SemanticStyleGAN: Learning Compositional Generative Priors for Controllable Image Synthesis and Editing (Supplementary Material)

Yichun Shi Xiao Yang Yangyue Wan Xiaohui Shen

{yichun.shi,yangxiao.0,wanyangyue,shenxiaohui.kevin}@bytedance.com

ByteDance Inc., USA

https://SemanticStyleGAN.github.io

A. Implementation Details

A.1. Fusion with Transparent Classes

Following Sec 3.1 in the main paper, a coarse semantic mask $\mathbf{m}$ is fused from pseudo-depth maps, which is further used to aggregate local feature maps. In general cases, the aggregation can be simply achieved by computing $\mathbf{f} = \sum_k^K \mathbf{m}_k \odot \mathbf{f}_i$, where the frontal class in the semantic mask will be chosen for the output feature. However, in the case of transparent classes, this formulation could be problematic. For example, although the whole eye area could be labeled as glasses in the semantic masks, we are still able to see the skin behind it. Thus, we treat such transparent classes separately during feature aggregation. In particular, we use a modified mask:

$$\tilde{\mathbf{m}}_k(i,j) = \frac{\mathbb{1}_{NT}(k)\exp(\mathbf{d}_k(i,j))}{\sum_{k'}^K \mathbb{1}_{NT}(k')\exp(\mathbf{d}_{k'}(i,j))} + \mathbb{1}_T(k)\mathbf{m}_k(i,j), \tag{1}$$

where $\mathbb{1}_T(k)$ is an indicator function that equals 1 if k is a transparent class and 0 otherwise. $\mathbb{1}_{NT}(k)$ is the opposite indicator function for non-transparent classes. The first part of Eq. (1) here means that we first aggregate the features without the transparent classes. Then in the second part of Eq. (1), we add the transparent features using their their original weights in mask m. In this way, the feature map will not be affected if there are no transparent classes. If there are, they would be added onto the feature map as additional residuals. Note that this formulation assumes that transparent classes do not overlap with each other. In our experiments, we set glasses and earings as transparent glasses. In fact, the model can also be trained stably by simply using the original mask $\mathbf{m}$ for fusion, but the texture behind transparent classes could be distorted.

A.2. Architecture Details

As shown in Fig. 3 in the main paper, each **local generator** is a modulated MLP (implemented by 1×1 convolution) that has 10 layers. The input and output feature maps are both of size 64×64. All the hidden layers has 64 channels. The Fourier feature at the input is first transformed into the hidden feature map with a linear fully connected (FC) layer. The "toDepth" layer is a FC layer that outputs a 1-channel pseudo-depth map. The "toFeat" is a FC layer that outputs a 512-channel feature map. To encourage the disentanglement between shape and texture, we stop the gradient between shape and texture layers except for the background generator. We also fix the pseudo-depth map of background

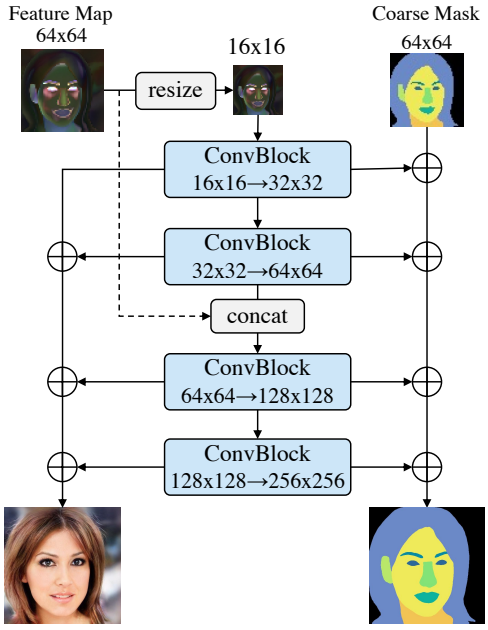

Figure 1. Details of the render net. Here, we take 256×256 model as the example. A "ConvBlock" is a StyleGAN2 convolution block that have 2 convolution layers. We remove the style modulation and add a linear segmentation output branch in each convolution layer. $\oplus$ indicates upsampling and summation.

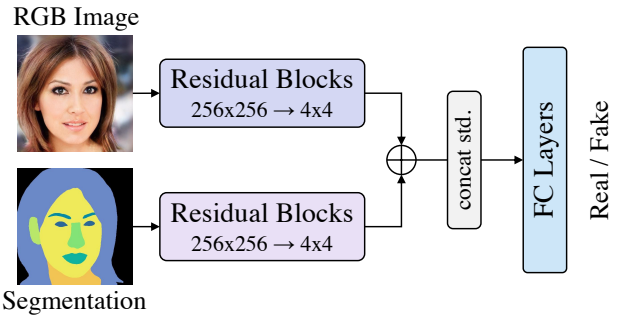

Figure 2. Details of the dual-branch discriminator. The "Residual Blocks" are the convolution layers. The image branch and segmentation are symmetric except the input channels. "concat std." is the step to of calculating standard deviation. The discriminator would be equivalent to StyleGAN2 discriminator if we remove the segmentation branch.

generator to be all 0s.

