Attention-aware Multi-stroke Style Transfer Supplementary Materials

Yuan Yao[1], Jianqiang Ren[2], Xuansong Xie[2], Weidong Liu[1], Yong-Jin Liu[1], Jun Wang[3]
[1]Tsinghua University, [2]Alibaba Group, [3]University College London

In this supplementary material we elaborate on implementation details of network architecture, as well as show extended results for our style transfer method and multi-stroke fusion control.

1. Implementation Details

We assemble self-attention module into the bottleneck layer of an encoder-decoder framework to form our self-attention autoencoder. The network architecture of our model is presented in Figure 2 of the main paper. In this section, we present more details of our self-attention autoencoder.

1.1. Encoder-decoder Architecture

Table 1 and 2 illustrate the detailed configurations of the encoder and decoder, respectively. The encoder is made of the first few layers of the VGG-19 [4] network. We take input image with size $512 \times 512 \times 3$ as an example and list the feature size for each layer. The max pooling operation is replace by the average pooling operation. The decoder is symmetric to the encoder, with all pooling layers replaced by nearest up-sampling. All convolutional layers use reflection padding to avoid border artifacts [1]. There are some notations; N: the number of output channels, K: kernel size, S: stride size.

As suggested in [2, 3], it is advantageous to match features across different levels in the VGG-19 encoder to fully capture the charateristics of the style. We use skip connections to perform style enhancement using adaptive instance normalization [1]. The three connections are $conv1\_1 \rightarrow inv\_con1\_2$, $conv2\_1 \rightarrow inv\_con2\_2$, $conv3\_1 \rightarrow inv\_con3\_2$, feeding with both output features.

1.2. Self-Attention Module

The architecture of the self-attention module is shown in Figure 1. Different from the way used in [5], where the output of self-attention feature map is added back to the input feature map to learn non-local evidence. We proposed to obtain a self-attention residual R_x by multiplying the feature map f_x with self-attention feature map A_x, and find it is effective to capture saliency characteristics.

2. Extra Ablation Study

2.1. Effects of style enhancement.

To demonstrate the capability of skip connections for style enhancement. We present the stylized results without the connections in Figure 2. By matching features across multiple levels, the results could capture more low-level characteristics (e.g., colors) of style images, thus exhibit higher fidelity to styles in terms of color saturation.

2.2. Effects of sparse loss L_{att}.

The sparse loss L_{att} is to encourage the self-attention module to pay attention to small salient regions instead of the whole image. We retrain our model without L_{att} and present the result in Fig. 3. It can be seen in Fig. 3(d) that without the sparse loss, the network tends to predict more trival parts as salient regions.

3. More Results of Our Method

In this part, we show some additional stylization results by the proposed method, as visualized in Figure 7 and 8. Following the default setting in the main paper, we use three stroke scenario for the proposed style transfer here.

4. Multi-stroke Fusion Control

Here we explain more details of the advantage of our attention-aware multi-stroke method.

4.1. Stroke control vs weight control

The weight control refers to controlling the balance between stylization and content preservation. This strategy has been adopted in previous style transfer methods [1, 2, 3]. As visualized in Figure 4, the weight control strategy directly interpolate on deep feature space as weighted sum of content and stylized features, demonstrating minor variations among range [0, 1]. Our multi-scale style swap enables continuous and discriminative stylized patterns by changing the scale coefficient in eq.(8) of the paper, and further generate integrated results via different combinations efficiently.

Table 1: Details of the encoder. We take input image with size $512 \times 512 \times 3$ as an example.

