Image Synthesis via Semantic Composition
Supplementary Material

In this supplementary file, our descriptions contain the following components:

- Detailed configuration of our proposed SC-GAN and the implementation of the given spatially conditional convolution and normalization.
- More semantic image synthesis results on the four benchmarks.
- The synthesis performance of our generator trained with a more effective discriminator and other tricks.
- The specification of how to apply our framework to unpaired image-to-image translation and the corresponding visual results.
- The limitations and some failure cases of our method.

Code is available at https://github/dvlab-research/SCGAN.

1. Network Architectures

SC-GAN consists of Semantic Vector Generator (SVG) and Semantic Render Generator (SRG). Their detailed designs are given blew. For convenience, we suppose Conv(k, s, c) indicates a convolutional operation whose kernel size, stride size, and output channel number of the used convolution are k, s, and c, respectively. The dilation ratio and padding size of Conv(k, s, c) are both set to 1. $\uparrow$ and $\otimes$ denotes a $2\times$ bilinear upsampling and concatenation (along with the channel dimension) operations, respectively. Besides, SCResBlock(k, s, c) denotes a residual block variant using spatially conditional convolution (SCC) and normalization (SCN). Its schematic illustration is given in Figure 4 of our paper. We use $\mathbf{y}[+\mathbf{x}]$ to indicate an extra input $\mathbf{x}$ of the current operation $\mathbf{y}$. For example, SCResBlock(3,1,512)$[+\mathbf{V}_i]$ indicates the semantic vectors $\mathbf{V}_i$ (in feature maps form) is also incorporated into SCResBlock(3,1,512) for generating dynamic operators.

SVG : $\mathbf{S} \downarrow \rightarrow$ Conv(3,1,512) $\rightarrow$ LReLU $\rightarrow$ Conv$_1$(3,1,512) $\rightarrow\uparrow\rightarrow\otimes(\mathbf{S}\downarrow)\rightarrow$ LReLU $\rightarrow$ Conv (3,1,256) $\rightarrow$ LReLU $\rightarrow$ Conv$_2$(3,1,256) $\rightarrow\uparrow\rightarrow\otimes(\mathbf{S}\downarrow)\rightarrow$ LReLU $\rightarrow$ Conv(3,1,128) $\rightarrow$ LReLU $\rightarrow$ Conv$_3$(3,1,128) $\rightarrow\uparrow\rightarrow\otimes(\mathbf{S}\downarrow)\rightarrow$ LReLU $\rightarrow$ Conv(3,1,64) $\rightarrow$ LReLU $\rightarrow$ Conv$_4$(3,1,64) $\rightarrow\uparrow\rightarrow\otimes(\mathbf{S}\downarrow)\rightarrow$ LReLU $\rightarrow$ Conv(3,1,32) $\rightarrow$ LReLU $\rightarrow$ Conv$_5$(3,1,32) $\rightarrow\uparrow\rightarrow\otimes(\mathbf{S}\downarrow)\rightarrow$ LReLU $\rightarrow$ Conv(3,1,32) $\rightarrow$ LReLU $\rightarrow$ Conv$_6$(3,1,32) $\rightarrow\uparrow\rightarrow\otimes(\mathbf{S})$ $\rightarrow$ LReLU $\rightarrow$ Conv(3,1,16) $\rightarrow$ LReLU $\rightarrow$ Conv(3,1,3) $\rightarrow$ Hardtanh $\rightarrow f_{\mathrm{V}}^{\mathrm{out}}(\mathbf{S})$,
where $\mathbf{S}$ indicates the input segmentation mask.

SRG : $z \rightarrow$ SCResBlock(3,1,512) $[+\mathbf{V}_1]\rightarrow\uparrow\rightarrow$ SCResBlock(3,1,512) $[+\mathbf{V}_2]\rightarrow\uparrow\rightarrow$ SCResBlock(3,1,512) $[+\mathbf{V}_2]\rightarrow\uparrow\rightarrow$ SCResBlock(3,1,256) $[+\mathbf{V}_3]\rightarrow\uparrow\rightarrow$ SCResBlock(3,1,128) $[+\mathbf{V}_4]\rightarrow\uparrow\rightarrow$ SCResBlock(3,1,64) $[+\mathbf{V}_5]\rightarrow\uparrow\rightarrow$ SCResBlock(3,1,32) $[+\mathbf{V}_6]\rightarrow\uparrow\rightarrow$ Conv (k3c3) $\rightarrow$ Hardtanh $\rightarrow \hat{\mathbf{I}}$,
where z is sampled from a standard normal distribution, $\mathbf{V}_i \dashleftarrow$ Conv$_i$ from SVG, and $\dashleftarrow$ denotes the adaptive pooling operation (in channel dimension).

1.1. Implementation of Key Components

The pseudo codes to realize the proposed spatially conditional convolution (SCC) and normalization (SCN) are given in Alg 1 and 2, respectively. In general, they are designed that the regional parameterized weights are generated by combining a group of candidates, and the manner how they are combined is indicated by the fed semantic vectors (e.g. the following $\mathbf{V}$).

2. More Experimental Results and Analysis

We give more visual comparisons on CelebAMask-HQ [3] (Figure 2, 3, and 4), Cityscapes [2] (Figure 5), ADE20K [7] (Figure 6, 7, 8, and 9), and COCO-Stuff [1] (Figure 10, 11, 12, and 13). Also, more multi-modal outputs (Figure 14 and 15) and interpolations (Figure 16 and 17) on CelebAMask-HQ are given, as well.

2.1. Performance with a Stronger Discriminator and Training Tricks

We evaluate the compatibility between our proposed generator and a newly introduced discriminator [6], along with some effective training techniques. Sushko *et al*. presented a powerful discriminator exploiting semantic layouts

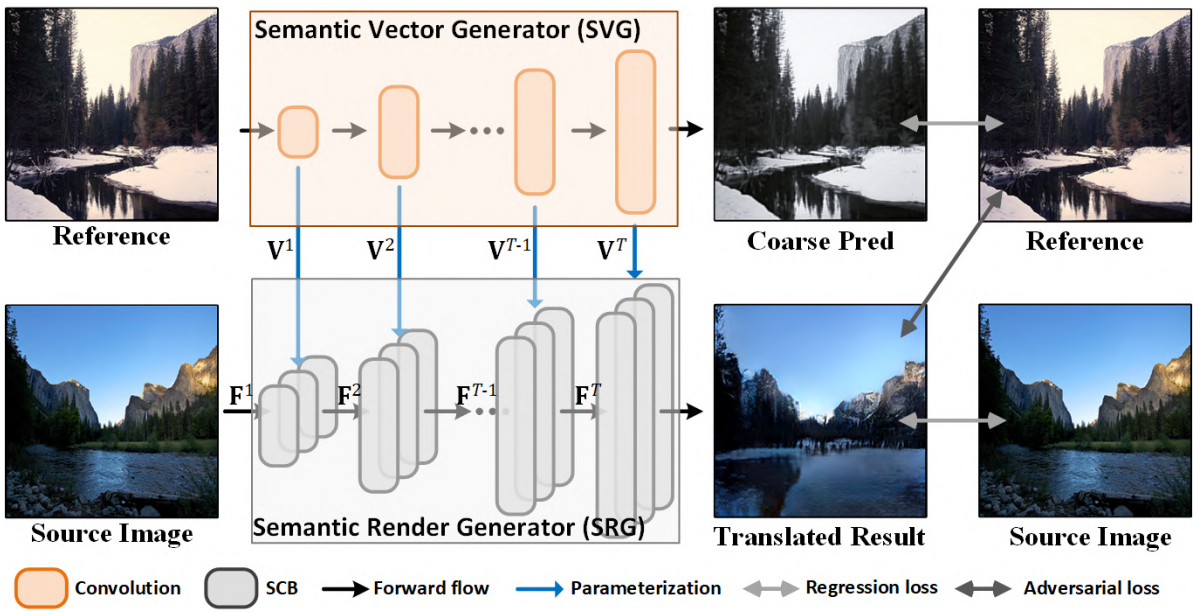


Figure 1. The framework of SC-GAN for unpaired image-to-image translation.

