Supplementary Materials for "*Masked and Adaptive Transformer for Exemplar Based Image Translation*" (Paper ID: 8685)

A Implementation Details

A.1 Details of Network Architecture.

As shown in Fig. 1, given an input image x_A and an exemplar image y_B , we first align x_A and y_B to an intermediate feature space by encoders $\mathcal{E}_A$ and $\mathcal{E}_B$, respectively. Afterwards, we use a *Masked and Adaptive Transformer* (MAT) for correspondence learning and feature augmentation. The output of MAT is fed into the decoder for producing an image. Table 1 shows details of our generator. Specially, it is composed of several ResBlocks and AdaIN-ResBlocks.

In the decoder $\mathcal{D}_B$, we first use two down-sampling layers for increasing the depth of the whole network, and then use several up-sampling layers to gradually decode an image. Following U-Net [2], the features over the encoding and down-sampling layers are skip-connected to the corresponding up-sampling layers in $\mathcal{D}_B$. In addition, we map the feature of the exemplar y_B, i.e. the output of $\mathcal{E}_B$, to a global style vector $\mathbf{z}$. Afterwards, $\mathbf{z}$ is fed to the ResAdaIN layers in $\mathcal{D}_B$ for global style control, in the manner of AdaIN [1].

AdaIN-ResBlock. The structure of AdaIN-ResBlock is as shown in Fig. 3a. Here, the global style vector $\mathbf{z}$ is mapped to modulating parameters (γ and β in Fig. 3a) by a three-layer MLP. In the MLP, we use leakyReLU as the activation function (Fig. 3b). AdaIN-ResBlock consists of a series of padding, convolution, and AdaIN [1] layers. Given a neuron activation $x_{c,h,w}$, the modulated output is formulated as:

$$x'_{c,h,w} = \frac{x_{c,h,w} - \mu_{h,w}}{\sigma_{h,w}}\gamma_c + \beta_c, \tag{1}$$

where h, w, and c stand for the height, width, and the number of channels, respectively. $\mu_{h,w}$ and $\sigma_{h,w}$ are the mean and standard deviation across each channel. The modulation parameter γ_c and β_c are the learned modulation parameters for the c-th channel. Finally, we use a residual connection to supplement semantic information from the input.

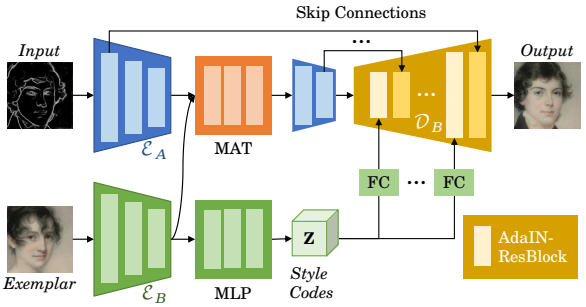

Figure 1: Overview of the proposed translation network.

Table 1: Details of network architecture. $k3s1$ indicates the convolutional layer with kernel size 3 and stride 1.

Sub-network	Layers	Output ($H \times W \times C$)
Encoders $\mathcal{E}_{A\_1}$ and $\mathcal{E}_B$	Conv2d	256×256×64
	Conv2d/k3s1 + Resblock/k3s1	256×256×64
	Conv2d/k4s2 + Resblock/k3s1	128×128×128
	Conv2d/k4s2 + Resblock/k3s1	64×64×256
	Conv2d/k4s2 + Resblock/k3s1	32×32×512
MAT	MAT blocks × 3	32×32×512
		32×32×512
		32×32×512
MLP	Linear + LeakyReLU	64
	Linear + LeakyReLU	64
	Linear	64
Decoder $\mathcal{D}_B$	Conv2d/k4s2 + Resblock/k3s1	16×16×512
	AdaIN-ResBlock + Resblock/k3s1	16×16×512
	AdaIN-ResBlock + Resblock/k3s1	32×32×512
	AdaIN-ResBlock + Resblock/k3s1	64×64×256
	AdaIN-ResBlock + Resblock/k3s1	128×128×128
	AdaIN-ResBlock + Resblock/k3s1	256×256×64
	Conv2d/k3s1 + Tanh	256×256×3

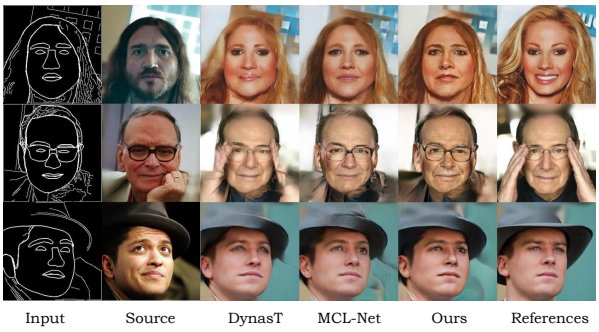

Figure 2: Shown are the failure columns on the CelebA[3] dataset. From left to right, each column shows Edge, GT, DyanST[4], MCL-Net[5], our proposed method and example plots, respectively.

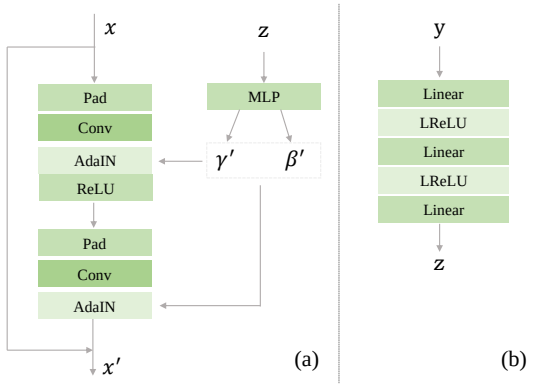

Figure 3: Detail structure demonstration. (a) ResAdaIN detail section, (b) Style Mapping Projector detail section.

A.2 Training Details

On the CelebA-HQ dataset, we exactly follow the training and test settings used in CoCosNet. For the other small datasets, we use VGG19 to extract high-dimensional features and calculate the cosine similarity between photos and stylistic images, to select the top 10 similarity exemplars for training. In the training stage of all the experiments, we usedata augmentation strategies (e.g., random cropping, flipping, etc.) to increase the network's generalization ability. On large datasets (i.e. CelebA-HQ and AAHQ) we train our model for 60 epochs, and the learning rate decays to 0 in the last 30 epochs. On the other small datasets, we train 200 epochs and the learning rate decays to 0 in the last 100 epochs. During all training scenarios, the contrastive style loss Lstyle only works in the second half of the training process. Detailed information is provided in Table 1.

A.3 Limitations.

Although our model can generate impressive results in most cases, it still has some limitations. We show a few cases of failure in Fig. A.1. First, the model has difficulty producing quality details when encountering complex scenes, such as the presence of occlusions, hats, or side faces. In these cases, the model fails to establish a precise cross-domain correspondence, resulting in unappealing appearances in generated images. Inspiringly, our model considerably outperforms previous methods in general. In the near future, we will explore solving this issue via semi-supervised learning or domain transfer technologies.

A.4 Additional Results.

Finally, we provide more results on all datasets in Figs.A.4. We also report the FID and SWD values w.r.t. the additional applications, including the generation of landscape photos, Chinese ink paintings of landscapes, and Chinese ink paintings of portraits (i.e. claboratestyle painting) in Table 2. We made it public to Google Cloud Drive at https://drive.google.com/file/d/1sBH59nIzZaXik9R1EOzrFCgFNXdNivit/view?usp=sharing

Table 2: Quality Scores

Datasets	FID ↓	SWD ↓
Chinese Painting	30.61	13.40
Natural Scenes	15.05	28.57
Meticulous Painting	30.30	21.84

References

[1] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE international conference on computer vision*, pages 1501–1510, 2017.

[2] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[3] Cheng-Han Lee, Ziwei Liu, Lingyun Wu, and Ping Luo. Maskgan: Towards diverse and interactive facial image manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5549–5558, 2020.

[4] Songhua Liu, Jingwen Ye, Sucheng Ren, and Xinchao Wang. Dynast: Dynamic sparse transformer for exemplar-guided image generation. *arXiv preprint arXiv:2207.06124*, 2022.

[5] Fangneng Zhan, Yingchen Yu, Rongliang Wu, Jiahui Zhang, Shijian Lu, and Changgong Zhang. Marginal contrastive correspondence for guided image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10663–10672, 2022.

Figure 4: More Qualitative results on DeepFashion datasets.

Figure 5: More Qualitative results on DeepFashion datasets.

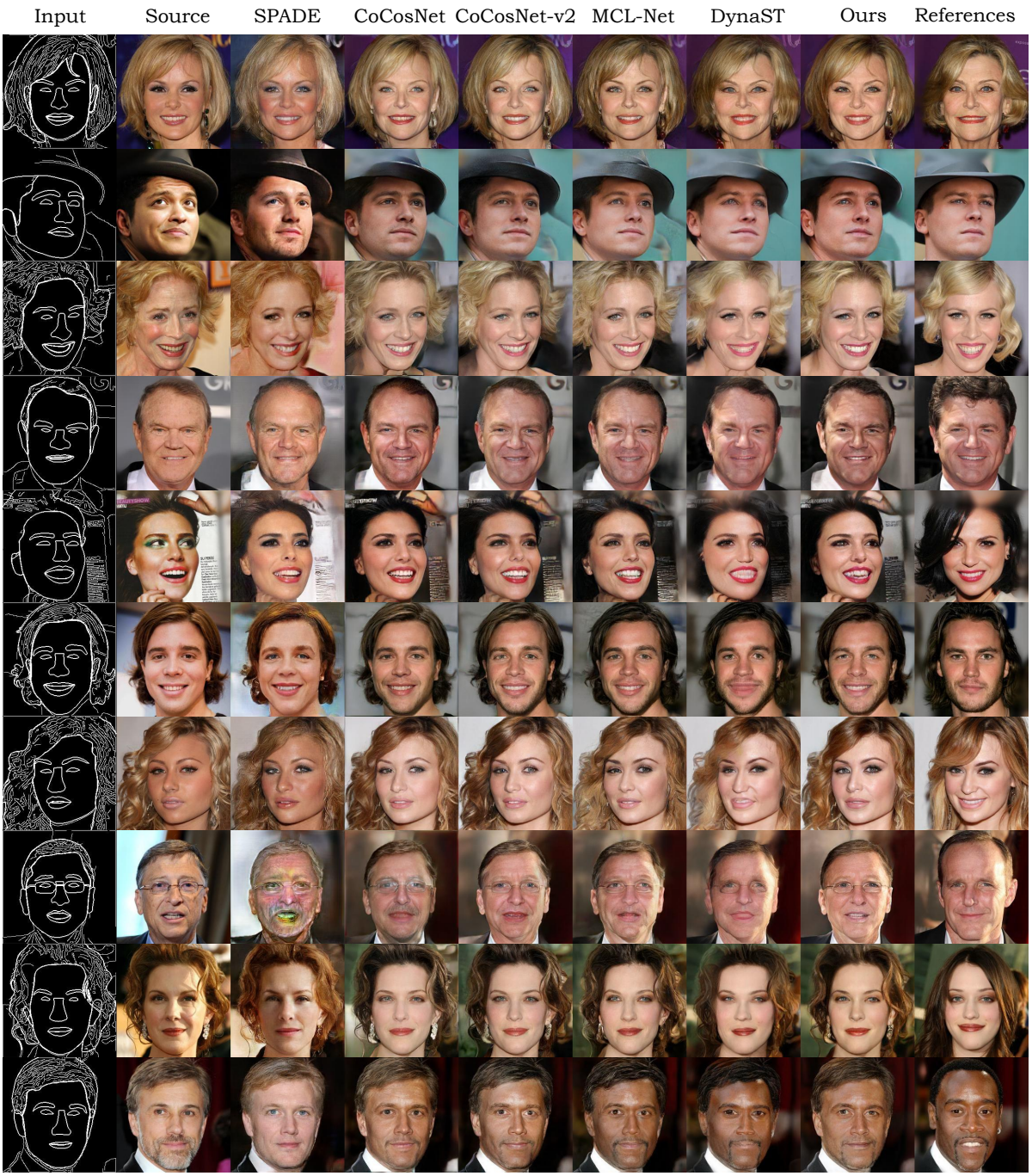

Figure 6: More Qualitative results on CelebA datasets.

