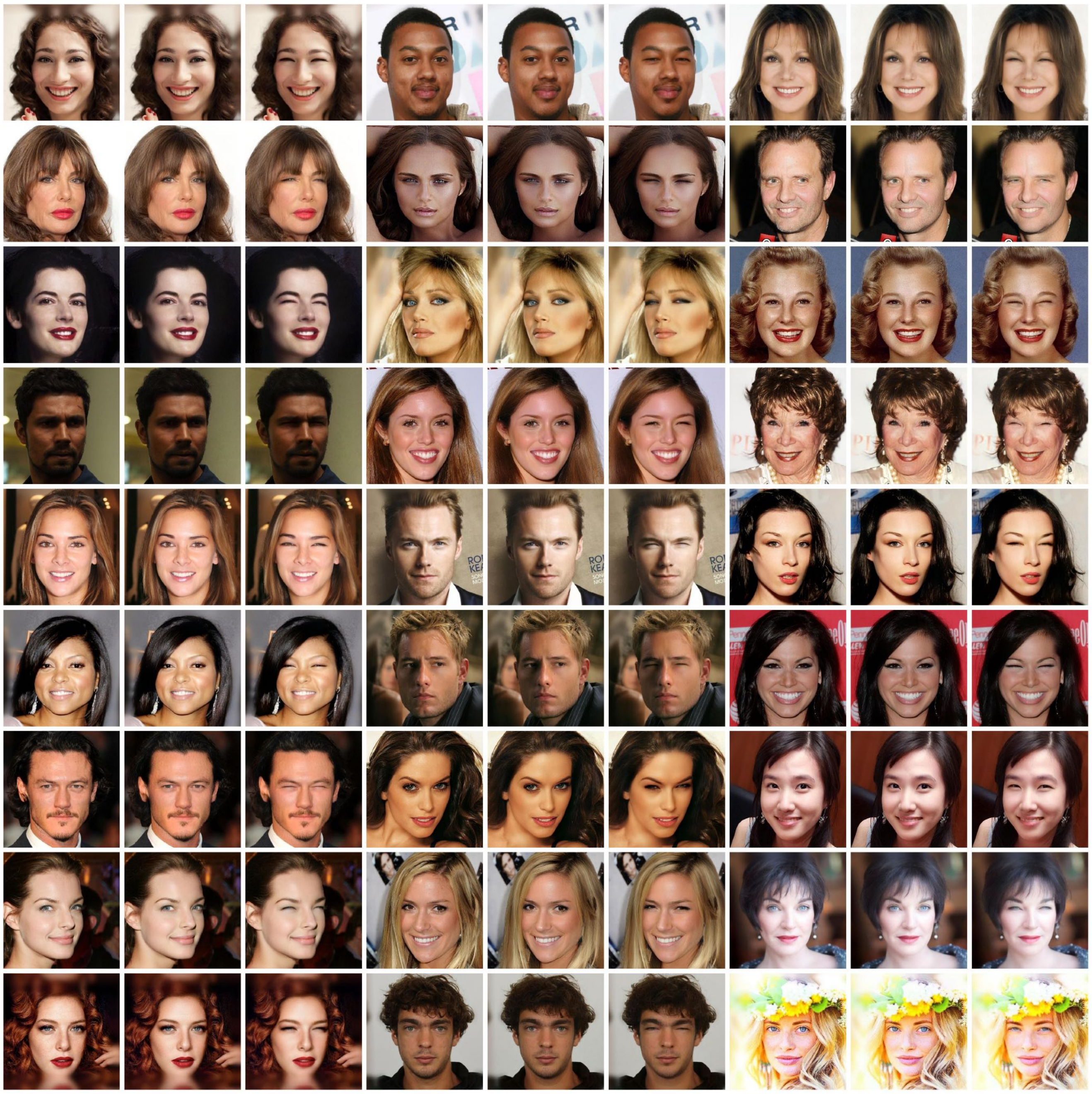

Figure 7. **Inversion and "Eye Closure" editing results.** The three columns represent **Input**, **Inversion**, and **Edit**.