Algorithm 1 The pseudo code of SCC (PyTorch style)

Input: Input feature maps $\mathbf{F} \in \mathcal{R}^{b\times c_{\text{in}}\times h\times w}$ and the generated semantic vectors $\mathbf{V} \in \mathcal{R}^{b\times n\times h\times w}$, and learnable parameter candidates $\{\mathbf{k}_i\}$ where $\mathbf{k}_i \in \mathcal{R}^{c_{\text{in}}\times ks\times ks\times c_{\text{out}}}$, and c_{in}, c_{out}, and ks denote the input, output channel number, and kernel size.

Output: The convolved feature maps $\hat{\mathbf{F}} \in \mathcal{R}^{b\times c_{\text{out}}\times h\times w}$.

$\mathbf{V} = \mathbf{V}$.unsqueeze(2) # shape: $b \times n \times 1 \times h \times w$
out $= [0] * n$
for $i = 1$ to n **do**
    out$[i] = \mathbf{k}_i(\mathbf{F})$.unsqueeze(1)
end for
$\hat{\mathbf{F}} =$ torch.cat($\mathbf{F}$, dim $= 1$) # shape: $b\times n\times c_{\text{out}}\times h\times w$
$\hat{\mathbf{F}} = \hat{\mathbf{F}} * \mathbf{V}$
$\hat{\mathbf{F}} =$ torch.sum($\hat{\mathbf{F}}$, dim $= 1$) # shape: $b \times c_{\text{out}} \times h \times w$

Algorithm 2 The pseudo code of SCN (PyTorch style)

Input: Input feature maps $\mathbf{F} \in \mathcal{R}^{b\times c_{\text{in}}\times h\times w}$ and the generated semantic vectors $\mathbf{V} \in \mathcal{R}^{b\times n\times h\times w}$, and learnable parameter candidates $\mathbf{A}$, where $\mathbf{A} \in \mathcal{R}^{n\times 2c_{\text{in}}}$.

Output: The normalized feature maps$\hat{\mathbf{F}} \in \mathcal{R}^{b\times c_{\text{in}}\times h\times w}$.

$\mathbf{V} = \mathbf{V}$.permute$(0, 2, 3, 1)$.contiguous() .view$(-1, n)$ # shape: $b \times n \times h \times w \rightarrow bhw \times n$
$\hat{\mathbf{F}} =$ BN($\mathbf{F}$) # shape: $b \times c_{\text{in}} \times h \times w$
$\mathbf{A} =$ torch.matmul$(\mathbf{V}, \mathbf{A})$.view$(b, h, w, -1)$.permute $(0, 3, 1, 2)$.contiguous() # shape: $b \times 2c_{\text{in}} \times h \times w$
$\mathbf{m}, \mathbf{s} =$ torch.split$(\mathbf{A}, n,$ dim $= 1)$
$\hat{\mathbf{F}} = \hat{\mathbf{F}} * (1 + \mathbf{s}) + \mathbf{m}$

Table 1. Quantitative results on the validation sets of Cityscapes and ADE20K from different methods. Ours w OASIS denotes our generator is trained with the discriminator and other techniques from OASIS [6].

Method	Cityscapes		ADE20K	
	mIoU ↑	FID ↓	mIoU ↑	FID ↓
SPADE [5]	62.3	71.8	38.5	33.9
CC-FPSE [4]	65.6	54.3	43.7	31.7
OASIS [6]	69.3	47.7	48.8	28.3
Ours	66.9	49.5	45.2	29.3
Ours w OASIS	**69.9**	**47.2**	**49.1**	**27.6**

(OASIS) by semantic segmentation loss. Their approach further strengthens GAN training by 1) balancing the class weights by their frequencies, 2) removing VGG loss, and 3) exponential moving average (EMA) model merging (for generator). During training, they mask generated regions with a random class and add 3D random noise to all input segmentation masks to enhance local detail synthesis. By integrating their techniques (except 3D random noises), our model can be further improved over performance, as given in Table 1. Compared with OASIS [6], our proposed generator yields better quantitative results with a smaller capacity (66.2M (ours) vs. 94M (OASIS)).

2.2. Unpaired Image-to-image Translation

As we claimed in the paper, our method is also applicable to unpaired image-to-image translation applications with minor modifications. Its corresponding framework is presented in Figure 1. Specifically, changing the input of

SRG from the random noise to the downsampled image from the source domain, then the output of SRG should be close to its input one in semantic layout (using perceptual loss), and similar to images from the target domain in style (texture, detail, etc). The input of SVG is set to the image from the target domain, and it still regresses to the input image like an autoencoder. Figure 18 gives visual results from our model on summer→winter dataset [8], in which our proposed method can alter the source image style by changing its color and texture according to the reference.

2.3. Limitations and Failure Cases

Although our design enhances generation performance by explicitly learning the relations between different semantics, it may fail to fully recover the intrinsic geometry in the original image when the given segmentation map is short of such cues (Figure 19). This is a common issue in numerous generative models, usually addressed by introducing extra modal (*e.g*. depth) or multi-view data.

Moreover, the explicit modeling between semantics by their appearances may lead to creating undesired objects/stuff in the target semantic region, as given in Figure 19. Though the details in building regions are vivid, unexpected plants are synthesized in these areas. We suppose it is caused by that our model learns such symbiotic bias between these two kinds of stuff in the training data. Utilizing segmentation masks to further constrain semantic vectors (*e.g*. adding semantic segmentation loss to SVG) may address this issue. We will study it in the future.

References

[1] Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. Cocostuff: Thing and stuff classes in context. In *CVPR*, pages 1209–1218, 2018. 1

[2] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, pages 3213–3223, 2016. 1

[3] Cheng-Han Lee, Ziwei Liu, Lingyun Wu, and Ping Luo. Maskgan: Towards diverse and interactive facial image manipulation. In *CVPR*, pages 5549–5558, 2020. 1