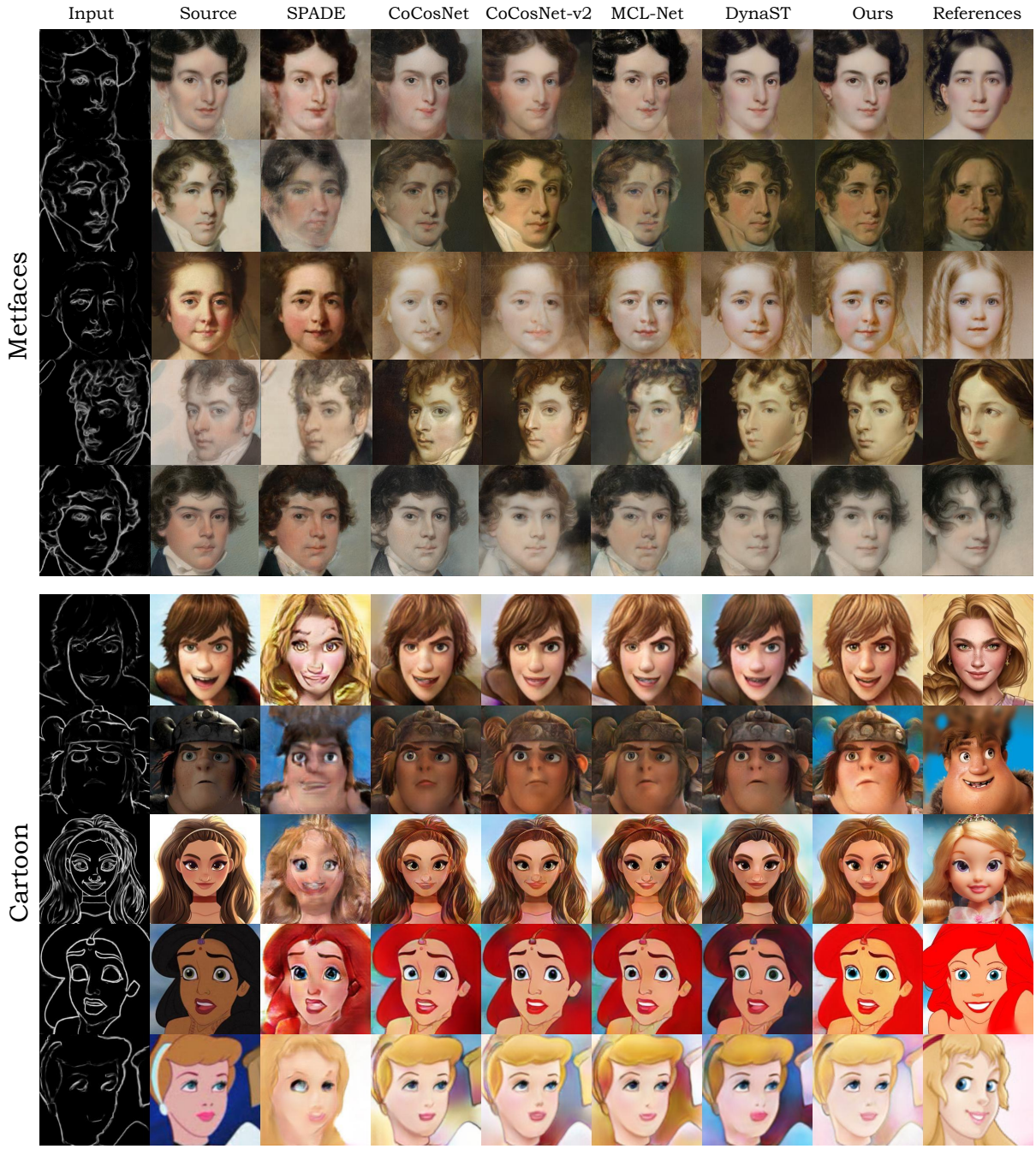

Figure 7: More Qualitative results on Metfaces and Cartoon datasets.

Figure 8: More Qualitative results on Ukiyo-e and AAHQ datasets.

Figure 9: Our results of edge to face synthesis (Metfaces dataset). First row: exemplars. Second row: our results.

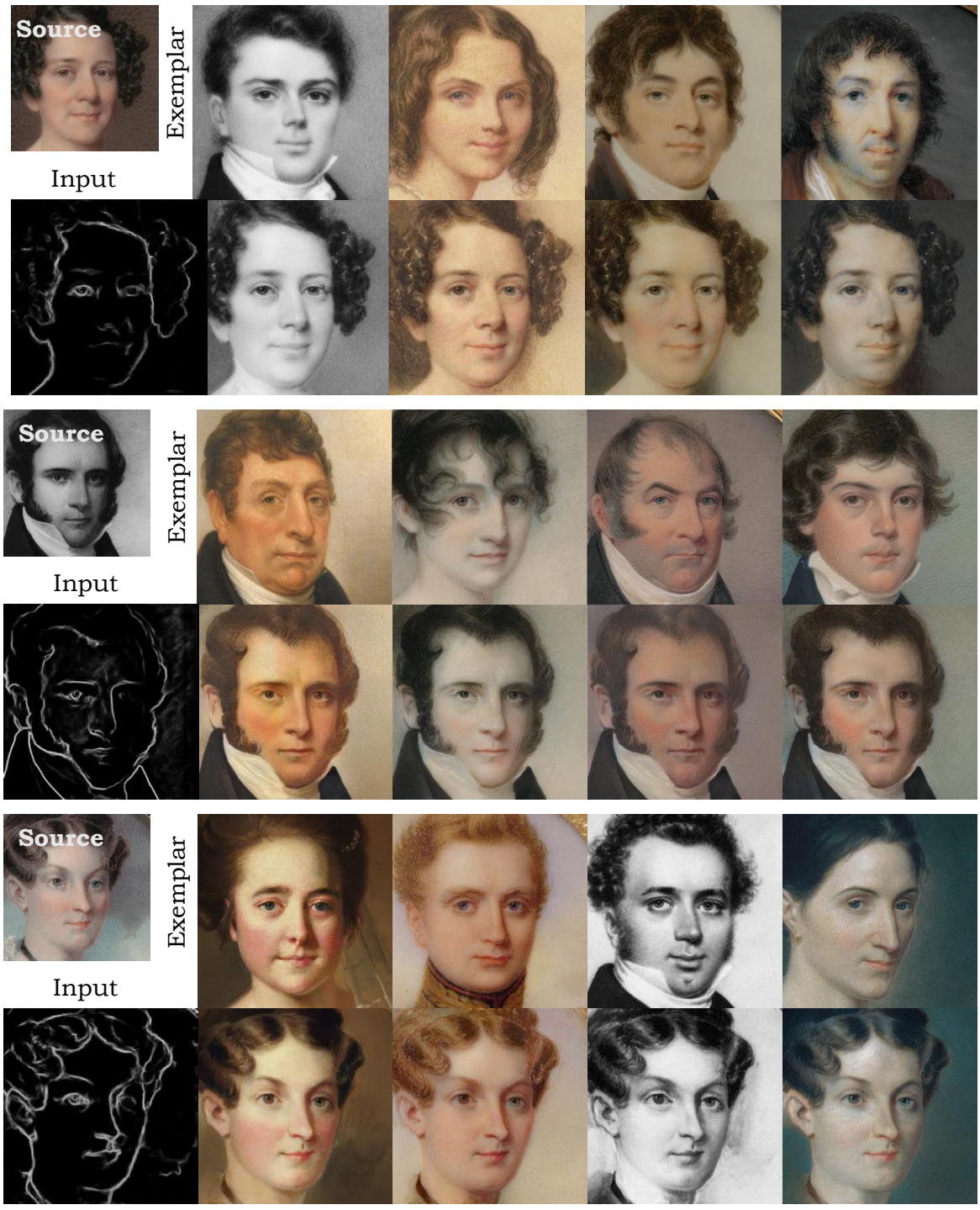

Figure 10: Our results of edge to face synthesis (Metfaces dataset). First row: exemplars. Second row: our results.

Figure 11: Our results of edge to face synthesis (Metfaces dataset). First row: exemplars. Second row: our results.

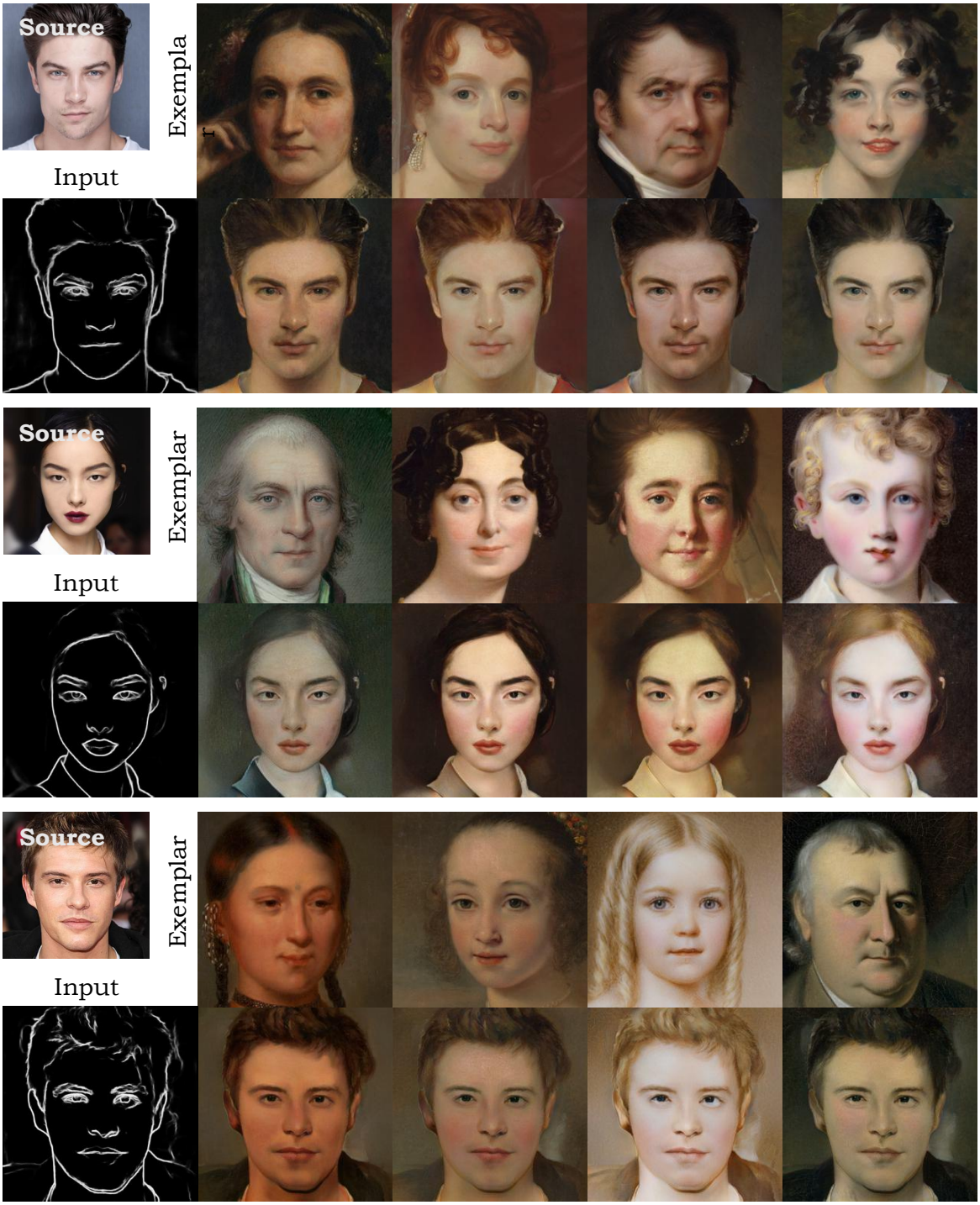

Figure 12: Style transfer portrait, given a face portrait, our methods can transform it into a custom portrait with an example style.