The detailed architecture of **render network** is shown in Fig. 1. Note that there is an upsampling and residual operation every layer for the segmentation mask, so $\Delta\mathbf{m}$ is not explicitly computed. Instead, we calculate $\mathcal{L}_{mask}$ by the difference between downsampled output segmentation and coarse mask $\mathbf{m}$.

The detailed architecture of **discriminator** is shown in Fig. 2. It is similar to StyleGAN2 discriminator except that we add an additional segmentation branch that is symmetric to image branch. During fine-tuning, we remove this branch and the discriminator reduces to an image discriminator.

A.3. Efficiency

The 256×256 and 512×512 models are trained on 4 and 8 32GB Tesla V100 GPUs, respectively. For the 512×512 model, our model takes about two and a half day to train 150,000 steps with a batch size of 32 on 8 32GB Nvidia Tesla V100 GPUs, where the best model is then selected. For inference, it takes 0.137s for our model to generate an image on a single GPU without parallelizing local generators.

B. Additional Discussion

Thanks to the reviewers, we provide additional discussions here to address some potentially shared concerns.

StyleSpace Wu *et al*. [9] showed that there exists an $\mathcal{S}$ space in the StyleGAN which is more locally disentangled than the $W+$ space that we used in the main paper. However, in spite of good editing results, their method can only control limited attributes by tuning individual feature channels. It does not learn additional attributes from given labels. It is not proved yet that one can achieve the same degree of local disentanglement if latent editing methods are

Figure 3. Results of InterFaceGAN on the $\mathcal{S}$ space of StyleGAN2.

Figure 4. Mask-conditioned models [69] can only transfer texture.

applied to the whole $\mathcal{S}$ space. Here, we briefly conduct such an experiment by applying InterFaceGAN to the $\mathcal{S}$ space of the original StyleGAN2. As shown in Fig. 3, $\mathcal{S}$ space still could suffer from spatial entanglement.

Mask-conditioned Models and StyleMapGAN Mask-conditioned image translation models, such as SEAN [10] can be applied to local editing since they also learn a latent space for each semantic area. However, conditioned on a fixed semantic mask, their editing is restricted to the texture of each area (See Fig. 4). Changing the shape would require manual effort. In contrast, our unconditional model can control **both the shape and texture** with latent codes. We also note that our model, without using segmentation inputs, achieves a much lower FID (6.43) on CelebA-HQ than [10] (17.66) and [8] (22.43). StyleMapGAN [5] has also shown the ability of local editing on synthesized images. However, by using a stylemap pyramid, it requires the editing area as an input, and the editing only happens to the texture in that area, a similar problem as SEAN. In contrast, our method is automatic and not restricted to fixed pixels.

C. Style Mixing and Additional Results

In the main paper, we showed that the proposed model can interpolate smoothly in a local latent space. Here, we show results on more fine-grained style mixing using our model. Different from StyleGANs [2–4], we can conduct style mixing between local generators to transfer a certain semantic component from one image to another. This conducted by transferring both the shape code $\mathbf{w}_s^k$ and texture

code $\mathbf{w}_t^k$. Fig. 5 shows the results of semantic style mixing using our model trained on CelebAMask-HQ [6]. Besides the local latent codes, we also show the transferring results of the coarse structure code $\mathbf{w}_{base}$. It can be seen that our model is able to transfer most local component styles between images, including small components such as eyes and mouth. However, it is also observable that the coarse structure code is currently encoding some information about these local components, such as expression and hair. Although a user or developer is able to change the number of coarse structure codes dynamically during testing (and even manipulate all the layers in a local generator), we believe it would be beneficial to further regularize the information in the coarse code in the future. Fig. 6 and Fig. 7 show the semantic style mixing results of a model after transfer learning (on BitMoji dataset [1]) and the model trained on DeepFashion dataset [7]. A similar effect can be seen on the DeepFashion that the coarse structure would affect certain components. Also, we see that sometimes the hair color is affected by the background on this dataset. Since the head in this dataset is rather small, we believe such entanglement is caused by the low-resolution (16×16) feature map that was fed into render network for blending, which is originally selected for face datasets. Further tuning the hyper-parameters of the render net might alleviate such issues.