Layer	Layer Information	Feature Size
*conv*1_1	Conv(N64, K3x3, S1), ReLU	(512, 512, 3) $\rightarrow$ (512, 512, 64)
*conv*1_2	Conv(N64, K3x3, S1), ReLU	(512, 512, 64) $\rightarrow$ (512, 512, 64)
*pool*_1	AveragePooling(K2x2, S2)	(512, 512, 64) $\rightarrow$ (256, 256, 64)
*conv*2_1	Conv(N128, K3x3, S1), ReLU	(256, 256, 64) $\rightarrow$ (256, 256, 128)
*conv*2_2	Conv(N128, K3x3, S1), ReLU	(256, 256, 128) $\rightarrow$ (256, 256, 128)
*pool*_2	AveragePooling(K2x2, S2)	(256, 256, 128) $\rightarrow$ (128, 128, 128)
*conv*3_1	Conv(N256, K3x3, S1), ReLU	(128, 128, 128) $\rightarrow$ (128, 128, 256)
*conv*3_2	Conv(N256, K3x3, S1), ReLU	(128, 128, 256) $\rightarrow$ (128, 128, 256)
*conv*3_3	Conv(N256, K3x3, S1), ReLU	(128, 128, 256) $\rightarrow$ (128, 128, 256)
*conv*3_4	Conv(N256, K3x3, S1), ReLU	(128, 128, 256) $\rightarrow$ (128, 128, 256)
*pool*_3	AveragePooling(K2x2, S2)	(128, 128, 256) $\rightarrow$ (64, 64, 256)
*conv*4_1	Conv(N512, K3x3, S1), ReLU	(64, 64, 256) $\rightarrow$ (64, 64, 512)

Table 2: Details of the decoder.

Layer	Layer Information	Feature Size
*inv_conv*4_1	Conv(N256, K3x3, S1), ReLU	(64, 64, 512) $\rightarrow$ (64, 64, 256)
*upsample*_1	Nearest Upsampling(x2)	(64, 64, 256) $\rightarrow$ (128, 128, 256)
*inv_conv*3_4	Conv(N256, K3x3, S1), ReLU	(128, 128, 256) $\rightarrow$ (128, 128, 256)
*inv_conv*3_3	Conv(N256, K3x3, S1), ReLU	(128, 128, 256) $\rightarrow$ (128, 128, 256)
*inv_conv*3_2	Conv(N256, K3x3, S1), ReLU	(128, 128, 256) $\rightarrow$ (128, 128, 256)
*inv_conv*3_1	Conv(N128, K3x3, S1), ReLU	(128, 128, 256) $\rightarrow$ (128, 128, 128)
*upsample*_2	Nearest Upsampling(x2)	(128, 128, 128) $\rightarrow$ (256, 256, 128)
*inv_conv*2_2	Conv(N128, K3x3, S1), ReLU	(256, 256, 128) $\rightarrow$ (256, 256, 128)
*inv_conv*2_1	Conv(N64, K3x3, S1), ReLU	(256, 256, 128) $\rightarrow$ (256, 256, 64)
*upsample*_3	Nearest Upsampling(x2)	(256, 256, 64) $\rightarrow$ (512, 512, 64)
*inv_conv*1_2	Conv(N64, K3x3, S1), ReLU	(512, 512, 64) $\rightarrow$ (512, 512, 64)
*inv_conv*1_1	Conv(N3, K3x3, S1), ReLU	(512, 512, 64) $\rightarrow$ (512, 512, 3)

4.2. Fusion control strategy

As mentioned in Section 4.4, our method can effectively integrate multiple stroke patterns with different control strategies. We present the fusion procedure in Figure 5. Given *K*+1 stroke feature maps $(f_{cs}^0, f_{cs}^1, \ldots, f_{cs}^K)$ and the corresponding attention map $\hat{A}_c$, we first generate *K*+1 clustering centers with attention values according to eq.(10), and then assign sequentially for stroke sizes, with higher attention values for finer stroke patterns. The integrated feature map is the weighted sum of the *K*+1 stroke feature maps based on eq. (11-12) in the paper.

The level of detail for multi-stroke fusion can be controlled by the smoothing factor γ. As visualized in Figure 6, we present the influence of different smoothing values for four-stroke fusion scenario. The larger it is, the more contributions for single stroke on corresponding attention area, leading to a more discriminative effect among these stroke patterns.

References

[1] Xun Huang and Serge J Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, 2017.

[2] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. Universal style transfer via feature transforms. In *NIPS*, 2017.

[3] Lu Sheng, Ziyi Lin, Jing Shao, and Xiaogang Wang. Avatar-net: Multi-scale zero-shot style transfer by feature decoration. In *CVPR*, 2018.

[4] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[5] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. *arXiv:1805.08318*, 2018.