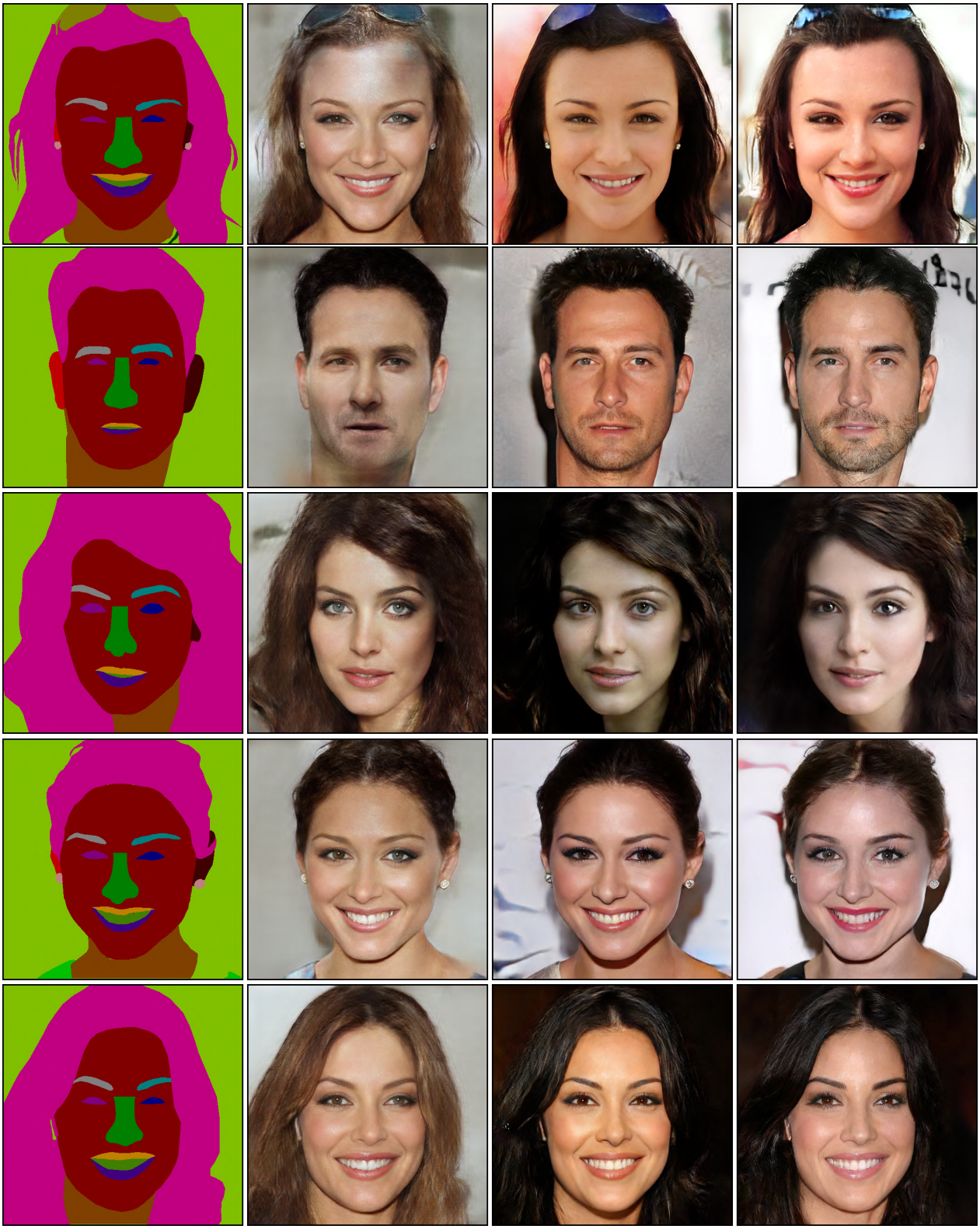
[4] Xihui Liu, Guojun Yin, Jing Shao, Xiaogang Wang, et al. Learning to predict layout-to-image conditional convolutions for semantic image synthesis. In *NeurIPS*, pages 570–580, 2019. 2

[5] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *CVPR*, pages 2337–2346, 2019. 2

[6] Vadim Sushko, Edgar Schönfeld, Dan Zhang, Juergen Gall, Bernt Schiele, and Anna Khoreva. You only need adversarial supervision for semantic image synthesis. *arXiv preprint arXiv:2012.04781*, 2020. 1, 2

[7] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *CVPR*, pages 633–641, 2017. 1

[8] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, pages 2223–2232, 2017. 3

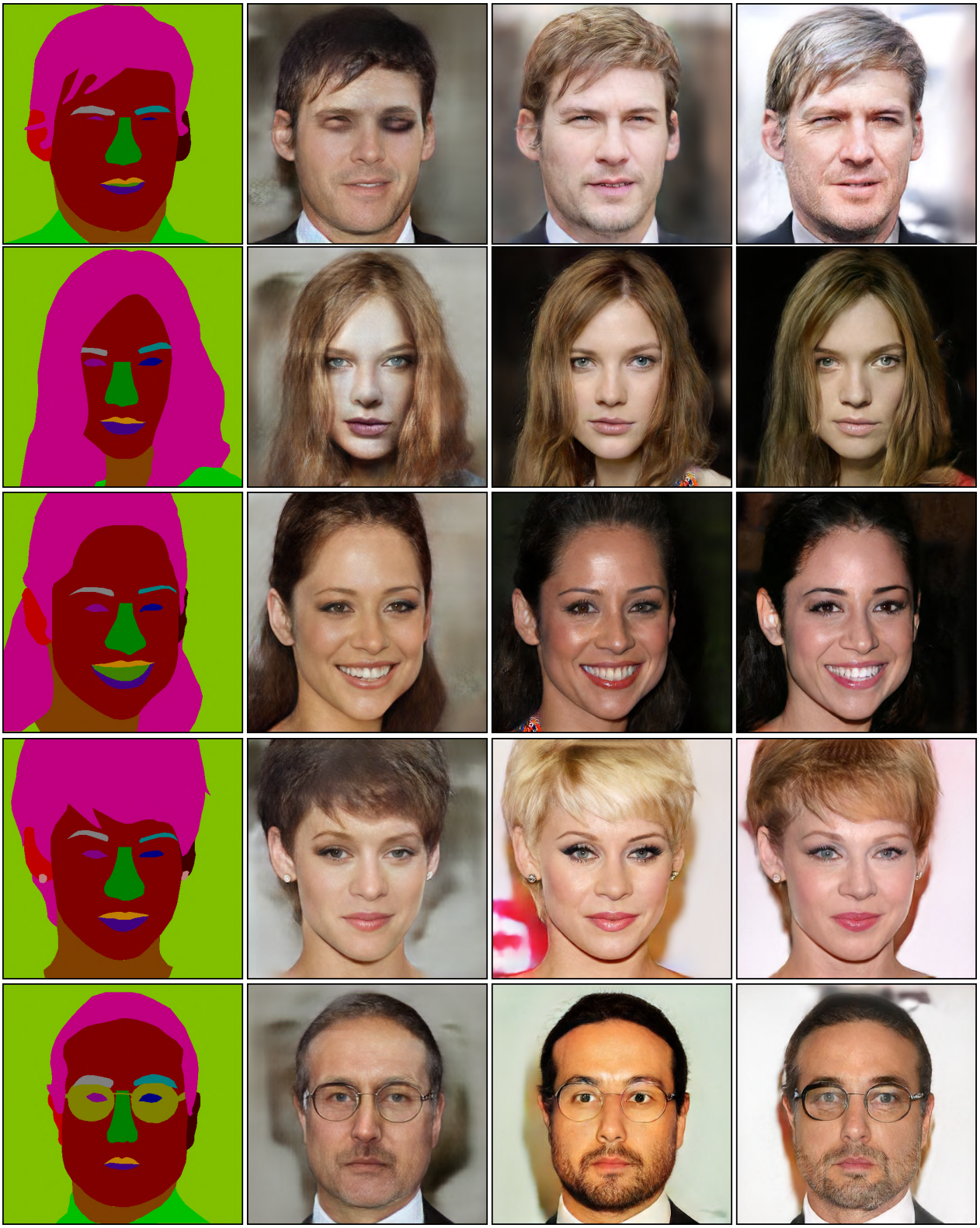


(1) The input (2) SPADE (3) MaskGAN (4) Ours.

Figure 2. Visual comparisons on CelebAMask-HQ.