Fig. 8 and Fig. 9 show more results on randomly sampled images and pseudo-depth maps, respectively Figs. 10 to 13 show more results on real face editing using our model and original StyleGAN2. As mentioned in the main paper, we see that StyleFlow is more sensitive to data imbalance and less robust. Taking bangs for instance, it tries to reduce the hair on the side but not in the front for our model. For beard, it tries to make face skin look darker for our model while completely fails on the original StyleGAN2. Note that we re-train both StyleFlow and InterFaceGAN using newly sampled images and our own attribute prediction model. Overall, we can observe that our model achieves much more localized control when editing output images.

References

[1] Bitmoji dataset. https://www.kaggle.com/mostafamozafari/bitmoji-faces/version/1, 2013. Released under CC BY 4.0. 3

[2] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. *arXiv:2106.12423*, 2021. 2

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

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

[5] Hyunsu Kim, Yunjey Choi, Junho Kim, Sungjoo Yoo, and Youngjung Uh. Exploiting spatial dimensions of latent in gan for real-time image editing. In *CVPR*, 2021. 2

[6] Cheng-Han Lee, Ziwei Liu, Lingyun Wu, and Ping Luo. Maskgan: Towards diverse and interactive facial image manipulation. In *CVPR*, 2020. 3

[7] Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *CVPR*, 2016. 3

[8] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *CVPR*, 2019. 2

[9] Zongze Wu, Dani Lischinski, and Eli Shechtman. Stylespace analysis: Disentangled controls for stylegan image generation. In *CVPR*, 2021. 2

[10] Peihao Zhu, Rameen Abdal, Yipeng Qin, and Peter Wonka. Sean: Image synthesis with semantic region-adaptive normalization. In *CVPR*, 2020. 2

Figure 5. Local style mixing of the model trained on CelebAMask-HQ. The first column shows randomly sampled images for editing. The remaining columns show the results of mixing local styles using the reference images in the first row.