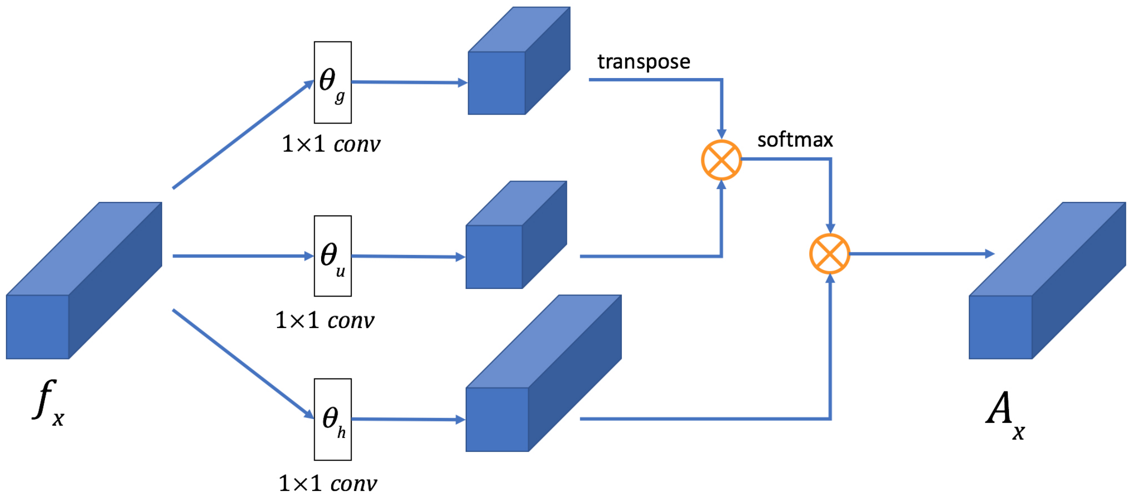

Figure 1: The architecture of our self-attention module. The $\otimes$ denotes matrix multiplication.

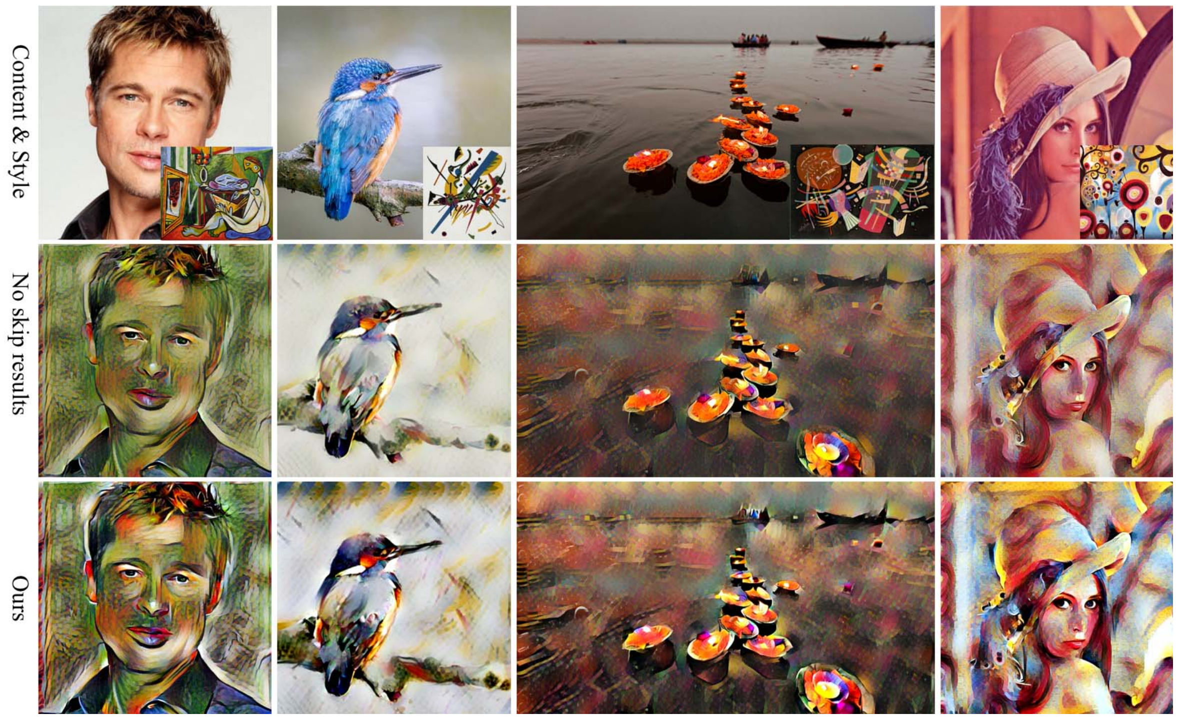

Figure 2: The comparison between method without the style enhancement by removing the skip connections. By matching multiple low level features, our results exhibit higher fidelity to styles in terms of color saturation.