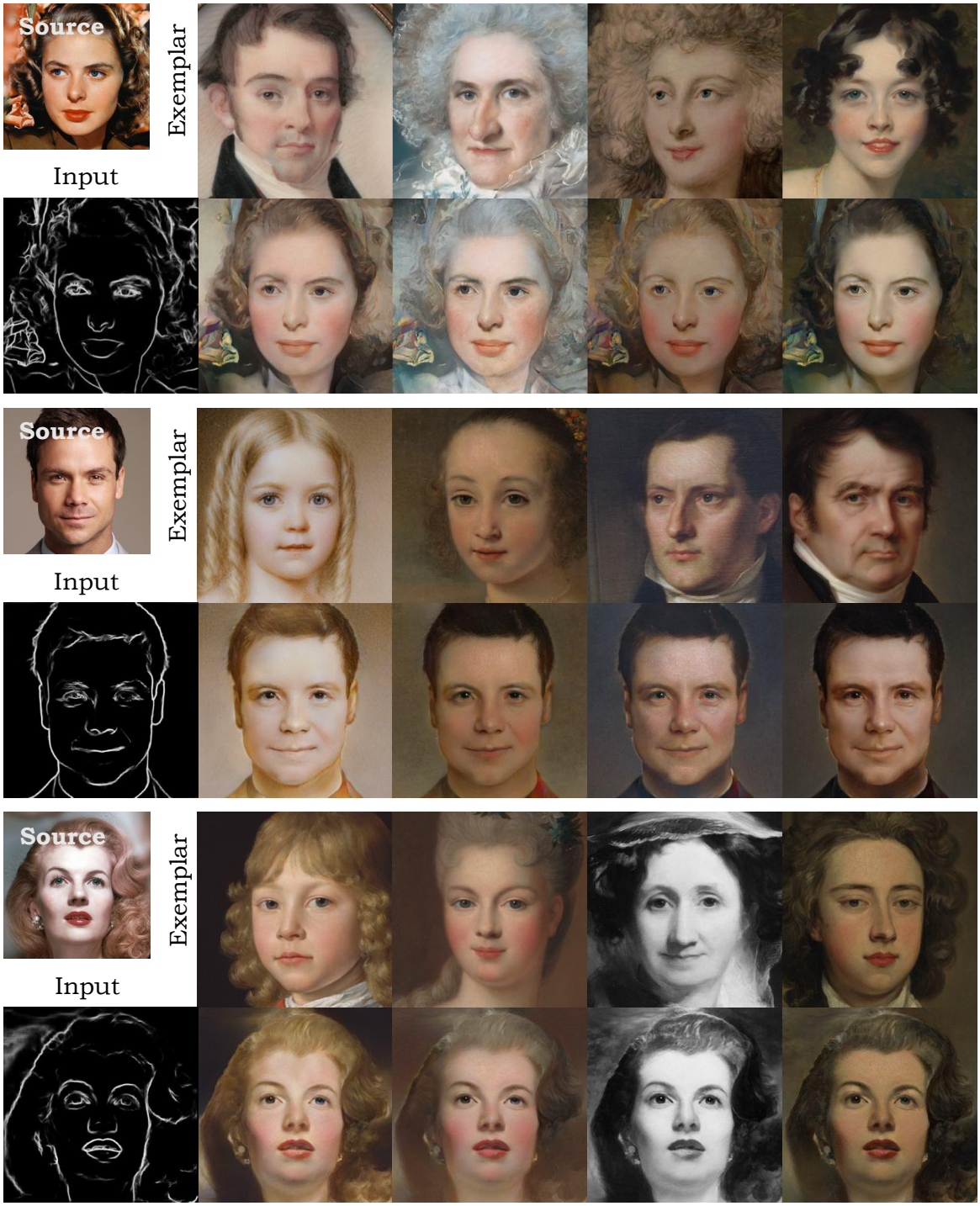

Figure 13: Style transfer portrait, given a face portrait, our methods can transform it into a custom portrait with an example style.

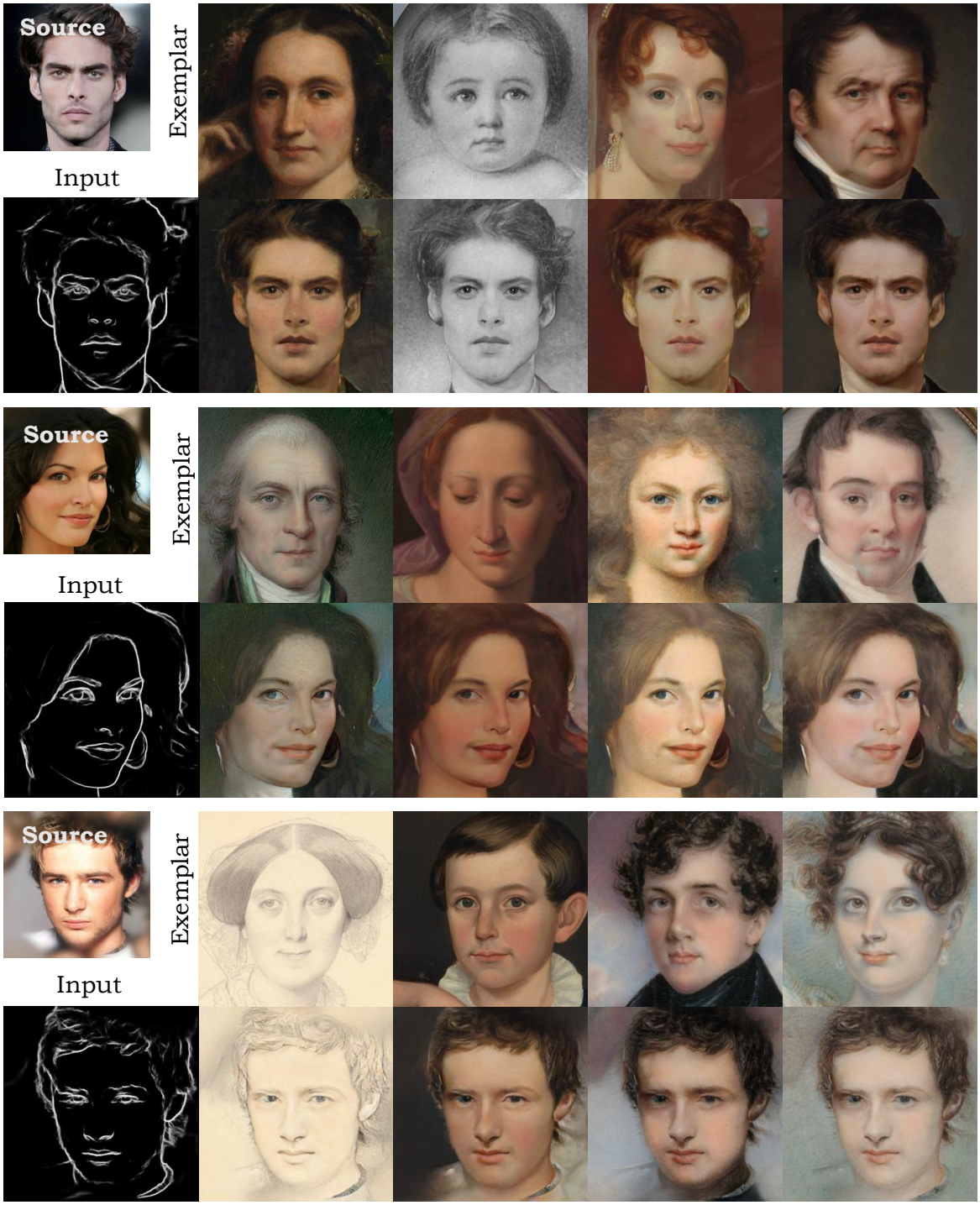

Figure 14: Style transfer portrait, given a face portrait, our methods can transform it into a custom portrait with an example style.

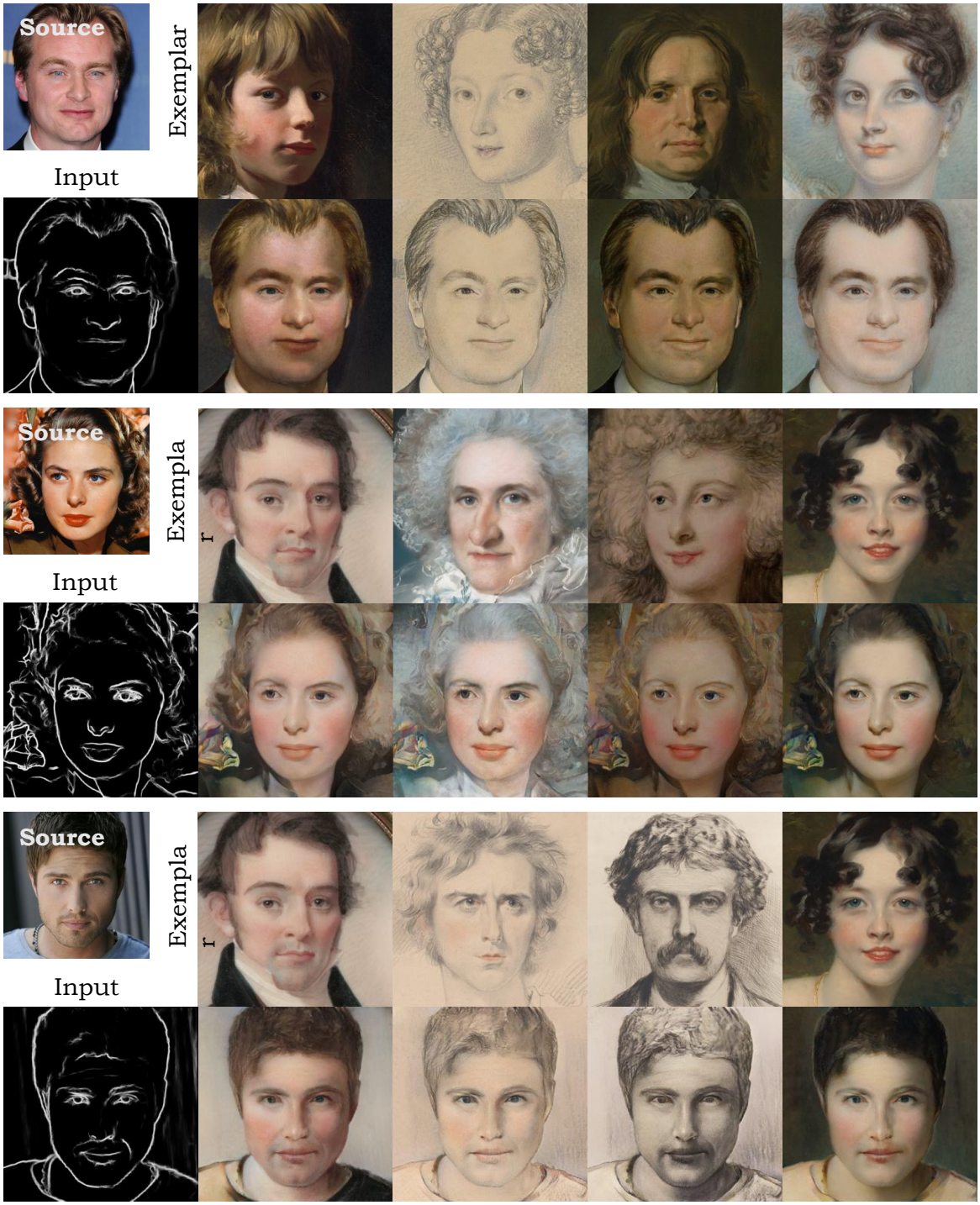

Figure 15: Style transfer portrait, given a face portrait, our methods can transform it into a custom portrait with an example style.

Figure 16: Our results of edge to face synthesis (CelebA-HQ dataset). First row: exemplars. Second row: our results.

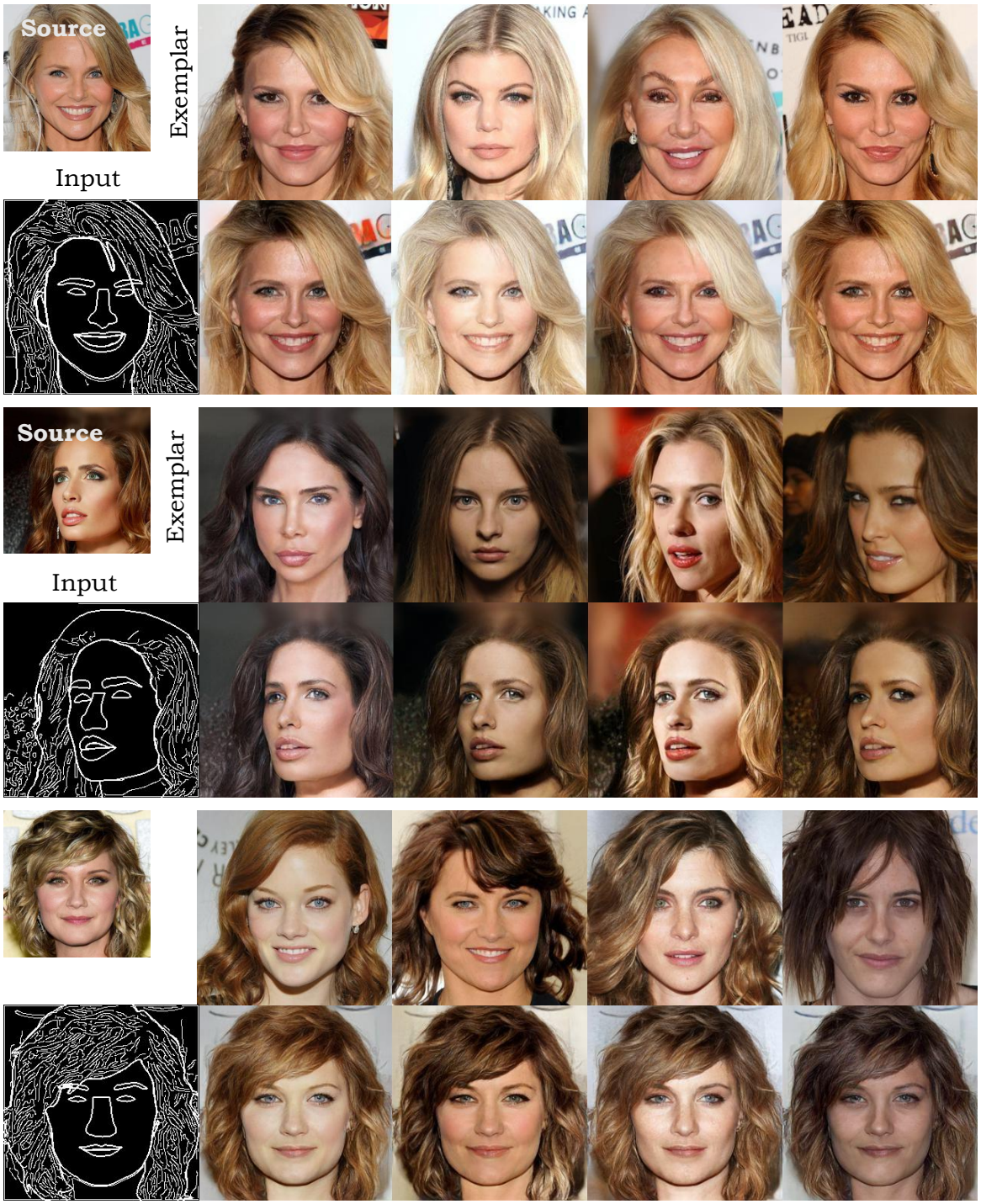

Figure 17: Our results of edge to face synthesis (CelebA-HQ dataset). First row: exemplars. Second row: our results.