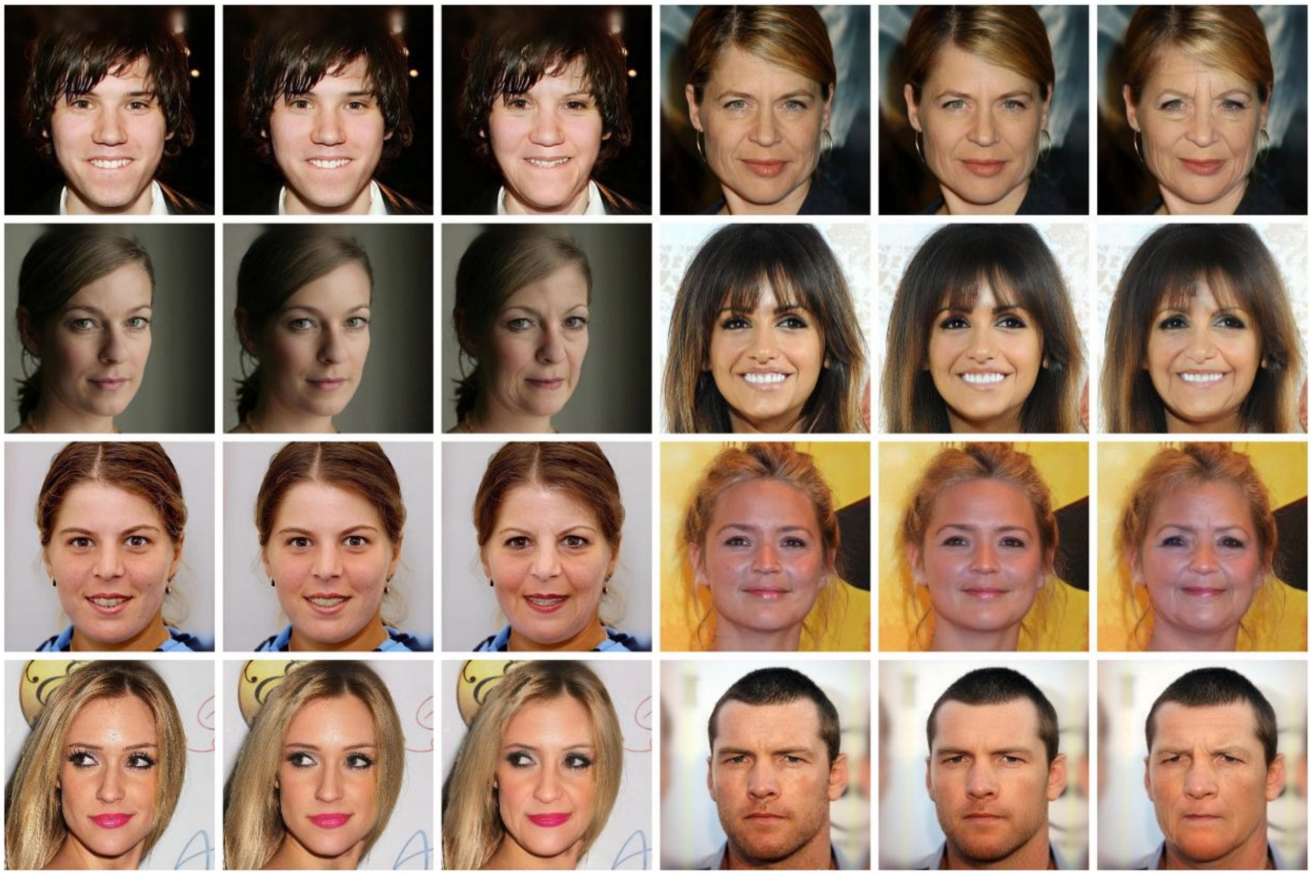

Figure 8. **Inversion and "Wrinkles" editing results.** The three columns represent **Input**, **Inversion**, and **Edit**.