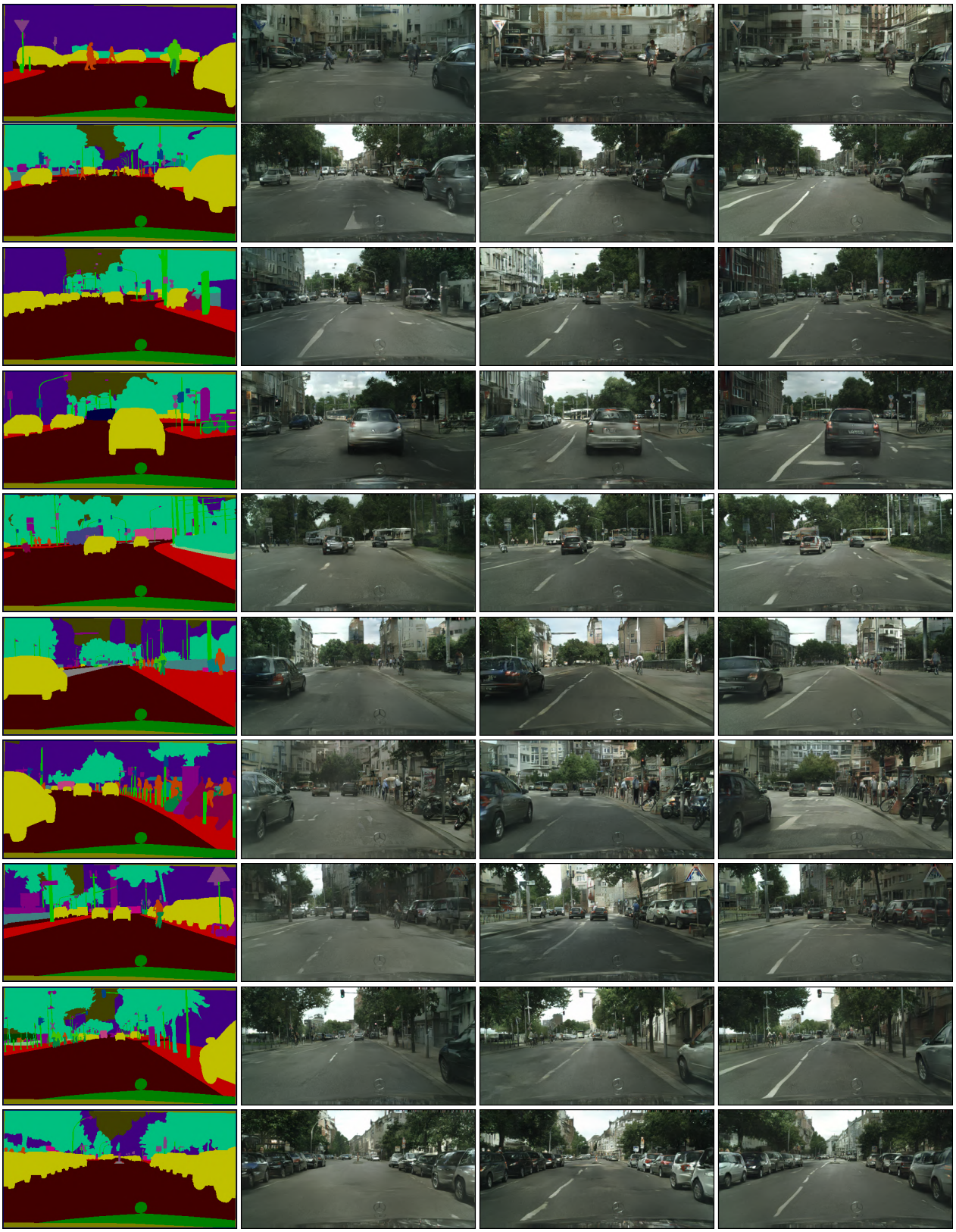
(1) The input (2) SPADE (3) MaskGAN (4) Ours.

Figure 3. Visual comparisons on CelebAMask-HQ.



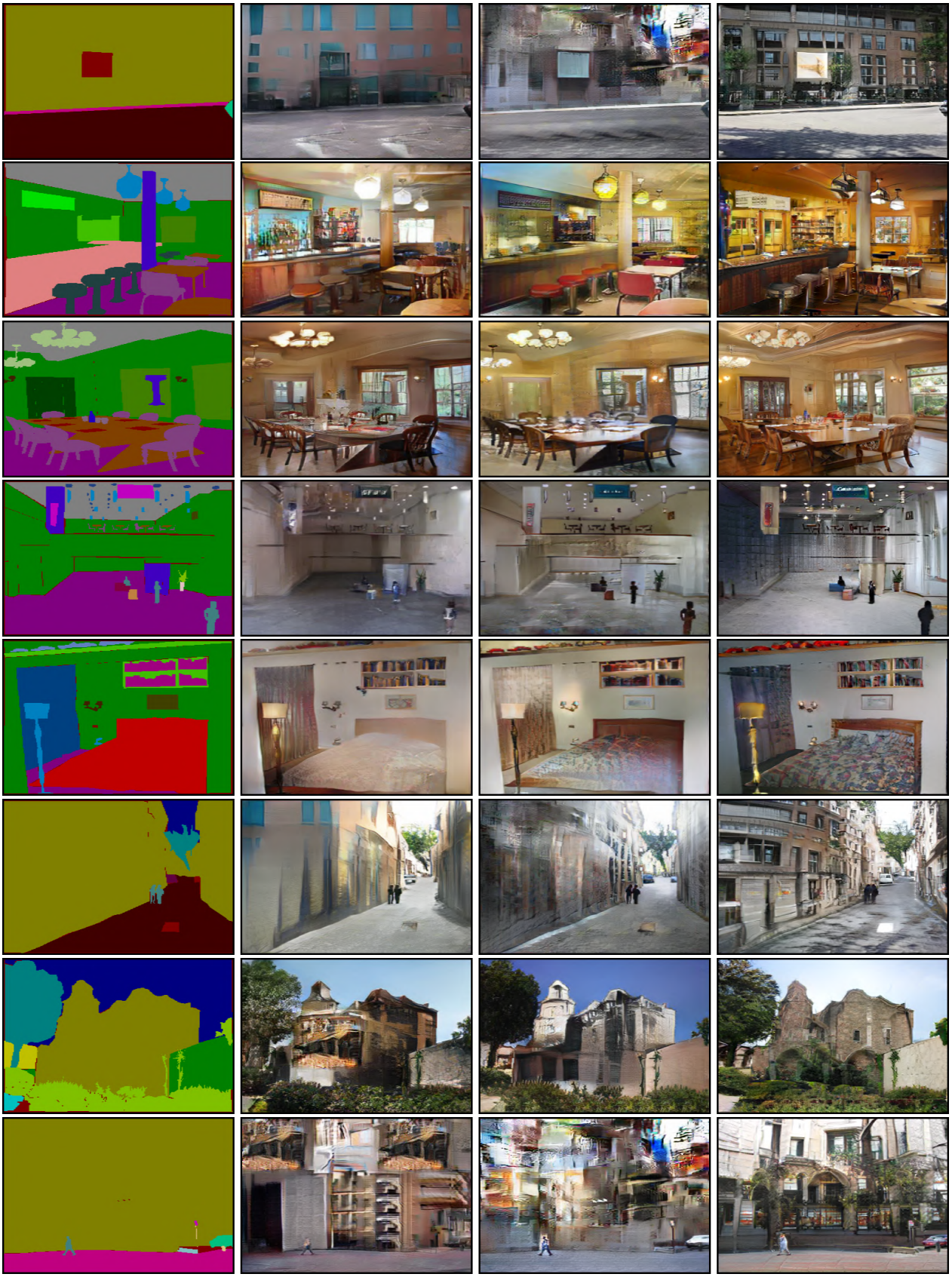
(1) The input (2) SPADE (3) MaskGAN (4) Ours.

Figure 4. Visual comparisons on CelebAMask-HQ.