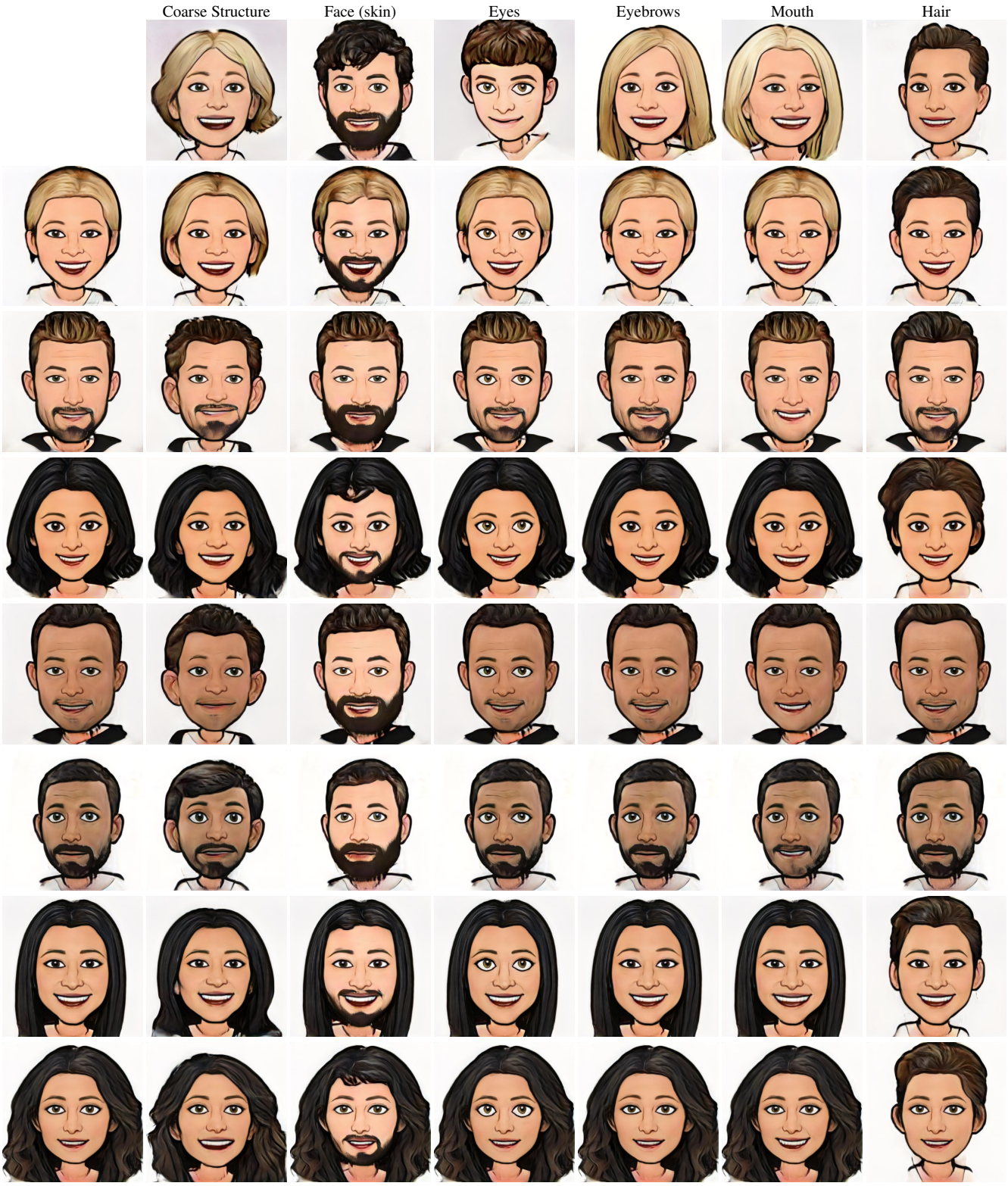

Figure 6. Local style mixing of the model fine-tuned on the BitMoji dataset. The first column shows randomly sampled images for editing. The remaining columns show the results of mixing local styles using the reference images in the first row.

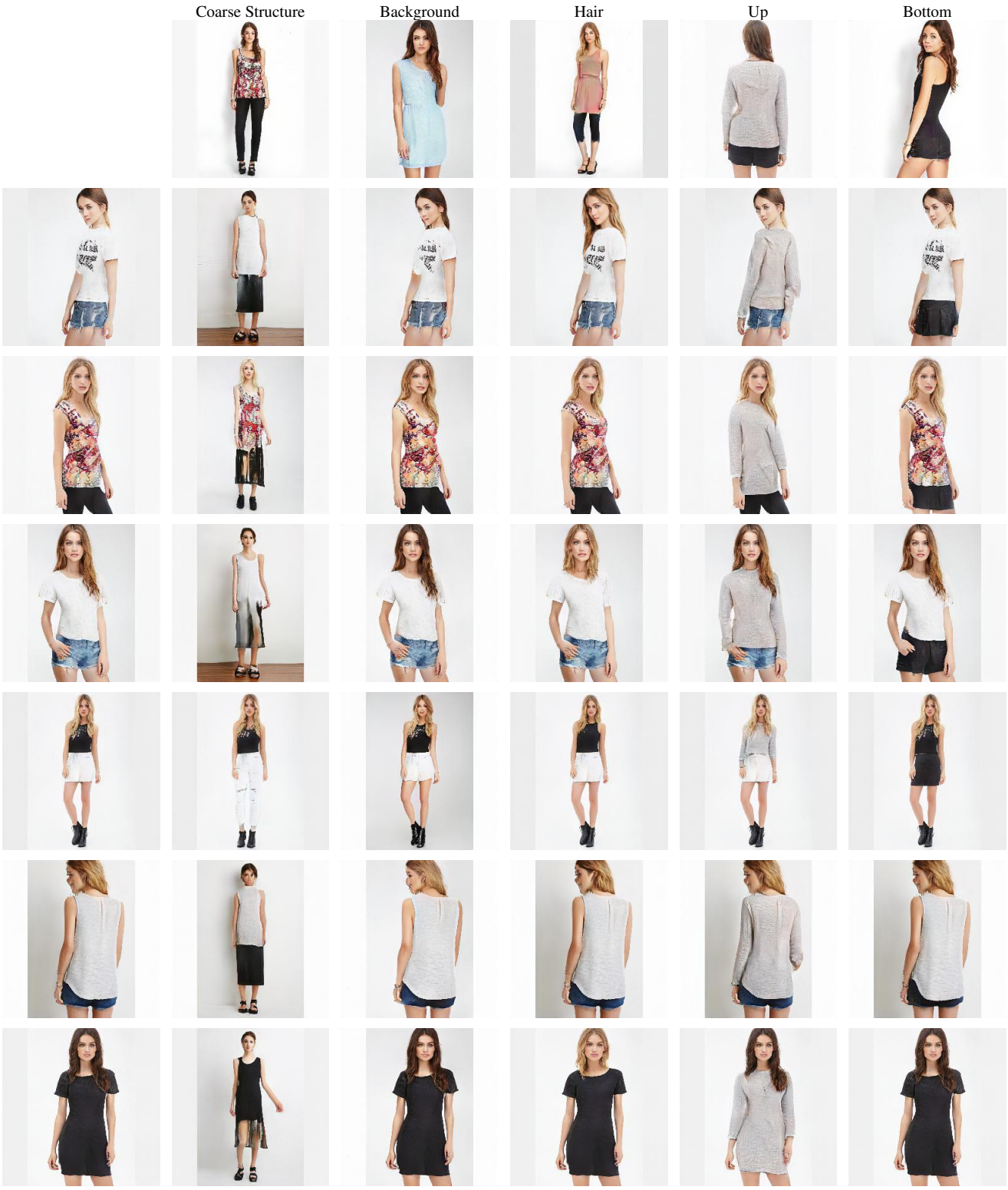

Figure 7. Local style mixing of the model trained on the DeepFashion dataset. The first column shows different randomly sampled images for editing. The remaining columns show the results of mixing local styles using the reference images in the first row.