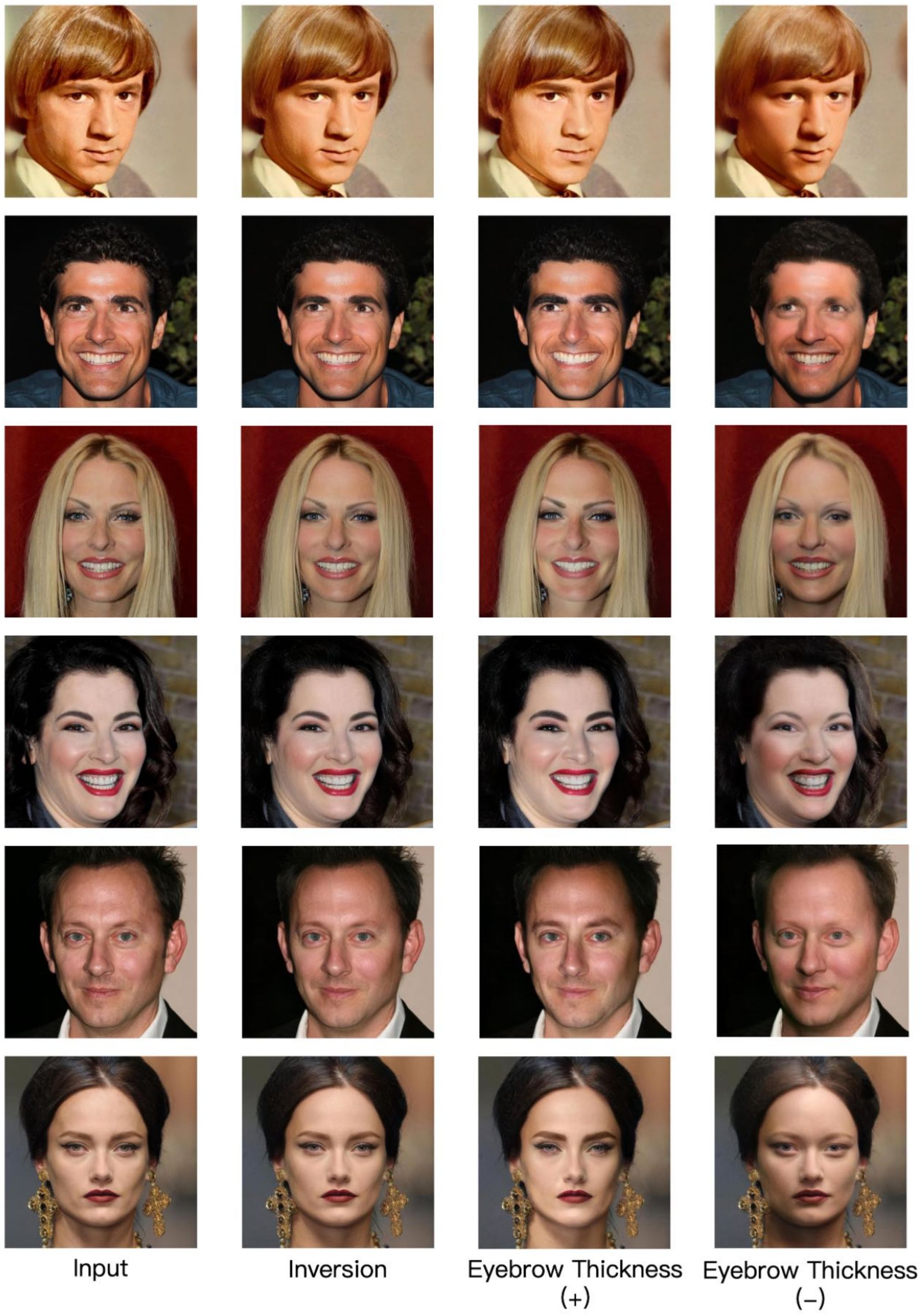

Figure 9. **Inversion and "Eyebrow Thickness" editing results.**