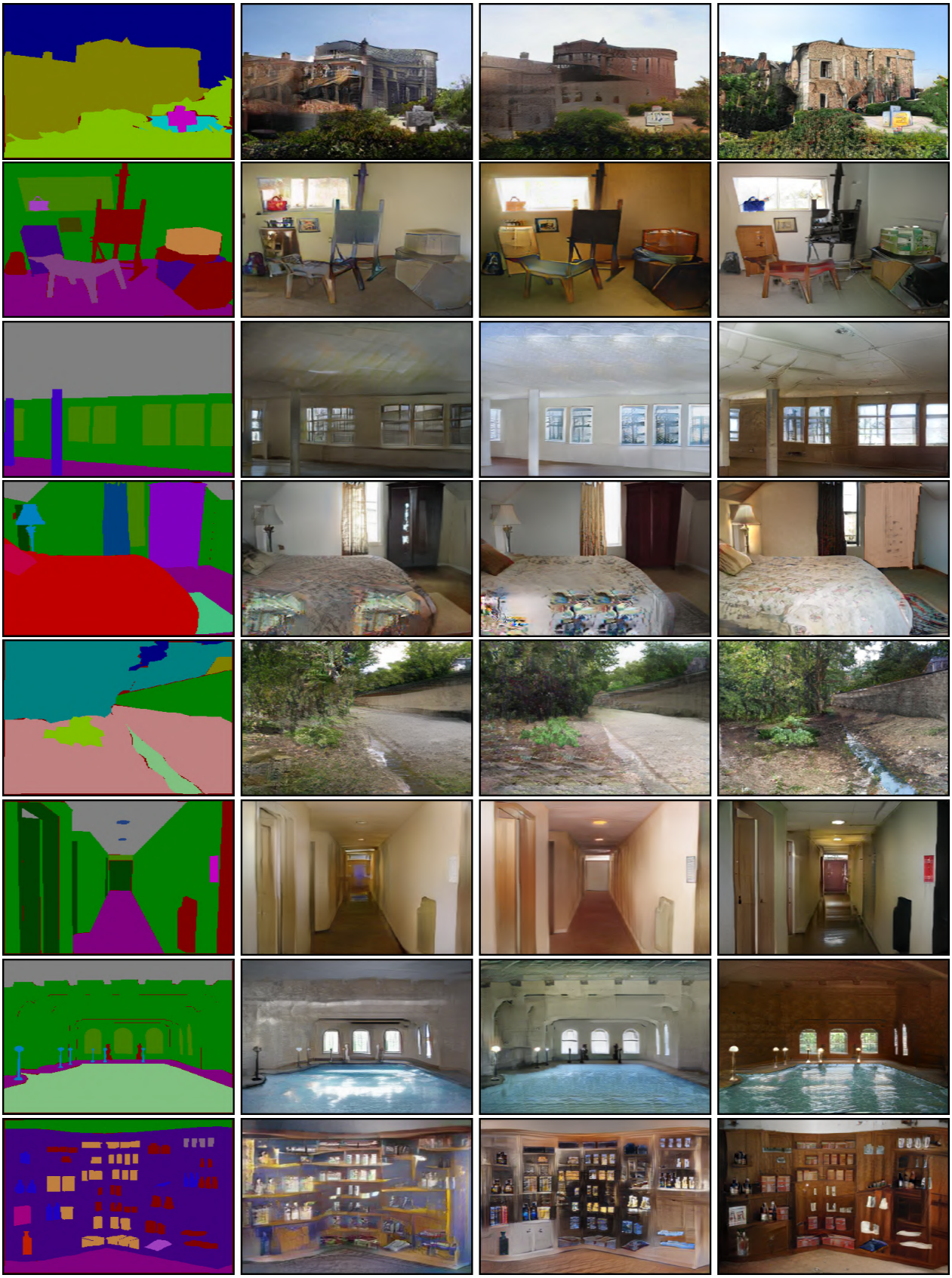
(1) The input (2) SPADE (3) CC-FPSE (4) Ours.

Figure 5. Visual comparisons on Cityscapes.



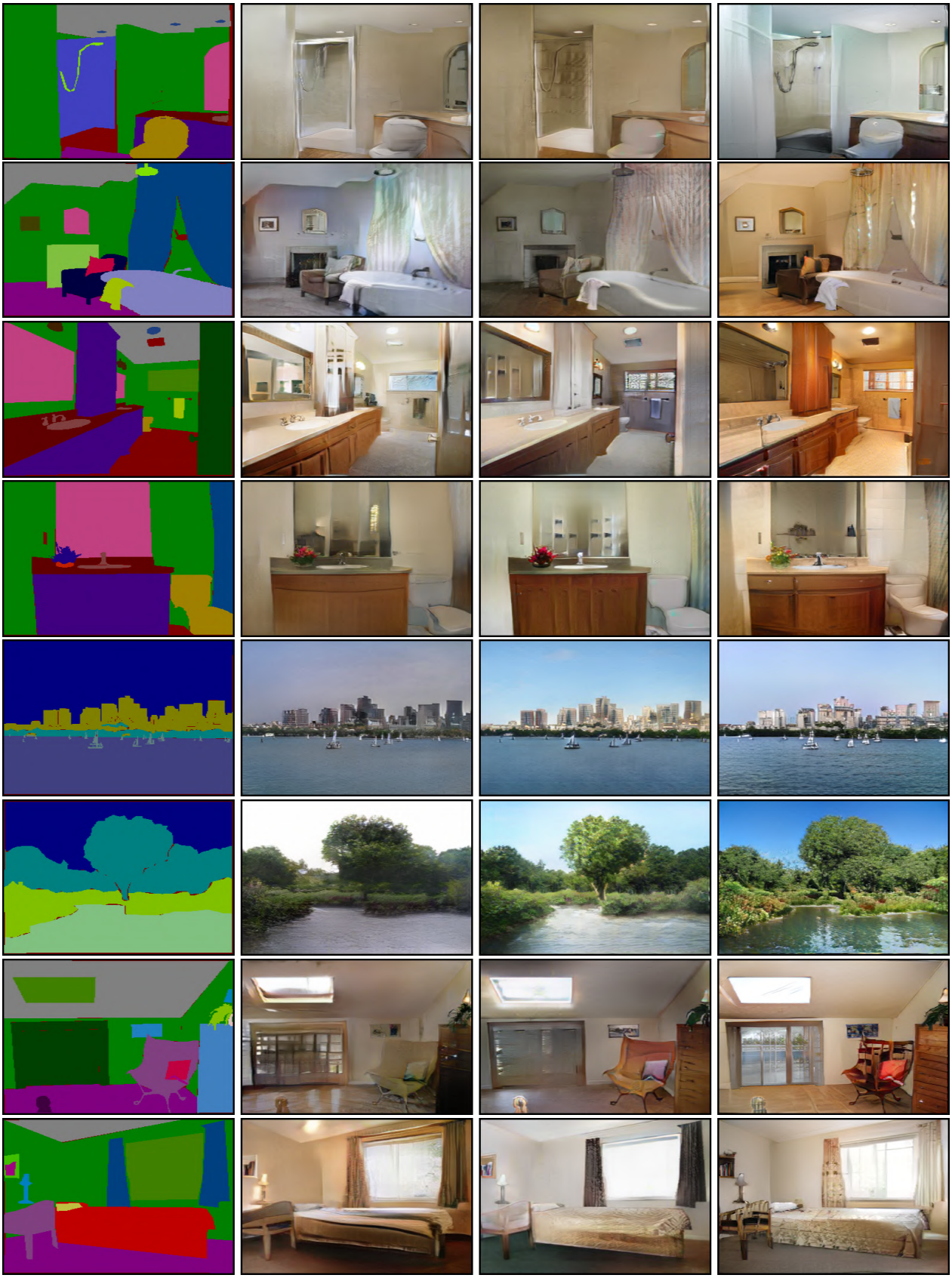
(1) The input (2) SPADE (3) CC-FPSE (4) Ours.

Figure 6. Visual comparisons on ADE20K.