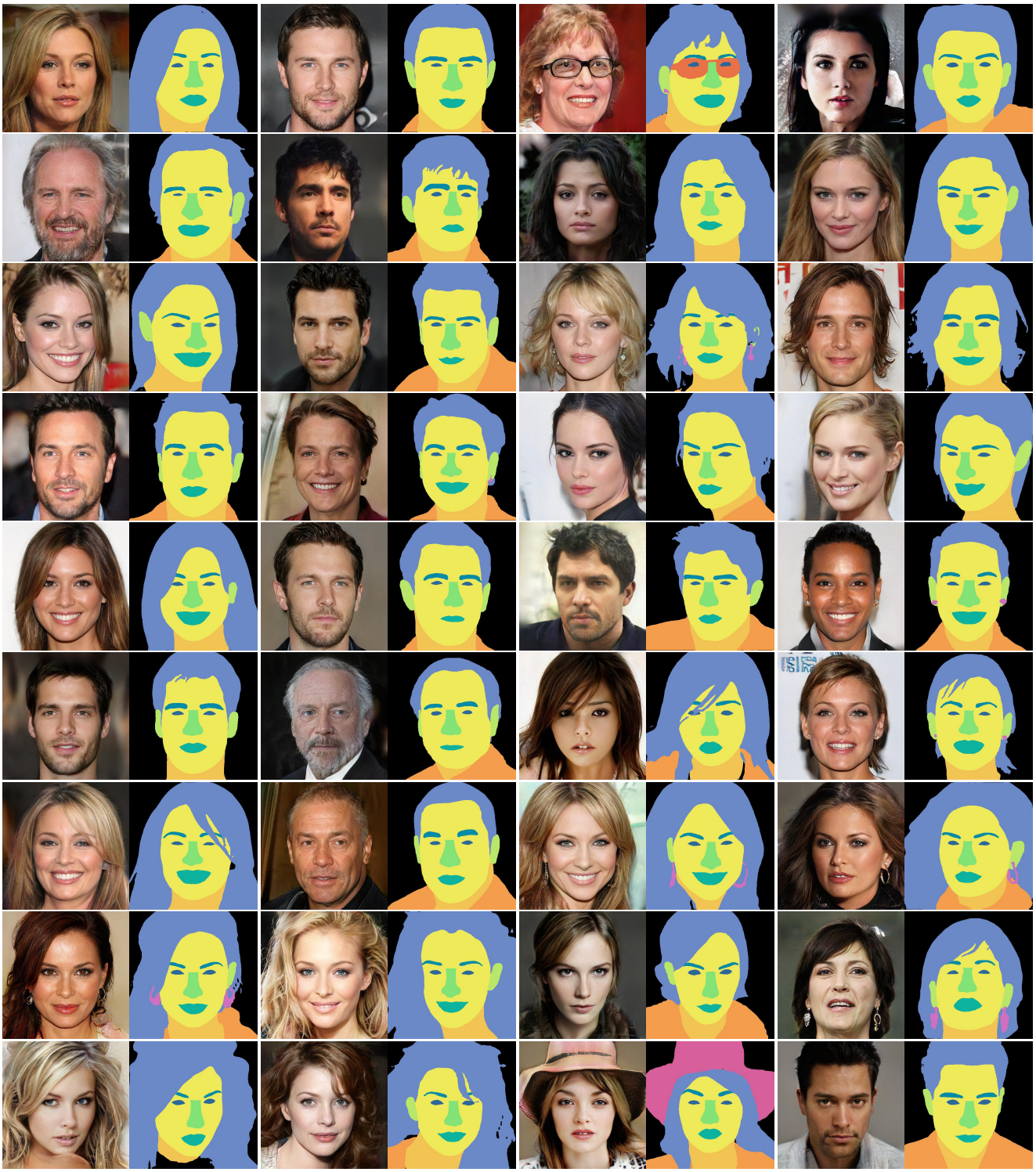

Figure 8. Example generated images and using our 512×512 trained on CelebAMask-HQ. On the right of each generated photo is the refined segmentation mask output by the model.