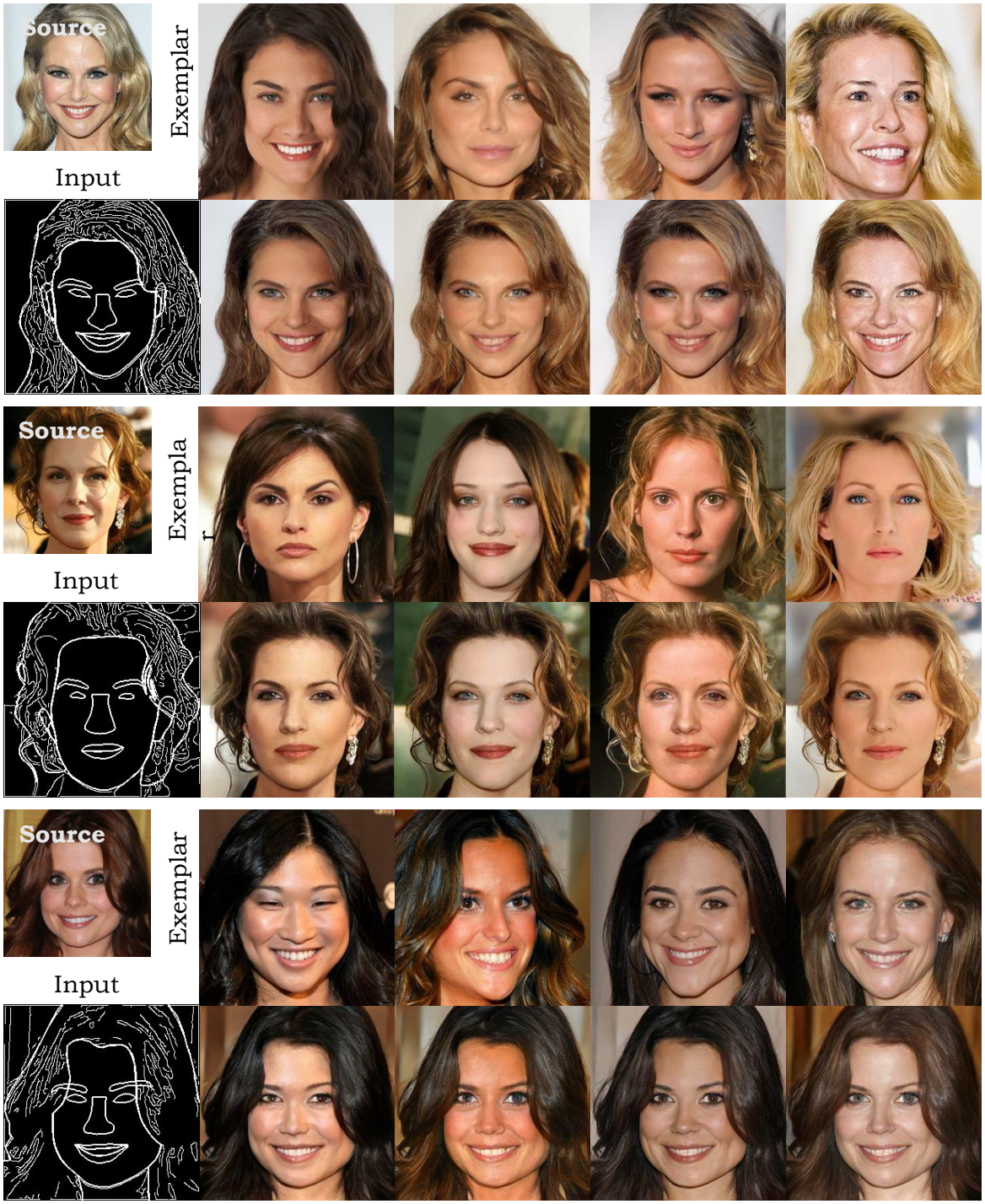

Figure 18: Our results of edge to face synthesis (CelebA-HQ dataset). First row: exemplars. Second row: our results.

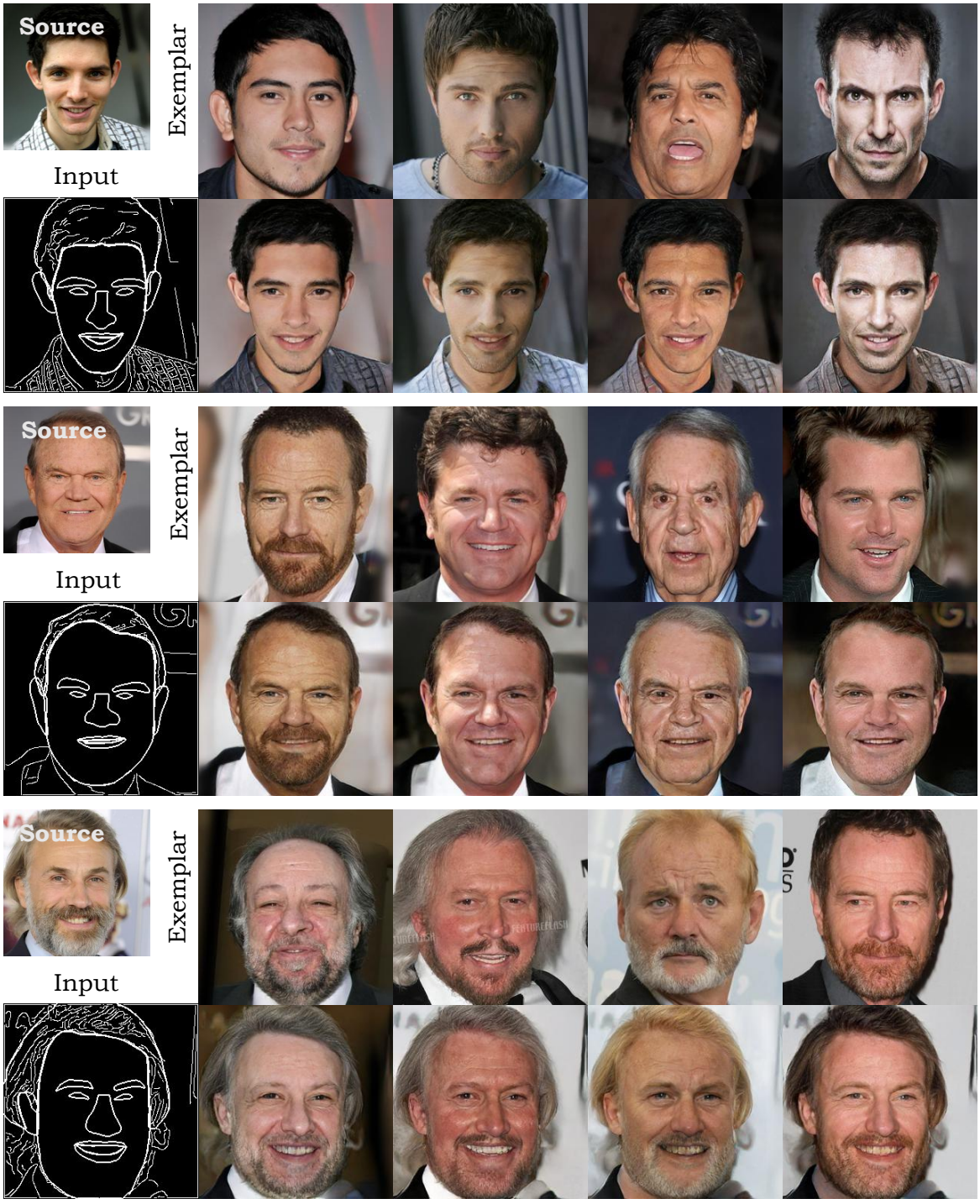

Figure 19: Our results of edge to face synthesis (CelebA-HQ dataset). First row: exemplars. Second row: our results.

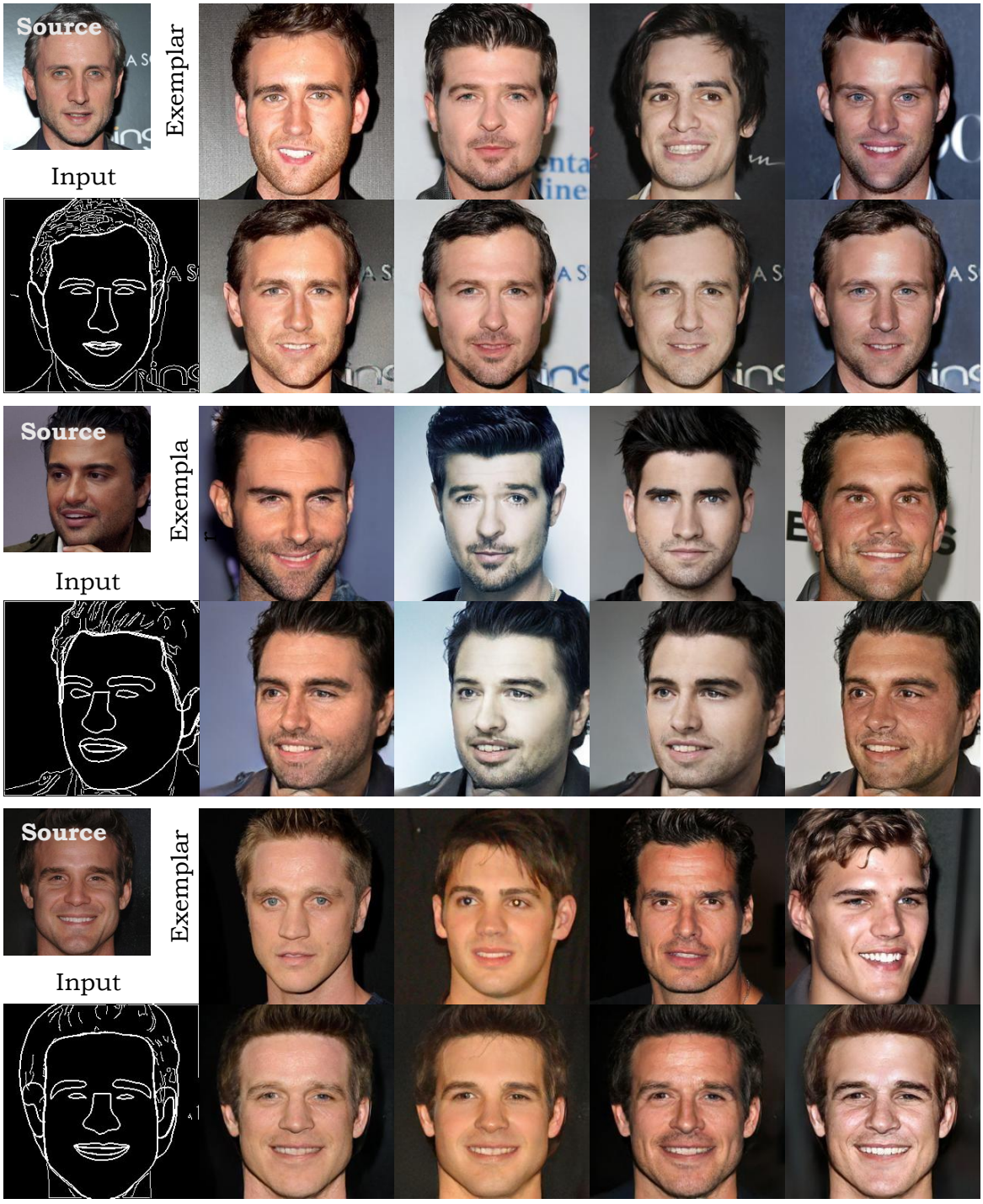

Figure 20: Our results of edge to face synthesis (CelebA-HQ dataset). First row: exemplars. Second row: our results.