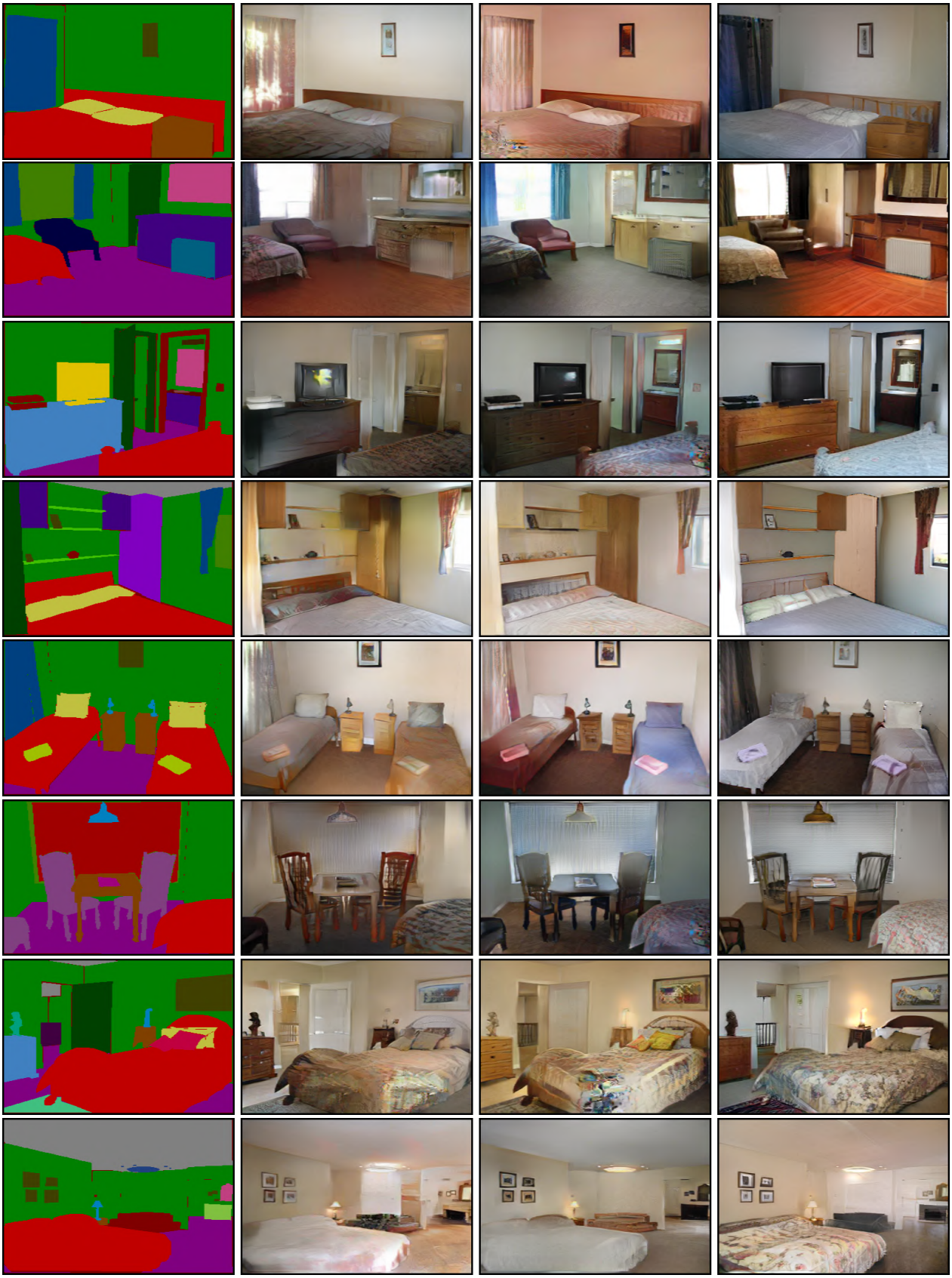
(1) The input (2) SPADE (3) CC-FPSE (4) Ours.

Figure 7. Visual comparisons on ADE20K.



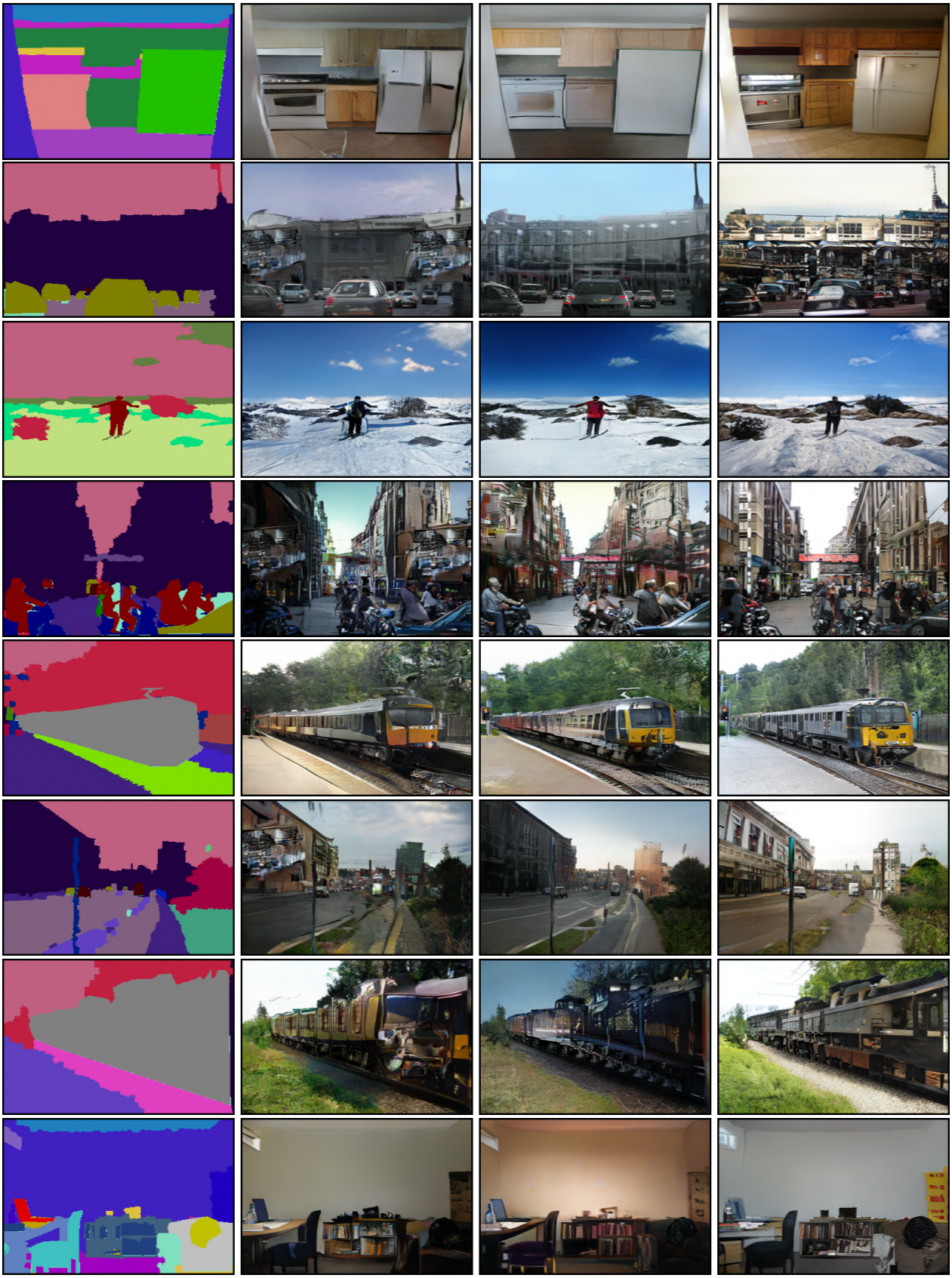
(1) The input (2) SPADE (3) CC-FPSE (4) Ours.

Figure 8. Visual comparisons on ADE20K.