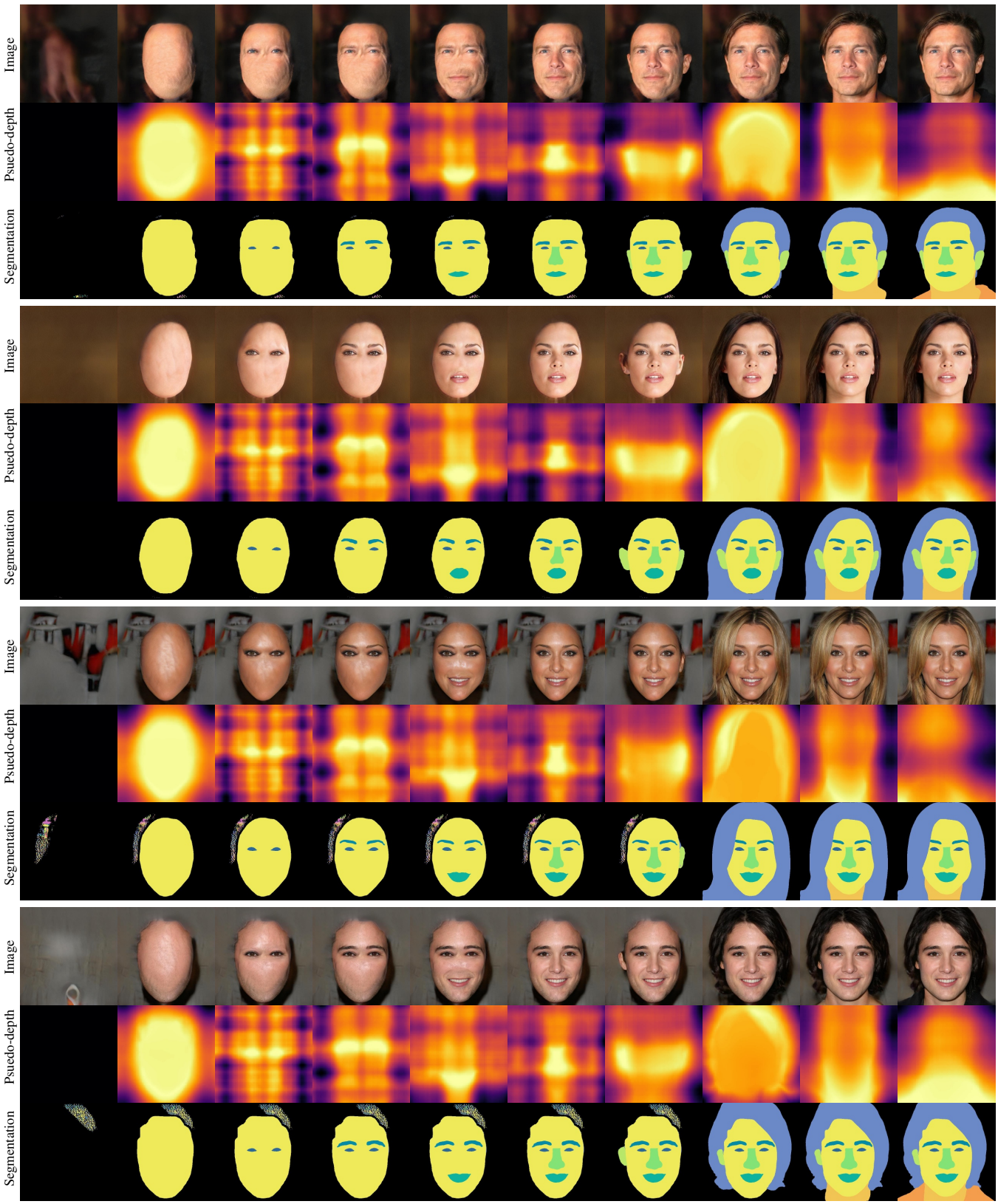

Figure 9. Illustration of compositional synthesis. Starting from background, we gradually add more components into the feature map. The second row of each sample shows the pseudo-depth map of each corresponding component used for fusion. During synthesis, all pseudo-depth maps are fused without an order.