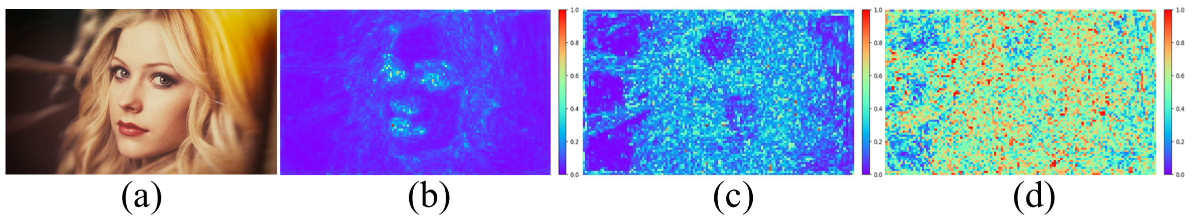

Figure 3: The effect of sparse loss L_{att} with different weight λ_{att}. (a) Content. (b) $|A_x|$ with $\lambda_{att} = 6$ (paper setting). (c) $|A_x|$ with $\lambda_{att} = 1$. (d) $|A_x|$ with $\lambda_{att} = 0$ (without L_{att}).

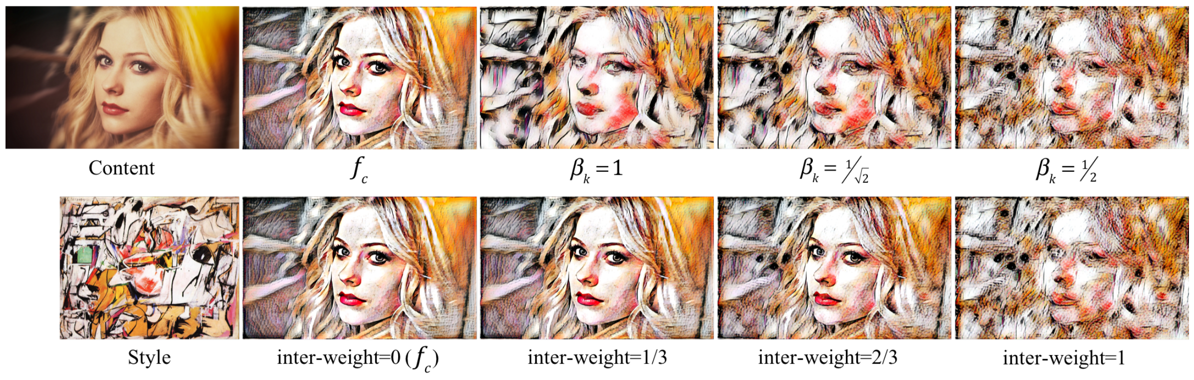

Figure 4: Comparison between our multi-scale stroke control and generally used weight control. The weight control simply conduct interpolation as weighted sum of content and stylized features, which demonstrate minor variations among range [0,1]. Our method could flexibly generate continuous and discriminative stylized patterns.