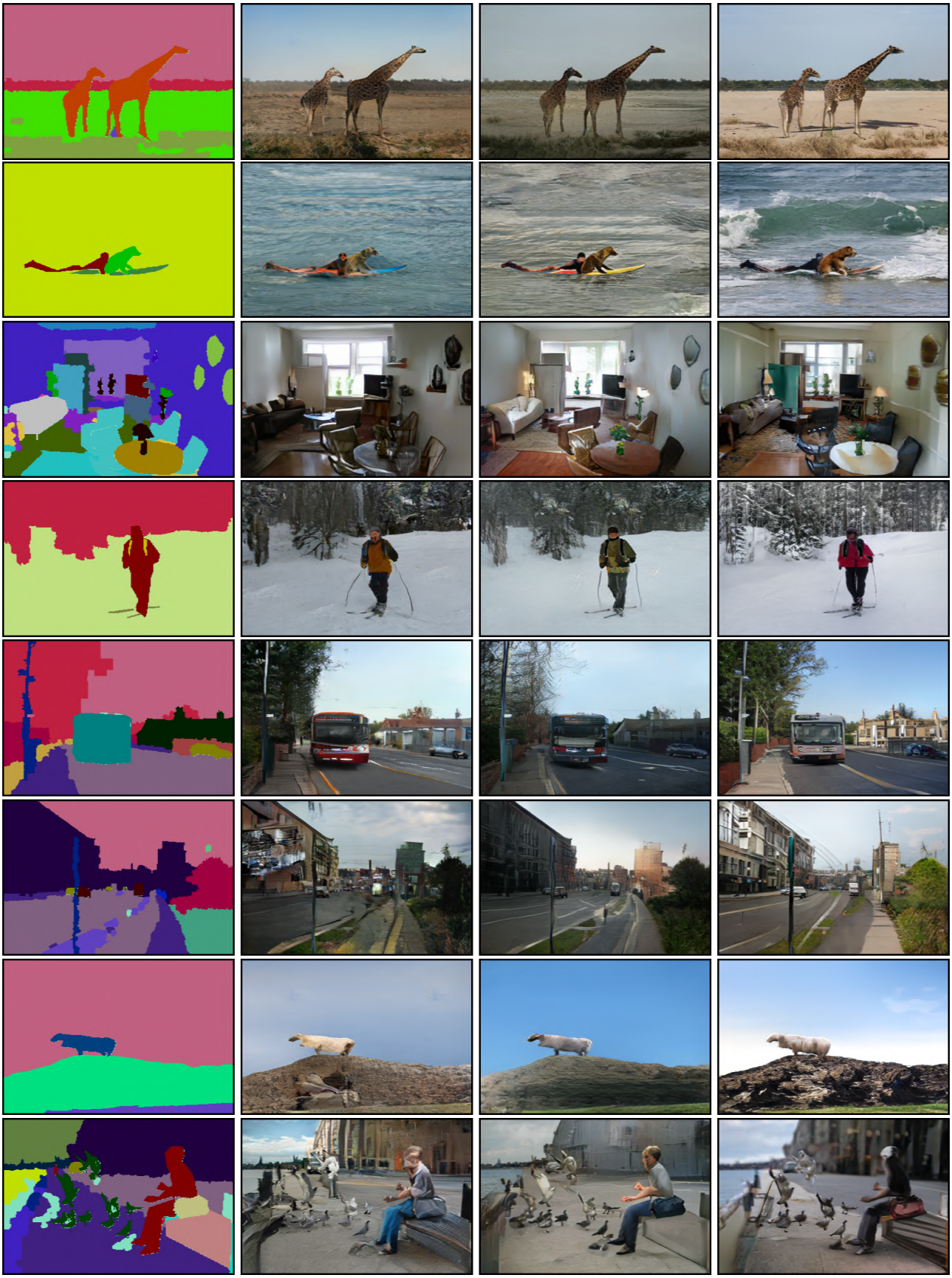
(1) The input (2) SPADE (3) CC-FPSE (4) Ours.

Figure 9. Visual comparisons on ADE20K.



(1) The input (2) SPADE (3) CC-FPSE (4) Ours.

Figure 10. Visual comparisons on COCO-stuff.

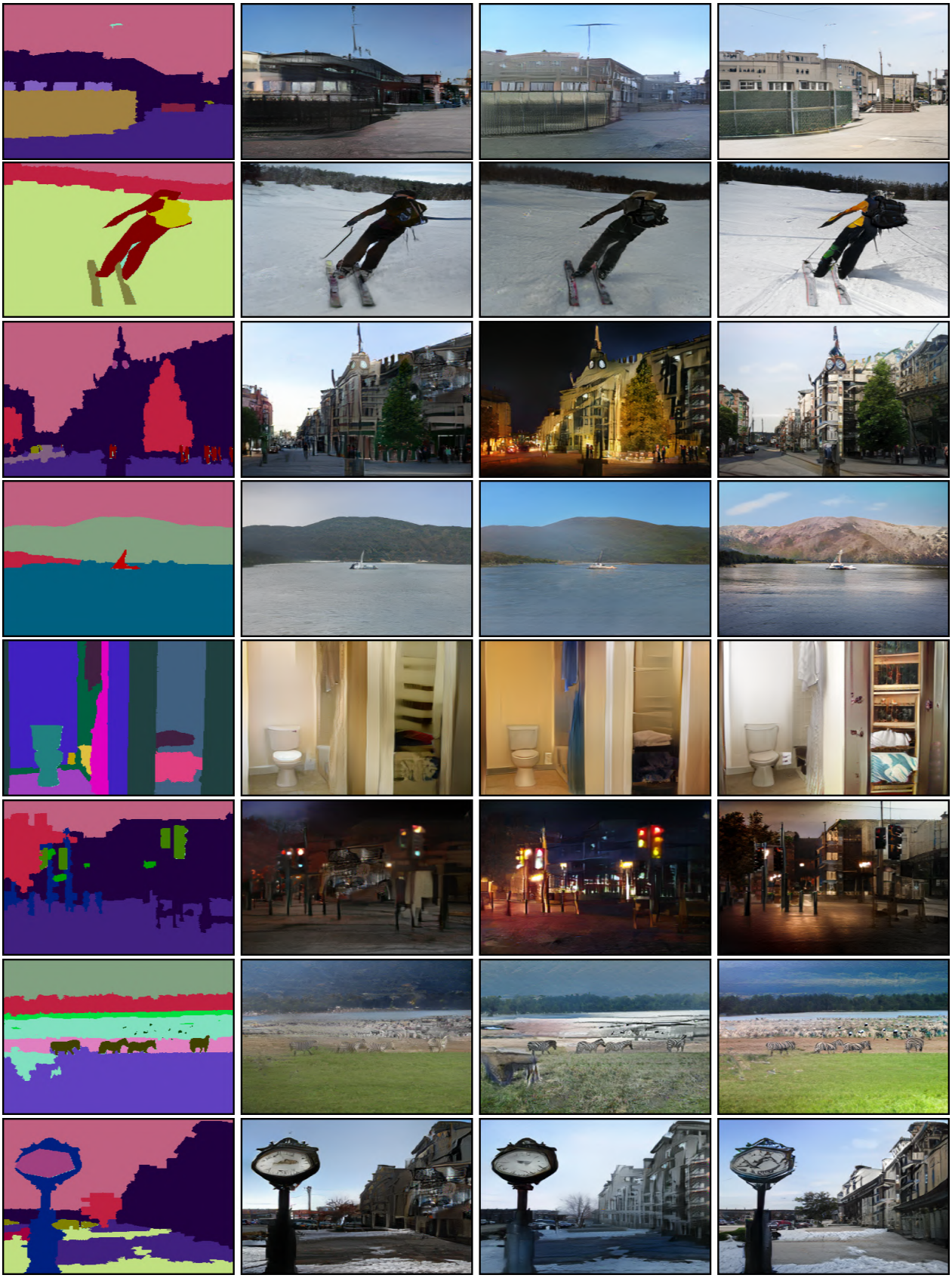


(1) The input (2) SPADE (3) CC-FPSE (4) Ours.