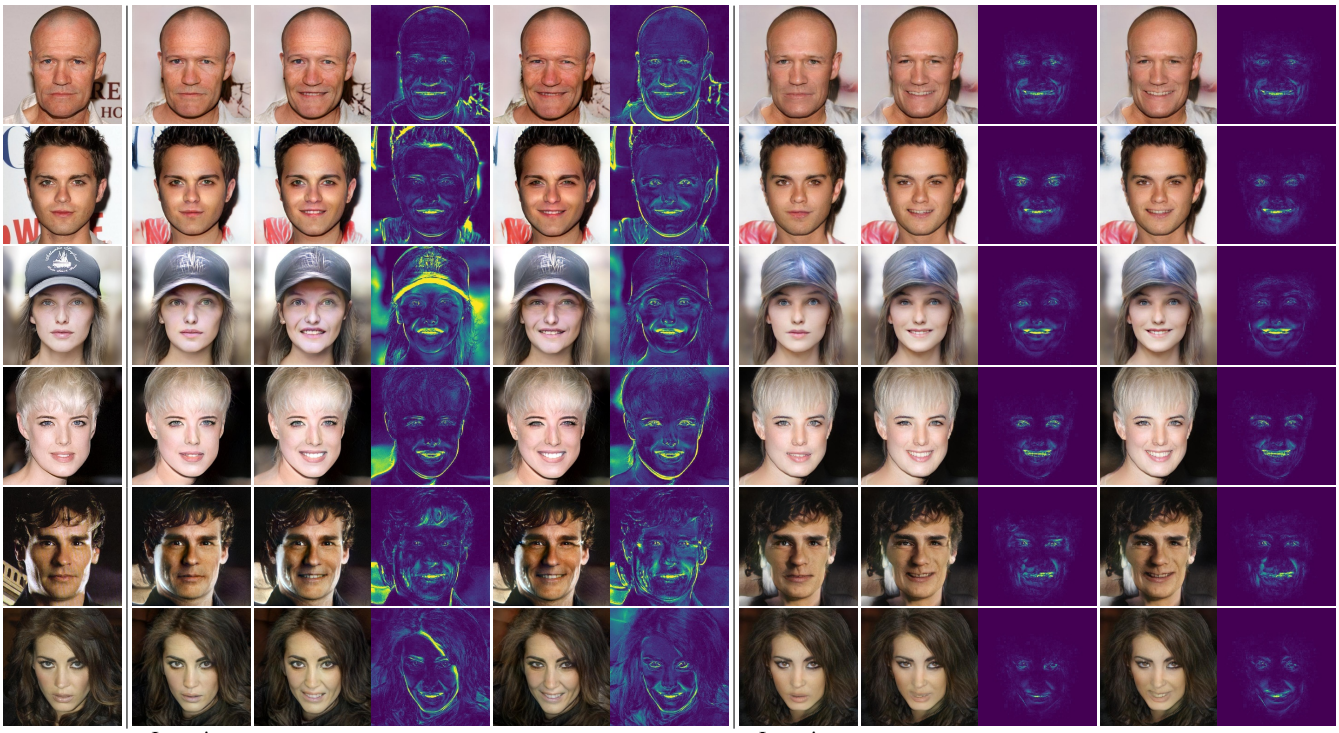

Figure 10. Results of GAN inversion and editing for the **smile** attribute. For each method, we show the inversion result of Restyle encoder, the edited image and the difference map between them.