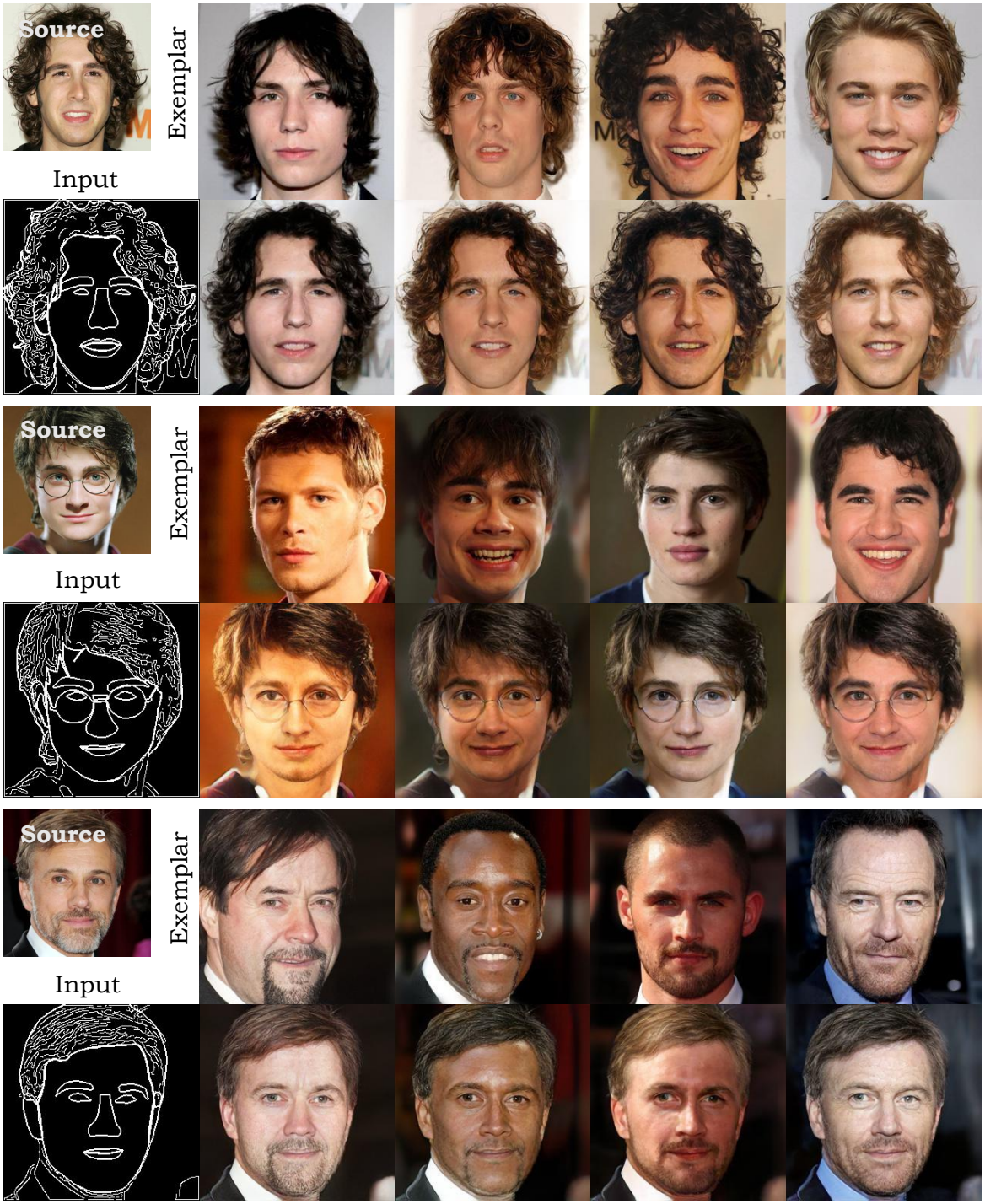

Figure 21: Our results of edge to face synthesis (CelebA-HQ dataset). First row: exemplars. Second row: our results.

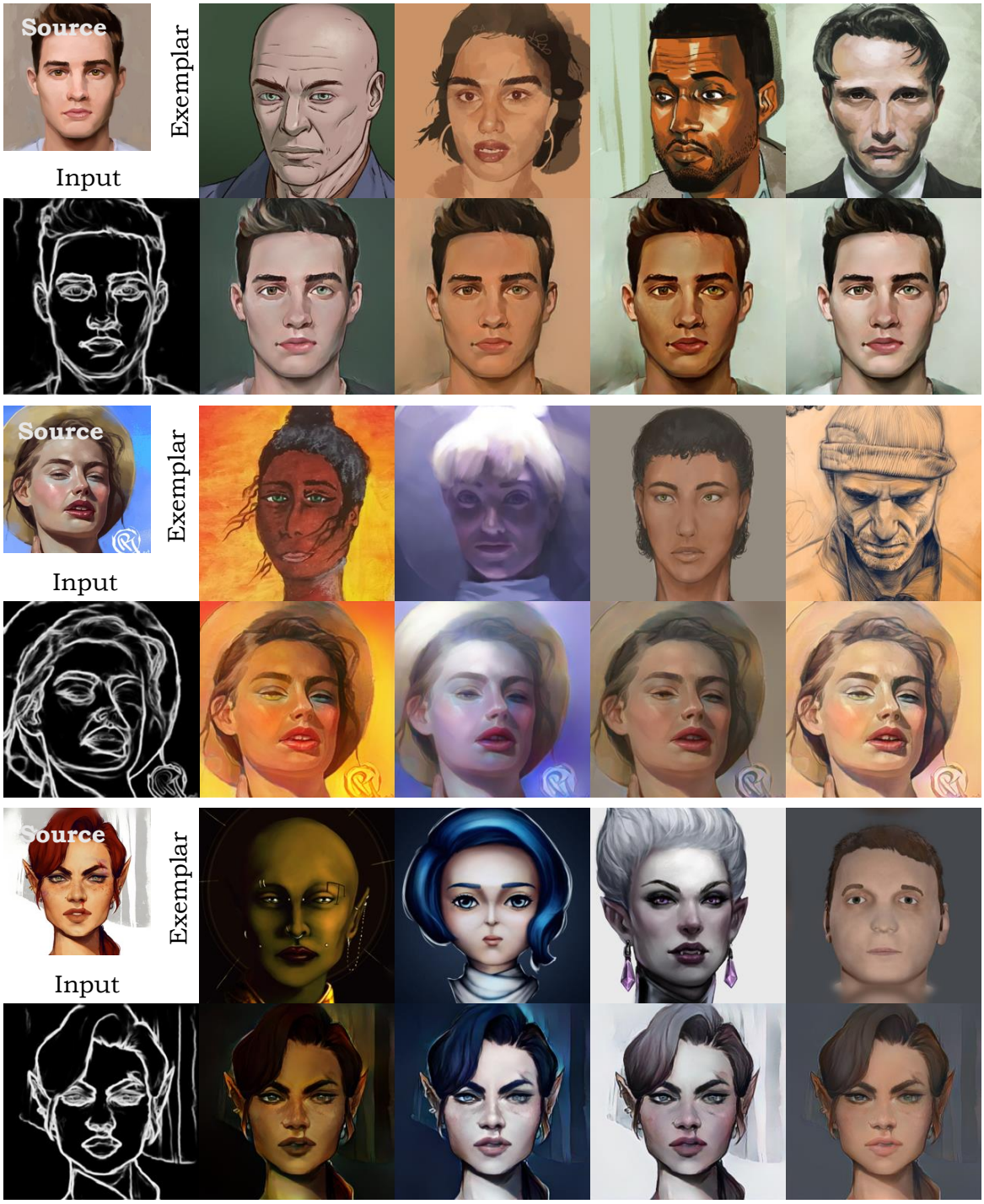

Figure 22: Our results of edge to face synthesis (AAHQ dataset). First row: exemplars. Second row: our results.

Figure 23: Our results of edge to face synthesis (AAHQ dataset). First row: exemplars. Second row: our results.

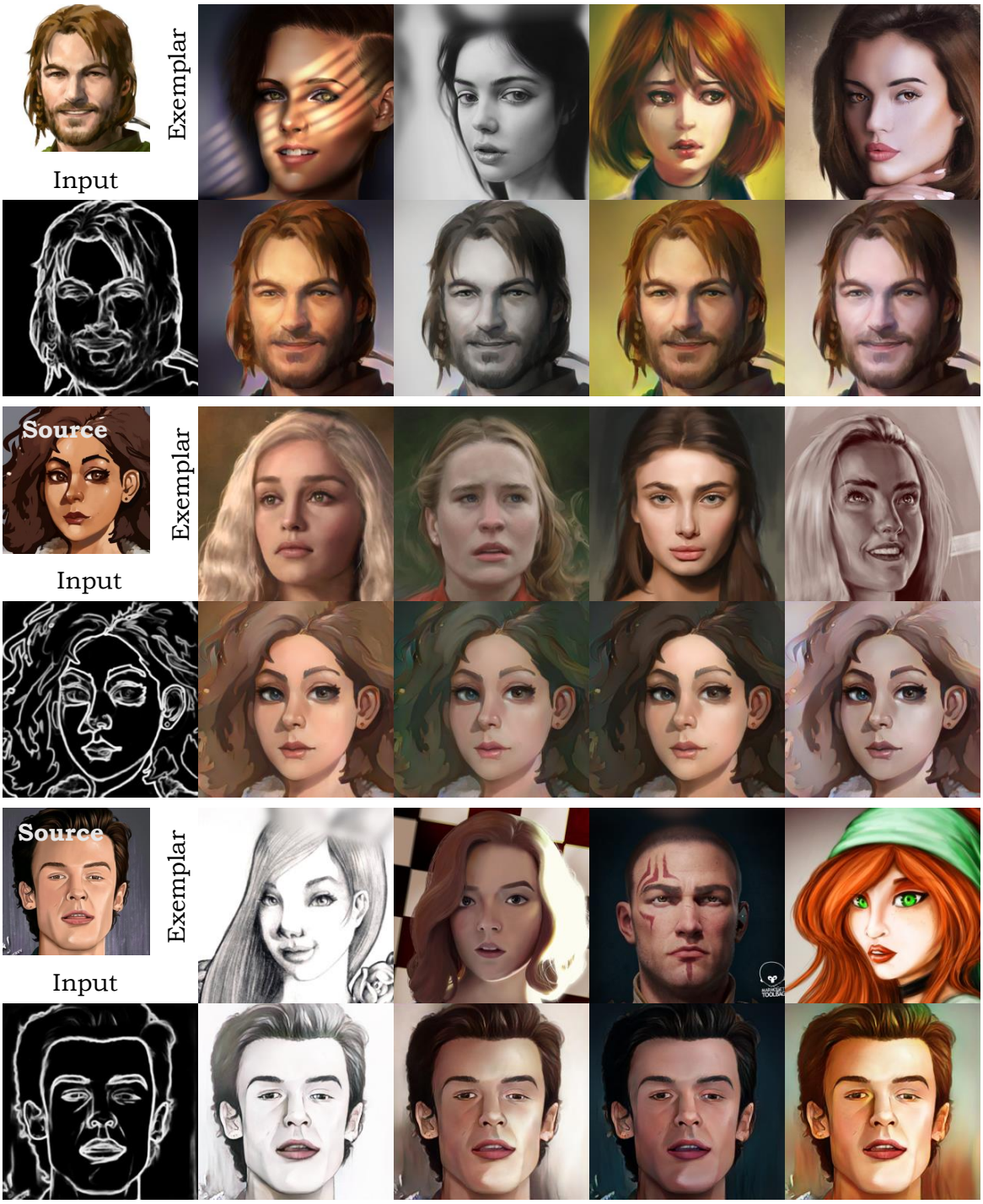

Figure 24: Our results of edge to face synthesis (AAHQ dataset). First row: exemplars. Second row: our results.