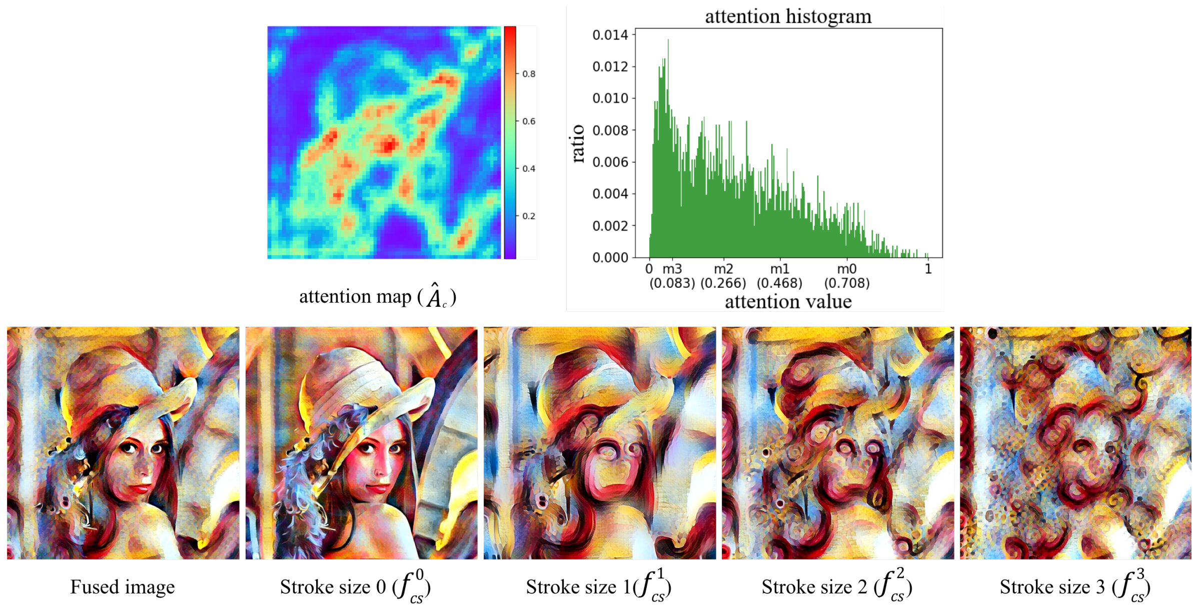

Figure 5: The procedure for our multi-stroke fusion. In the attention histogram, we mark the attention value of clustering centers obtained by applying k-means on the attention map $\hat{A}_c$, with higher attention values assigned to finer stroke patterns. The integrated feature map is generated seamlessly as the weighted sum of these stroke feature maps according to the proposed fusion strategy.

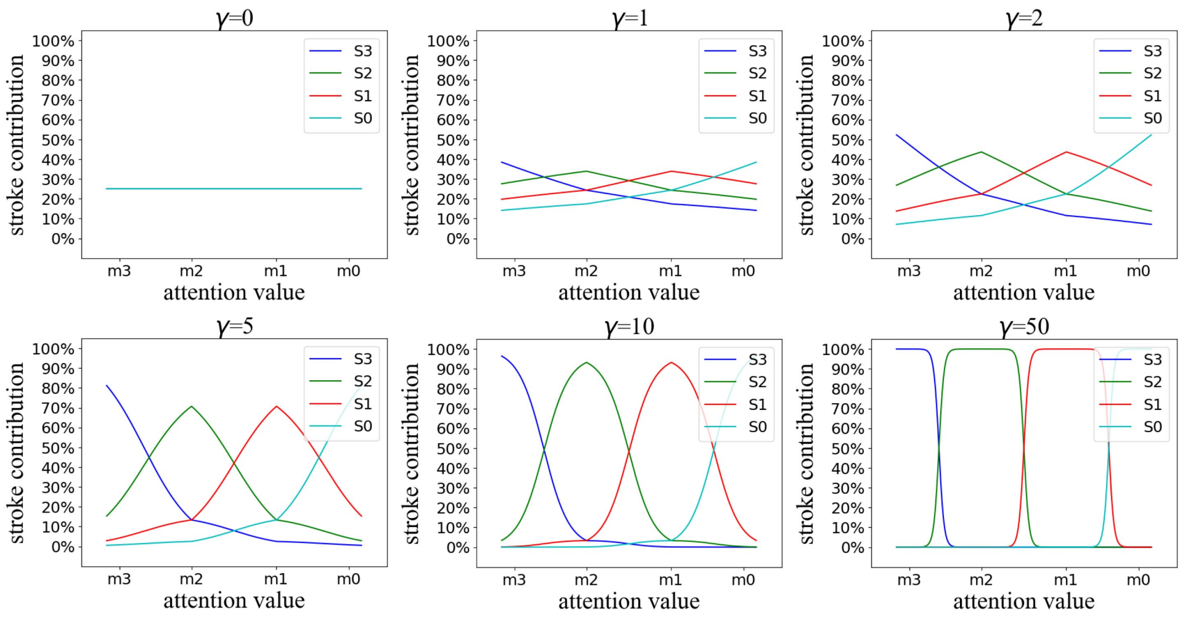

Figure 6: Varying values of smoothing factor γ affecting the fusion contribution for each stroke pattern. Each stroke size is assigned with an intensity center for a specific attention area. With the increasing of γ, each stroke pattern tends to contribute more on its own attention area, leading to a more discriminative effect among these stroke patterns.