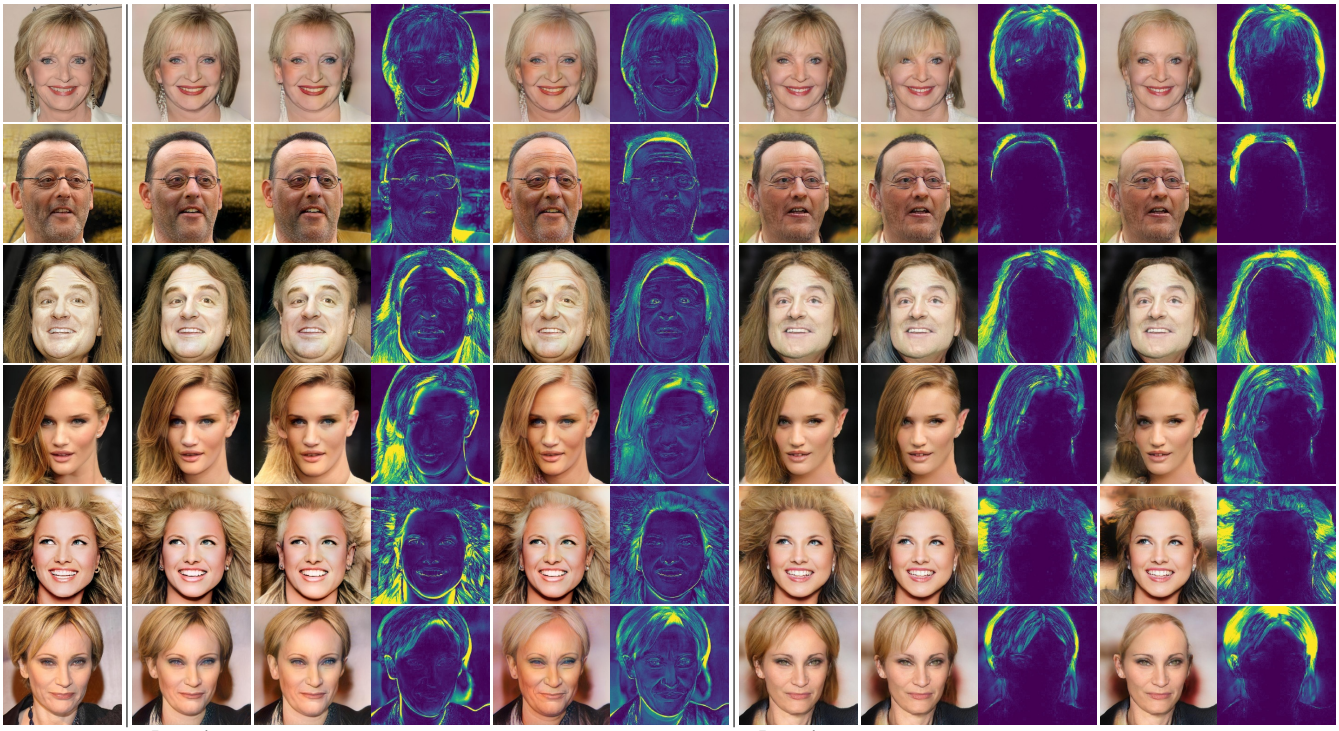

Figure 11. Results of GAN inversion and editing for the **bald** attribute. For each method, we show the inversion result of Restyle encoder, the edited image and the difference map between them.

Figure 12. Results of GAN inversion and editing for the **bangs** attribute. For each method, we show the inversion result of Restyle encoder, the edited image and the difference map between them.

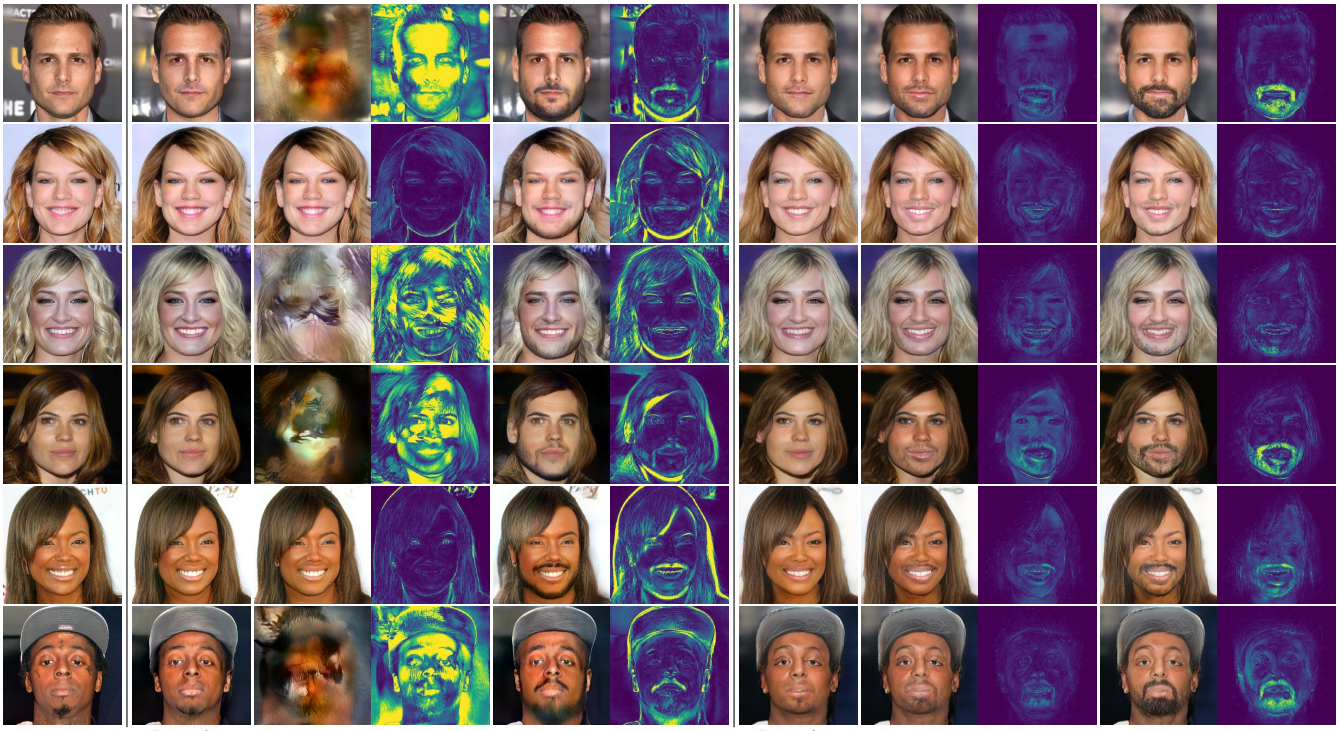

Figure 13. Results of GAN inversion and editing for the **beard** attribute. For each method, we show the inversion result of Restyle encoder, the edited image and the difference map between them.