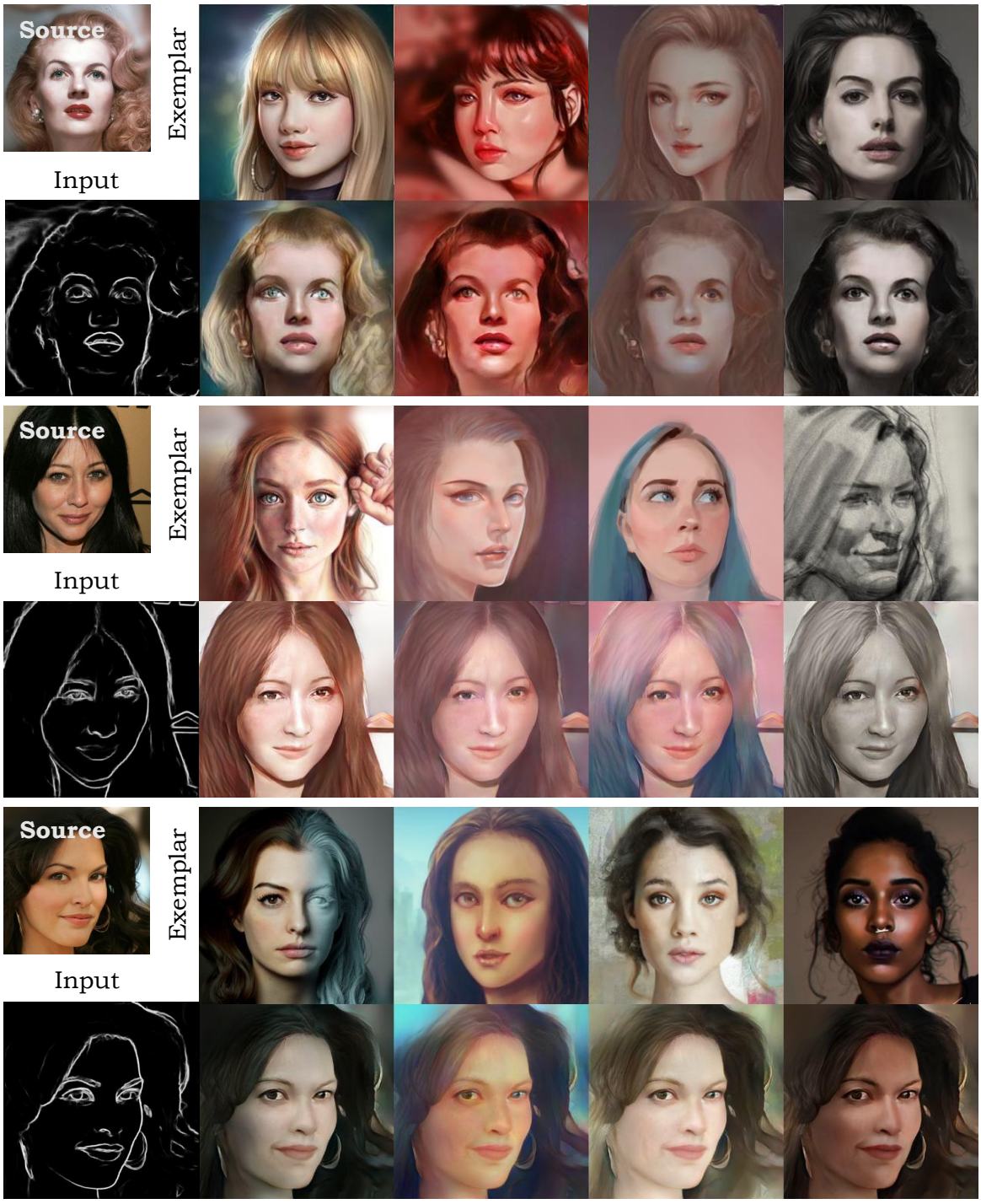

Figure 25: Style transfer portrait, given a face portrait, our methods can transform it into a custom portrait with an example style.

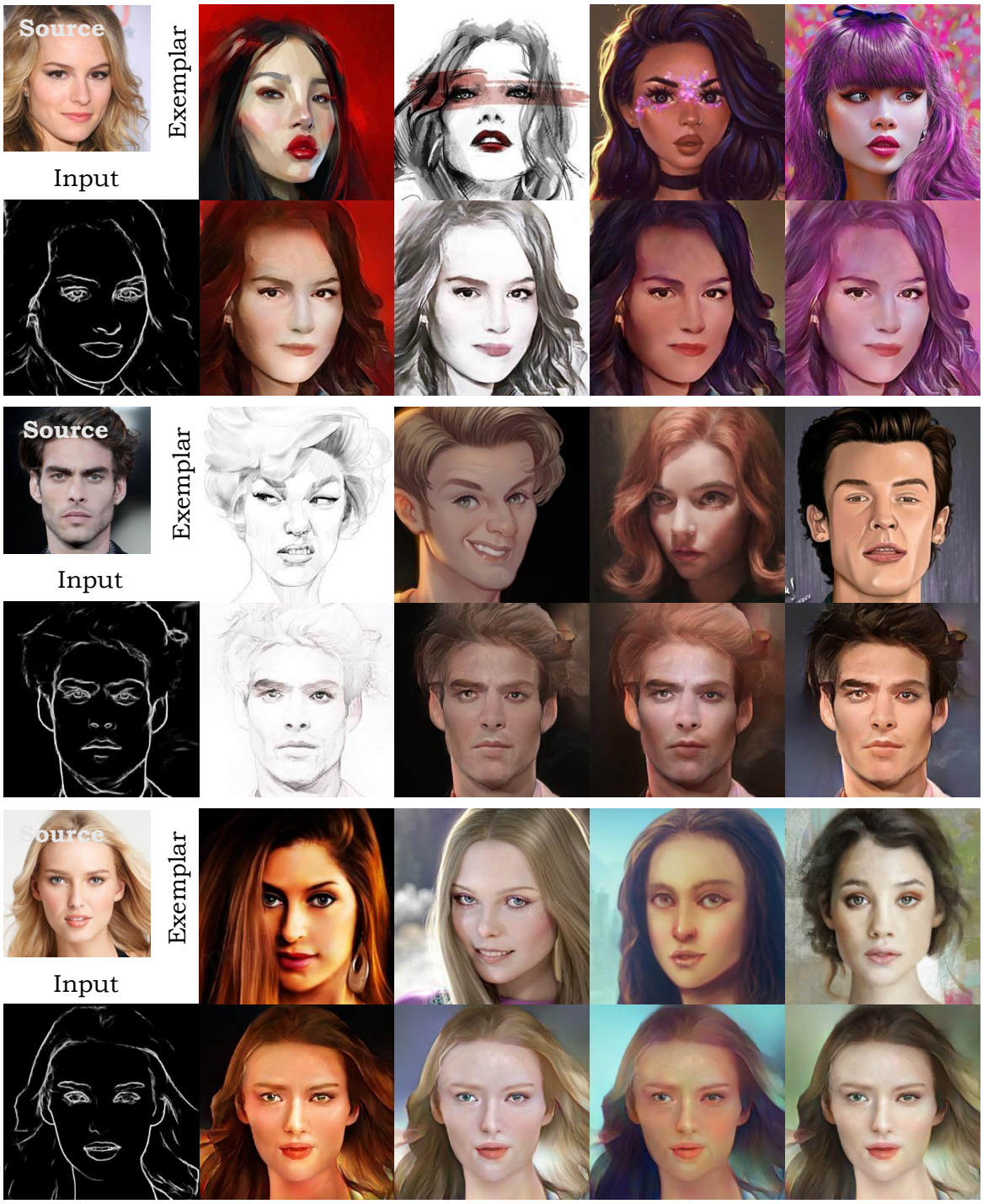

Figure 26: Style transfer portrait, given a face portrait, our methods can transform it into a custom portrait with an example style.

Figure 27: Our results of edge to face synthesis (Ukiyo-e Faces). First row: exemplars. Second row: our results.

Figure 28: Our results of edge to face synthesis (Ukiyo-e Faces). First row: exemplars. Second row: our results.

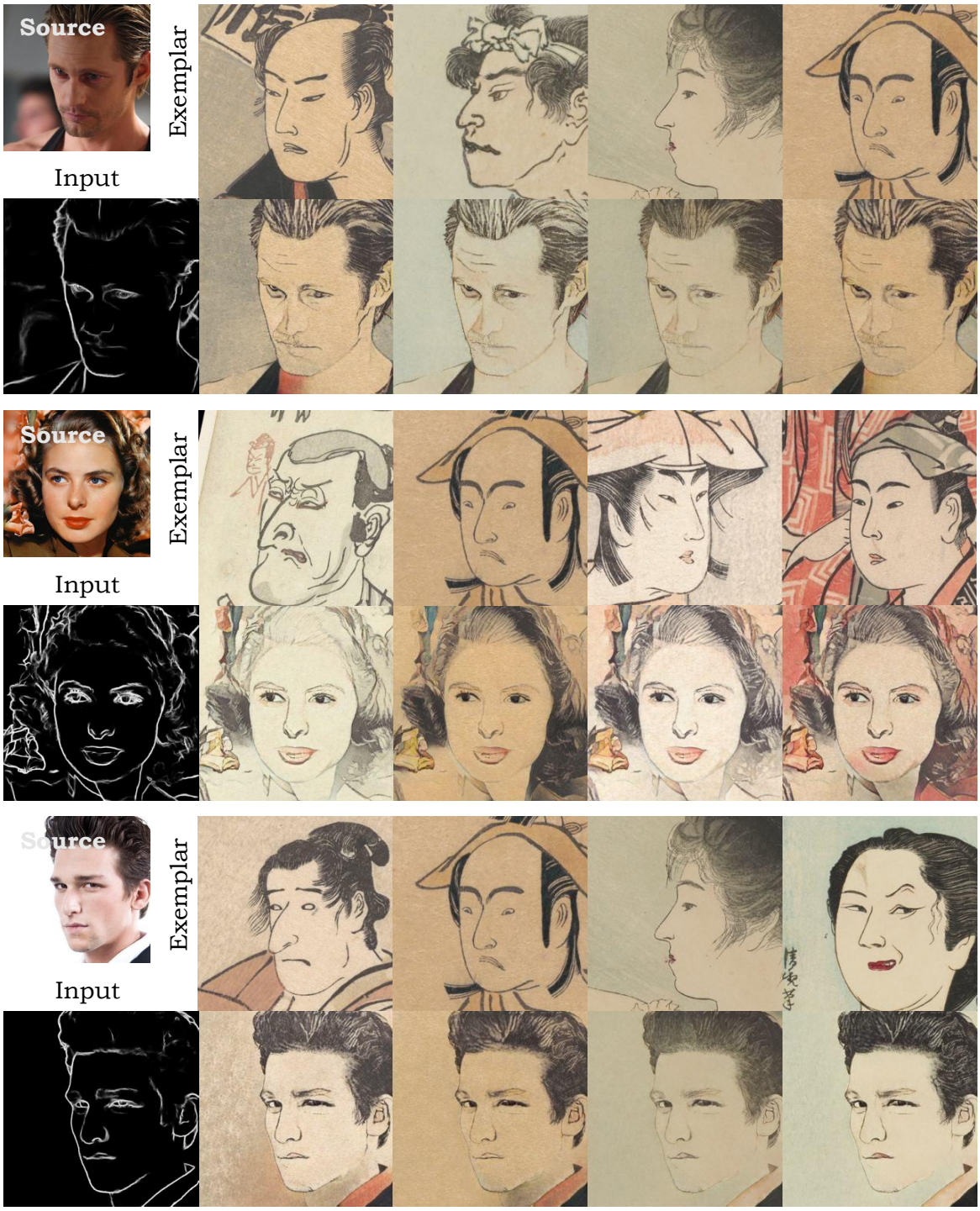

Figure 29: Style transfer portrait, given a face portrait, our methods can transform it into a custom portrait with an example style.