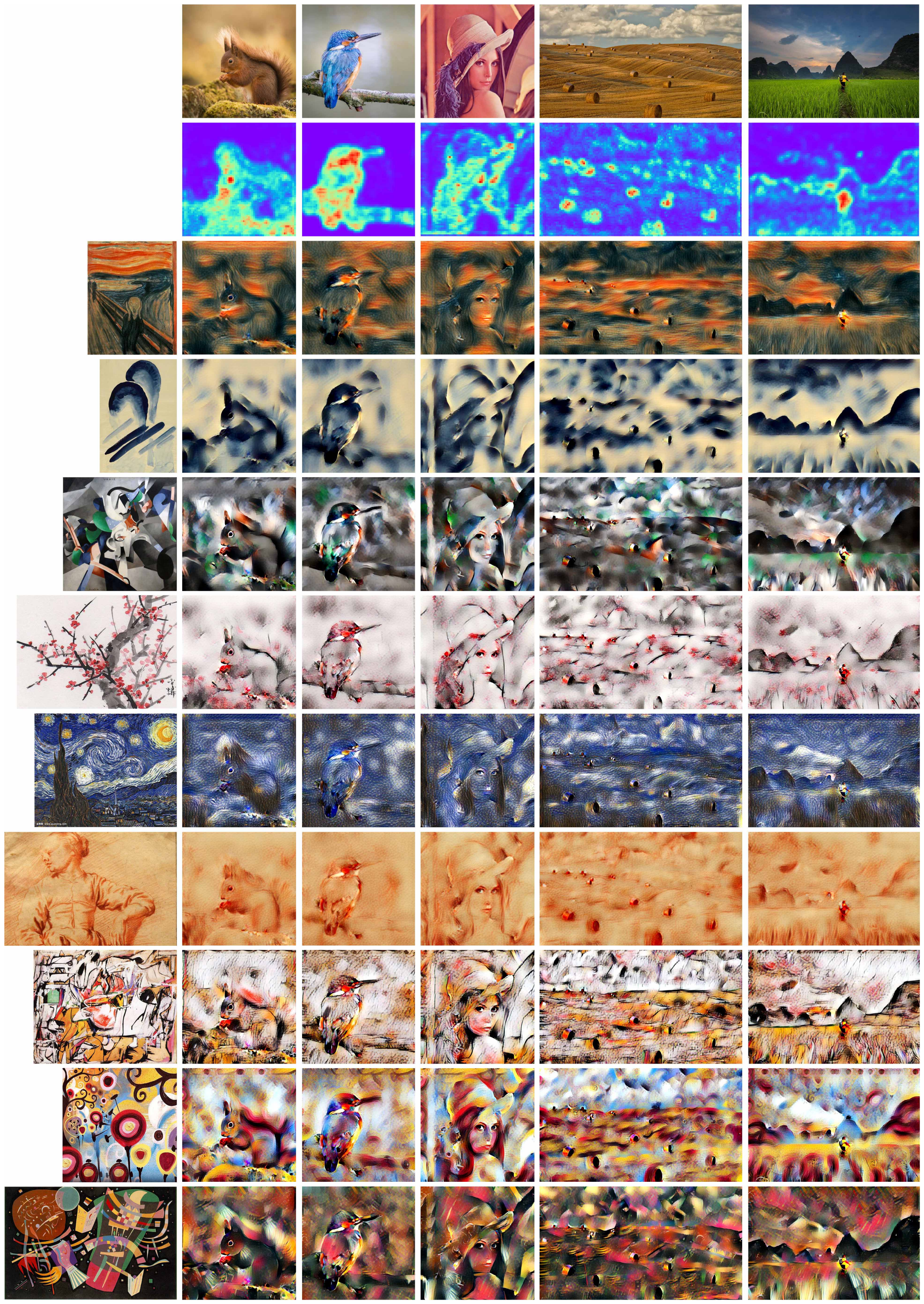

Figure 7: More results of our attention-aware multi-stroke style transfer method (Set 1). The **1st row** is the content images, the **2nd row** is the corresponding attention maps and the **1st column** is the style images.