Figure 11. Visual comparisons on COCO-stuff.

(1) The input (2) SPADE (3) CC-FPSE (4) Ours.

Figure 12. Visual comparisons on COCO-stuff.



(1) The input (2) SPADE (3) CC-FPSE (4) Ours.

Figure 13. Visual comparisons on COCO-stuff.

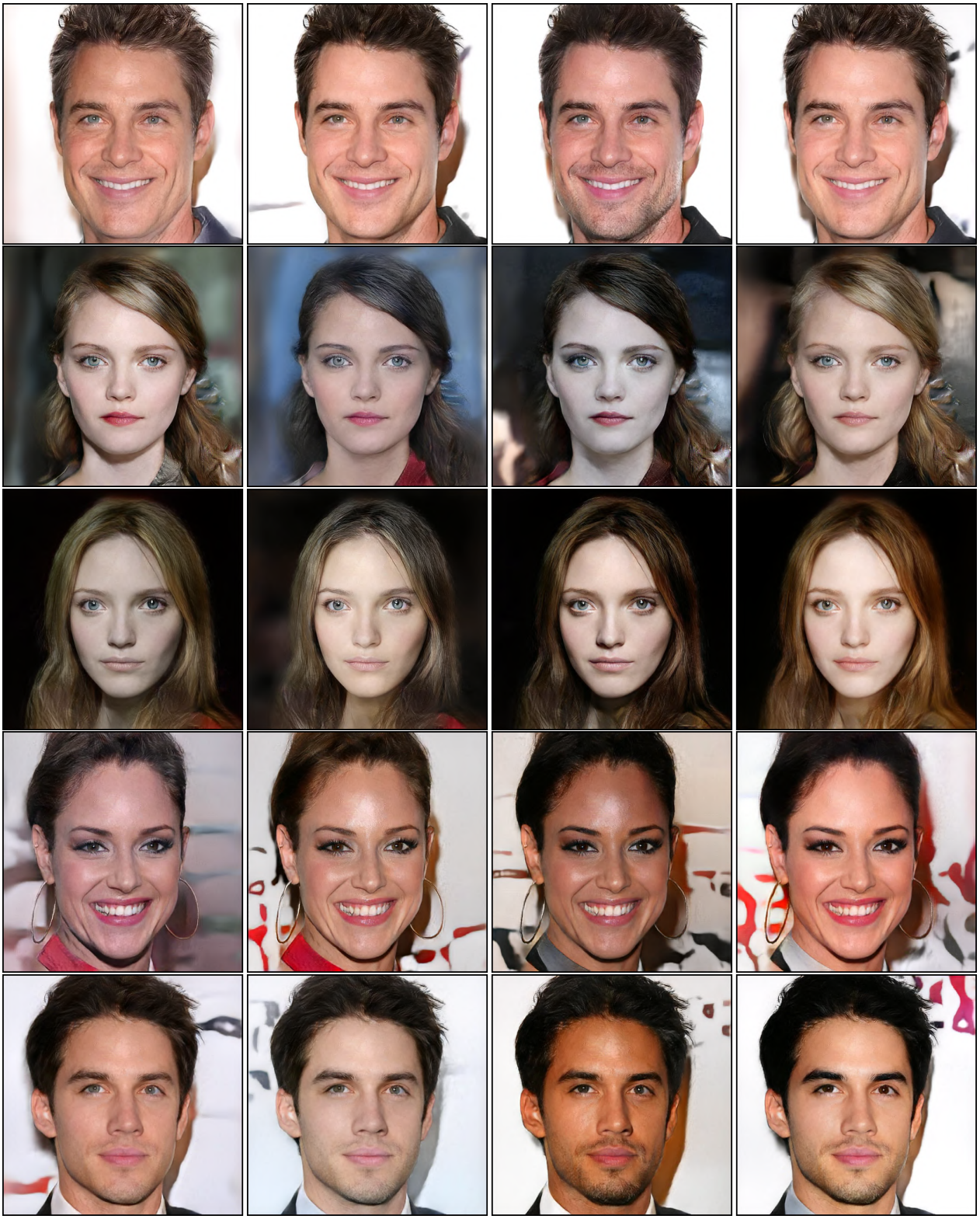


Figure 14. Multi-modal predictions of our method on CelebAMask-HQ.

Figure 15. Multi-modal predictions of our method on CelebAMask-HQ.

Figure 16. Multi-modal interpolation of our method on CelebAMask-HQ.

Figure 17. Multi-modal interpolation of our method on CelebAMask-HQ.

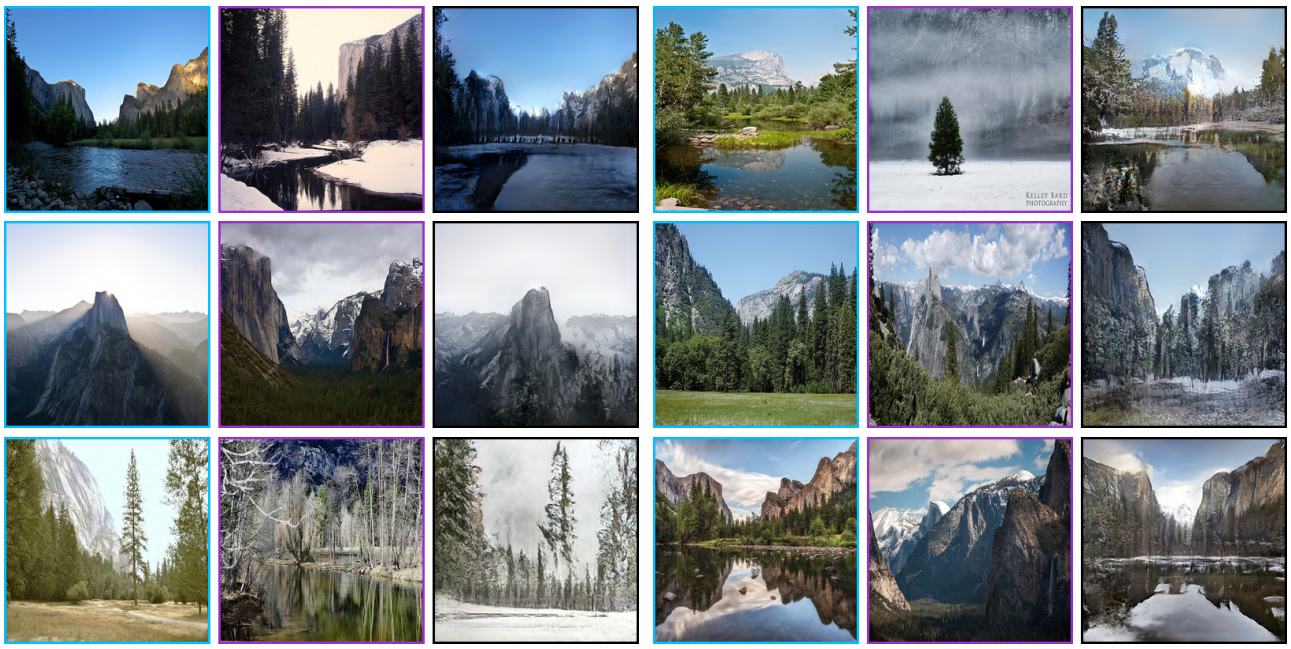


Figure 18. Unpaired image-to-image translation results from our model on summer→winter. The source images, their reference ones (target images), and their corresponding translated results are marked by blue, purple, and black rectangles, respectively.

(1) The input (2) SPADE (3) CC-FPSE (4) Ours.

Figure 19. Failure cases. The top row: failing to add dimension and depth in facade. The bottom row: introducing undesired objects in the given semantic regions.