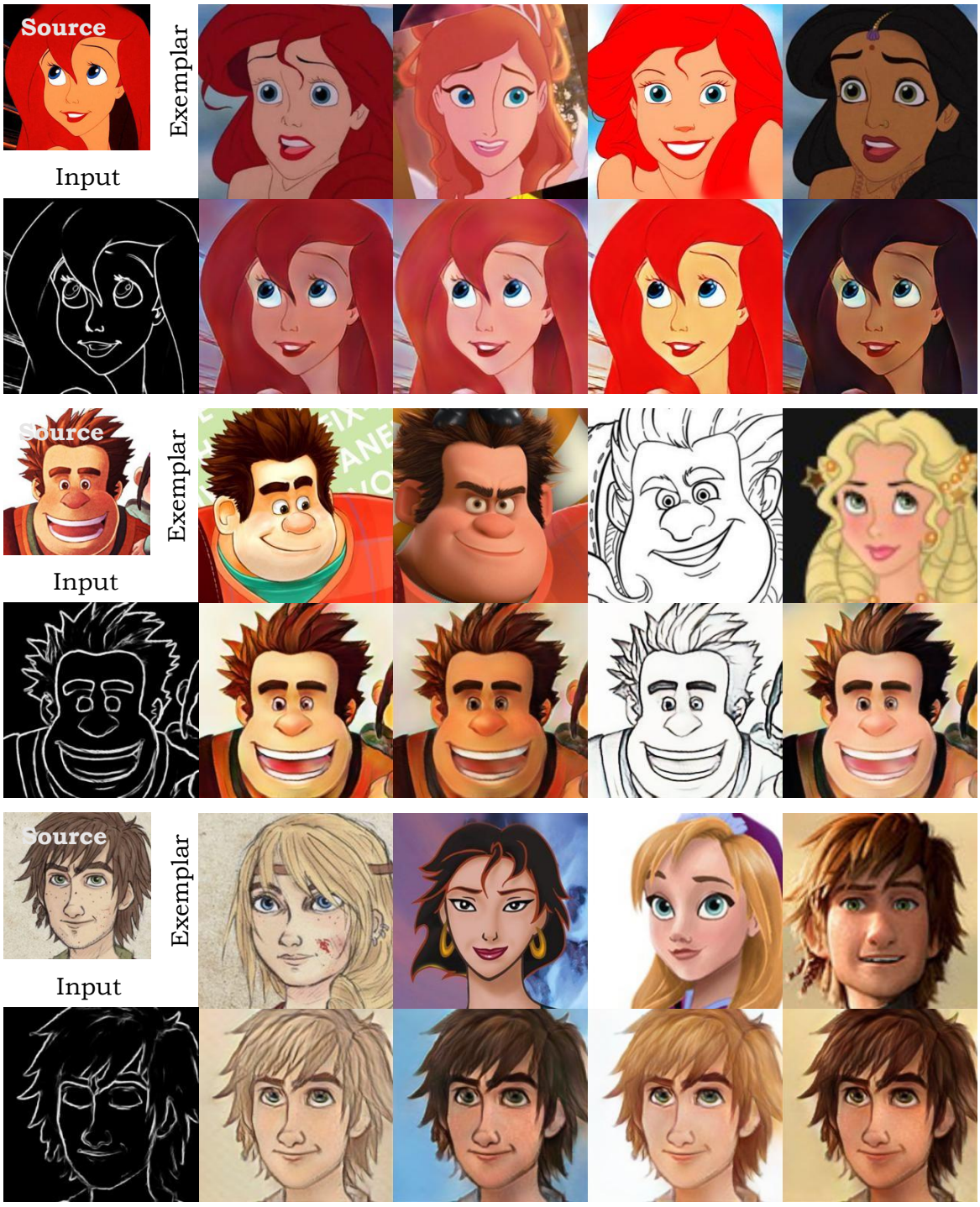

Figure 30: Our results of edge to face synthesis (Cartoon dataset). First row: exemplars. Second row: our results.

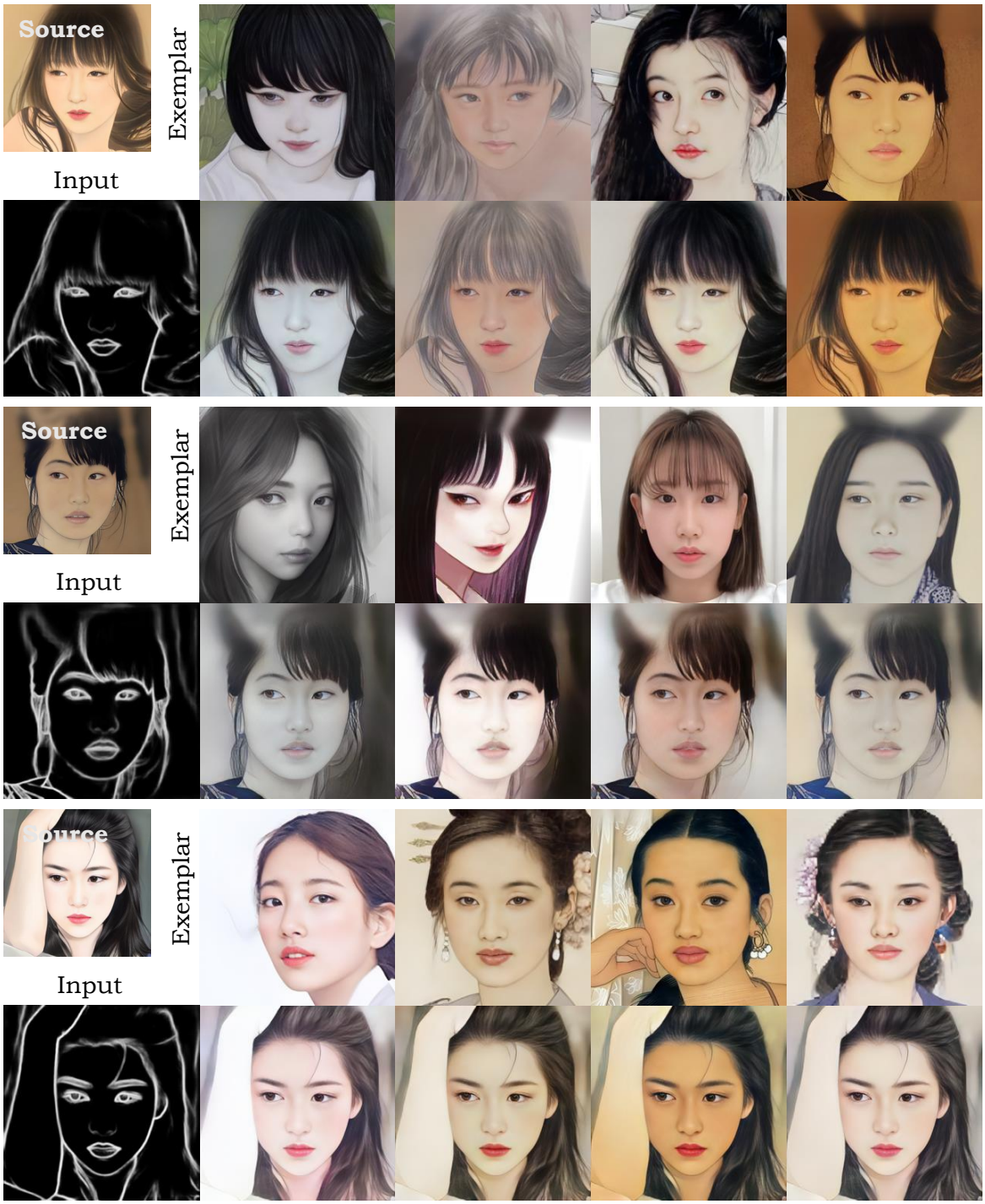

Figure 31: Our results of edge to face synthesis (Meticulous Painting dataset). First row: exemplars. Second row: our results.

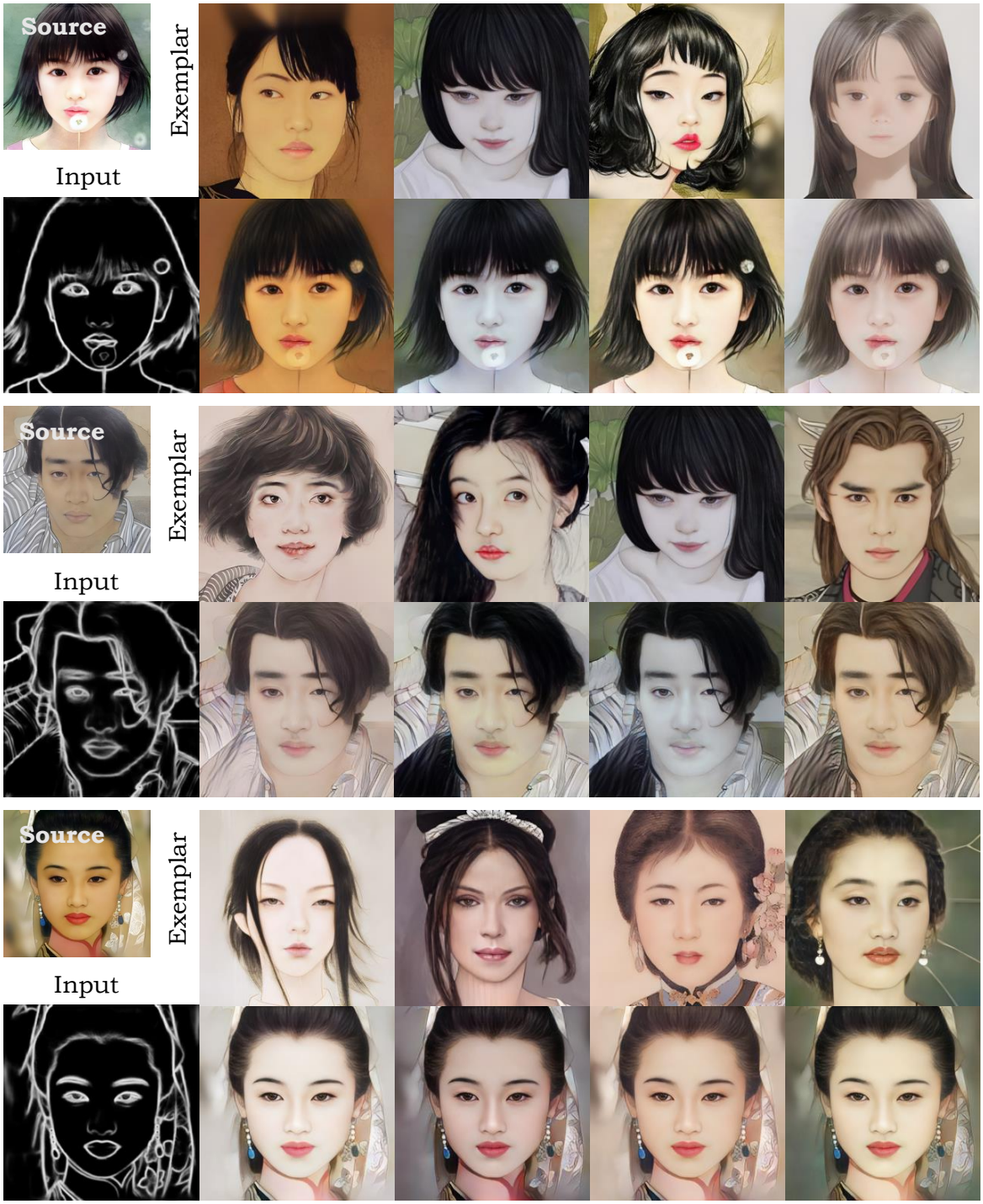

Figure 32: Our results of edge to face synthesis (Meticulous Painting dataset). First row: exemplars. Second row: our results.

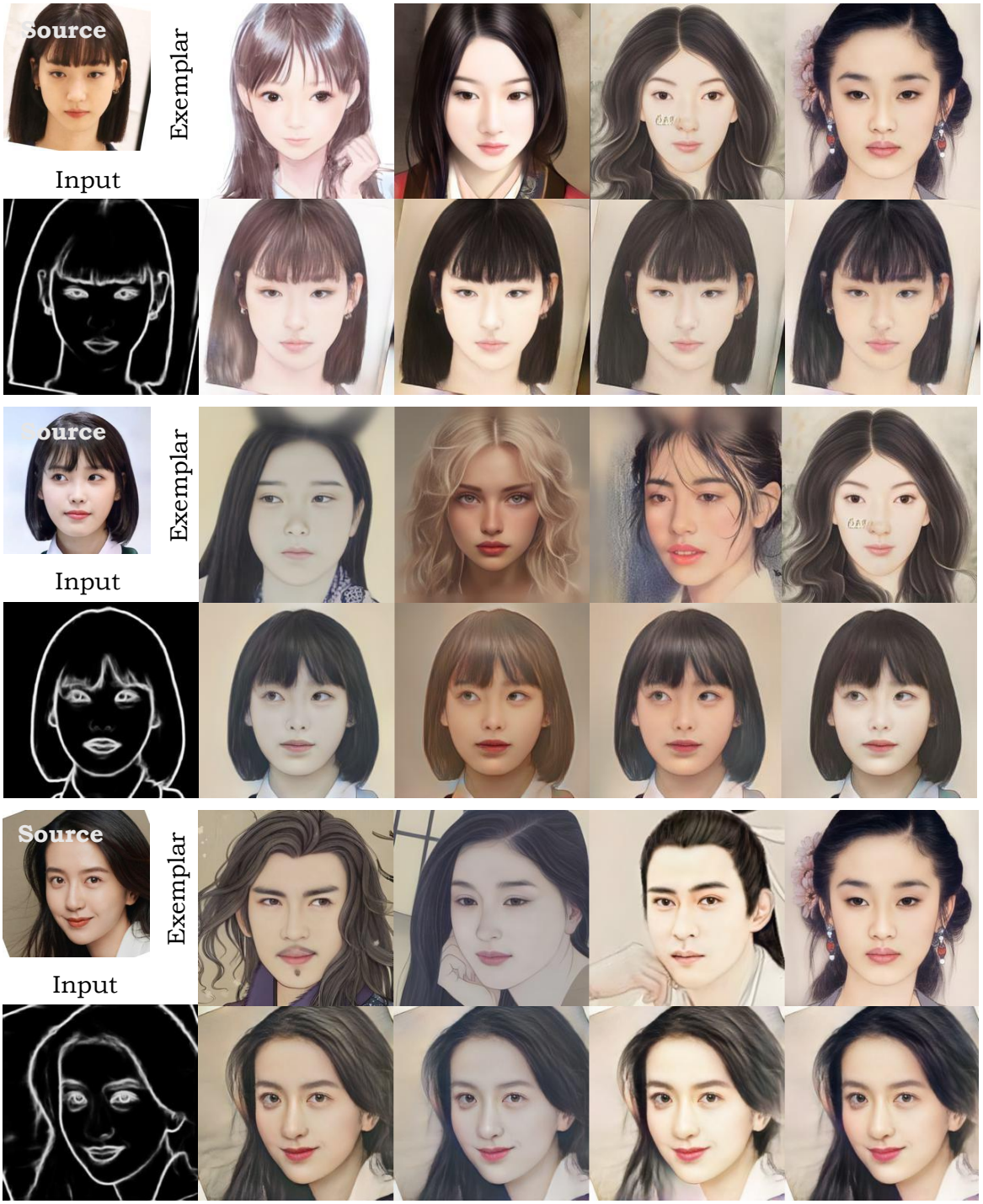

Figure 33: Style transfer portrait, given a face portrait, our methods can transform it into a custom portrait with an example style.