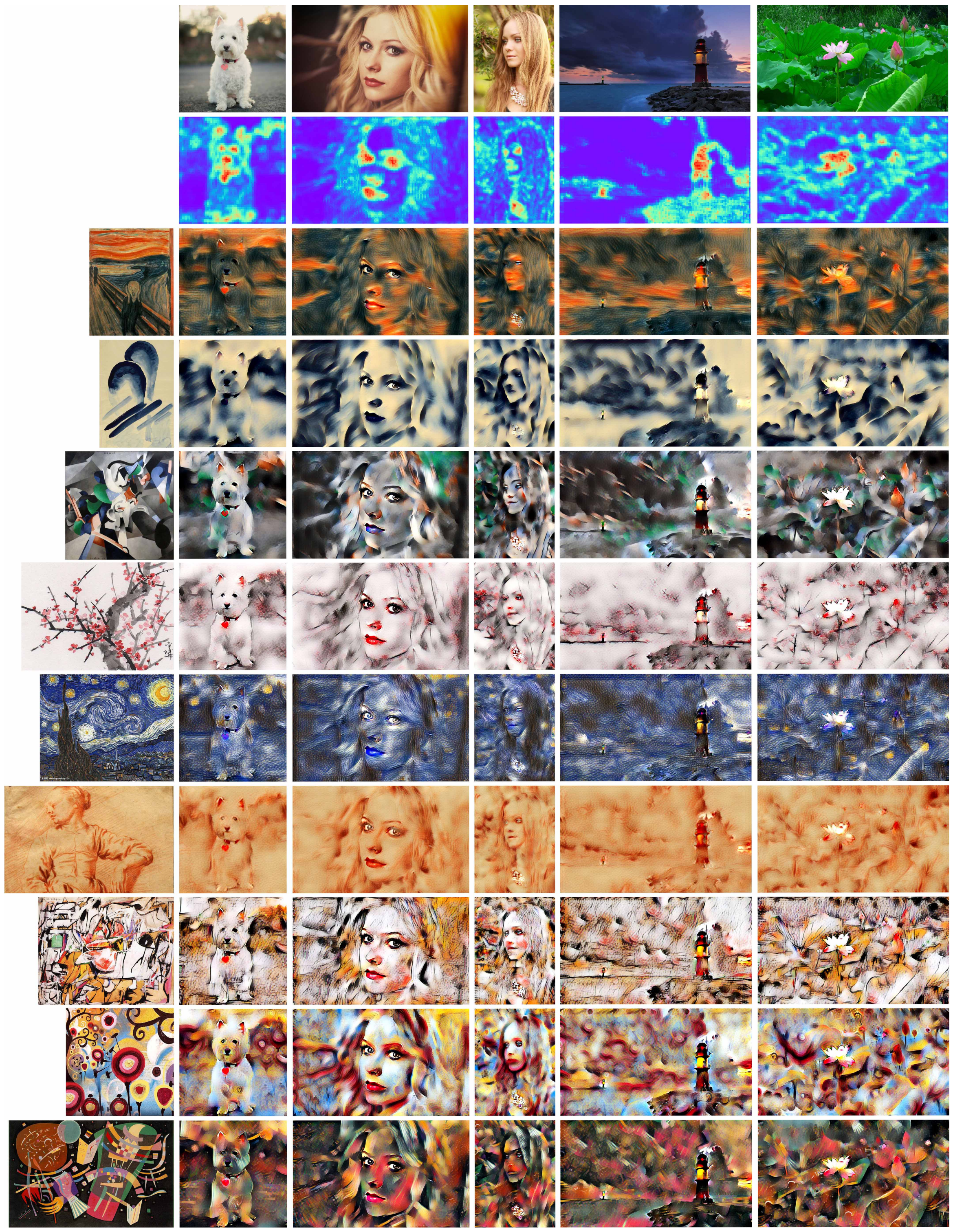

Figure 8: More results of our attention-aware multi-stroke style transfer method (Set 2). The **1st row** is the content images, the **2nd row** is the corresponding attention maps and the **1st column** is the style images.