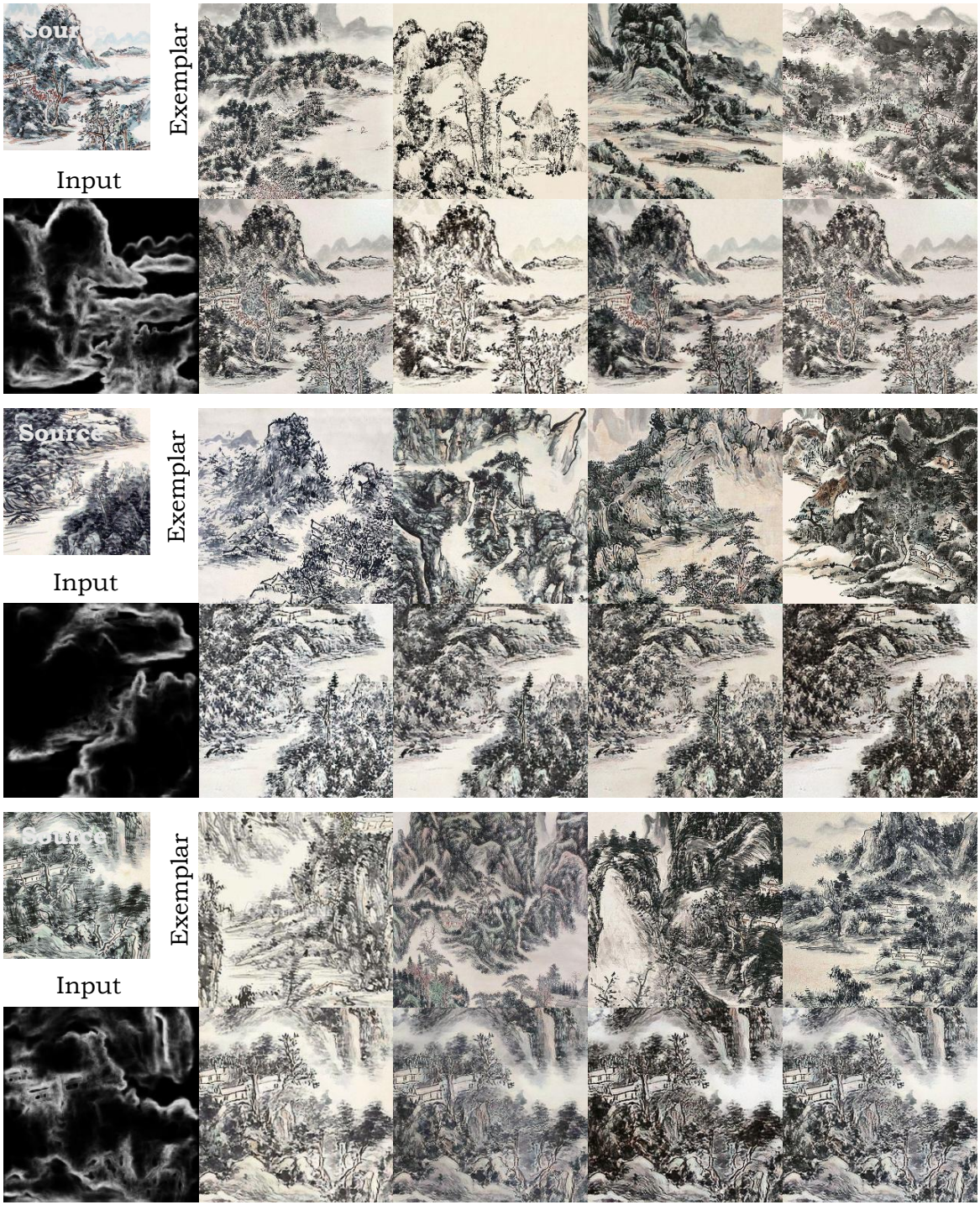

Figure 34: Our results of edge to Landscape painting (landscape dataset). First row: exemplars. Second row: our results.

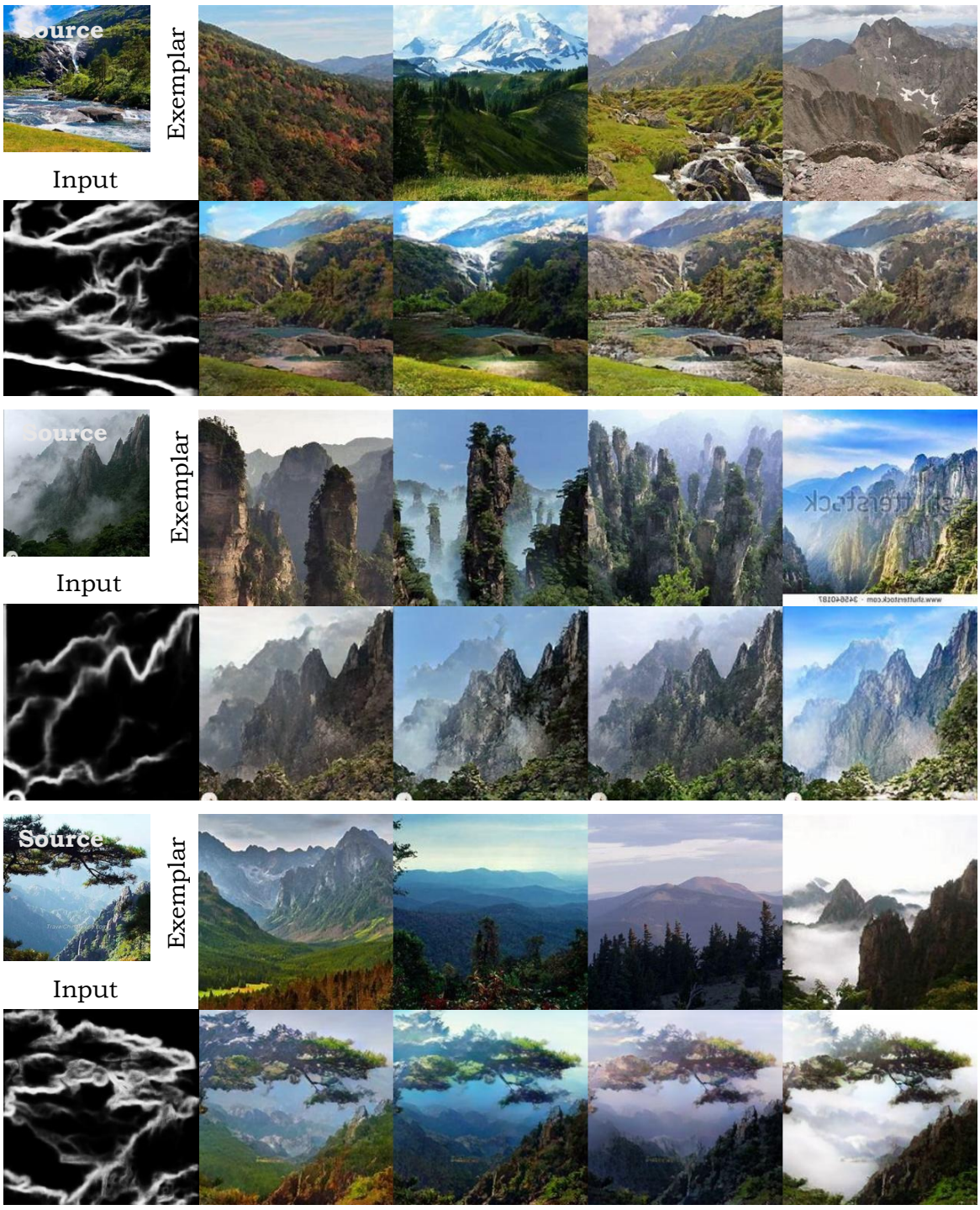

Figure 35: Our results of edge to Scenes (Scenes dataset). First row: exemplars. Second row: our results.

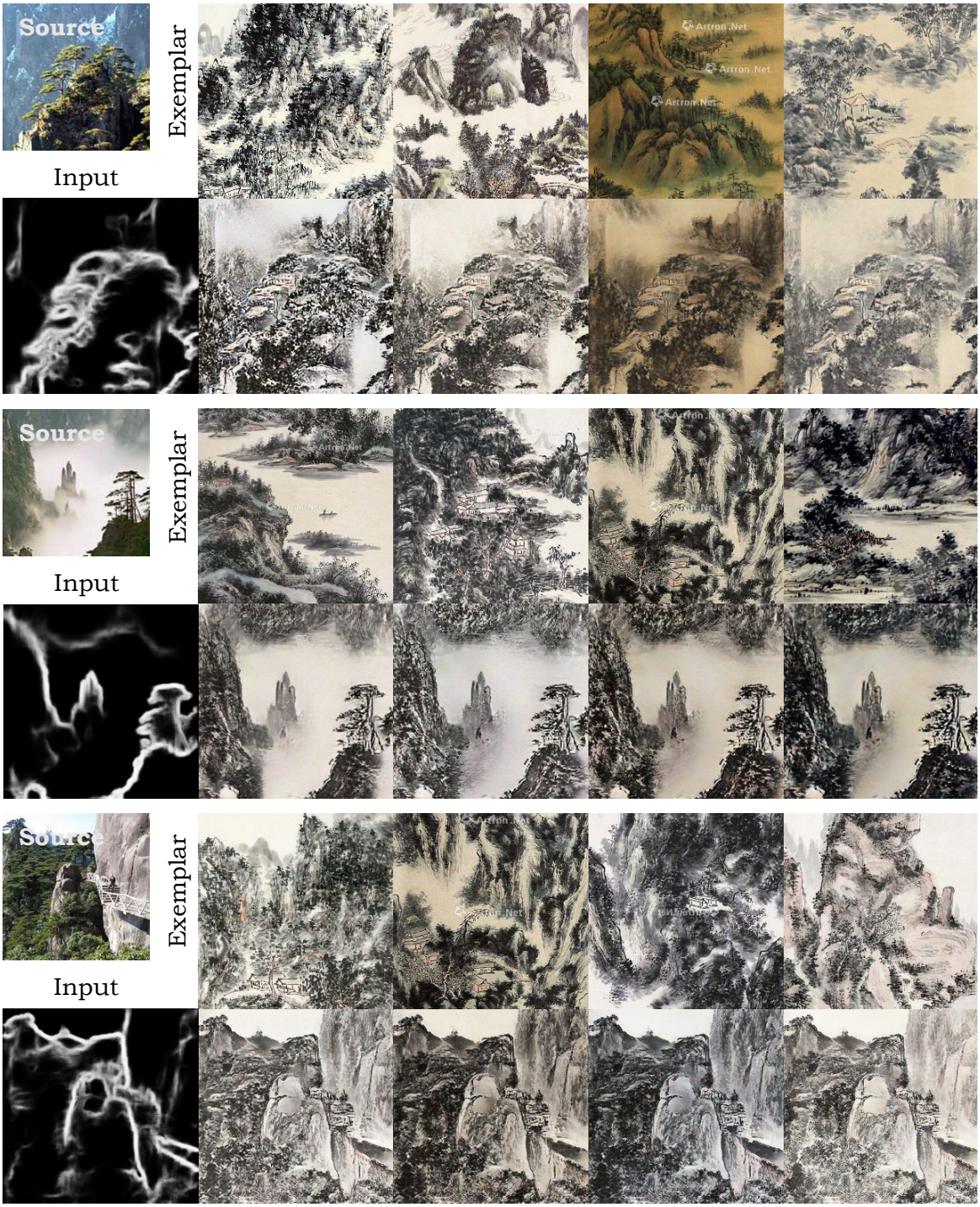

Figure 36: **Scene style transfer**. input a scene edge map can generate a landscape painting style image.