Dae Young Park[1,*] and Kwang Hee Lee[1,*]

[1]Artificial Intelligence Research Institute, Korea
[1]likebullet86@gmail.com, [2]lkwanghee@gmail.com

Figure 1: Result of the proposed SANet. We can transfer various styles to content images with high-quality.

Abstract

Arbitrary style transfer aims to synthesize a content image with the style of an image to create a third image that has never been seen before. Recent arbitrary style transfer algorithms find it challenging to balance the content structure and the style patterns. Moreover, simultaneously maintaining the global and local style patterns is difficult due to the patch-based mechanism. In this paper, we introduce a novel style-attentional network (SANet) that efficiently and flexibly integrates the local style patterns according to the semantic spatial distribution of the content image. A new identity loss function and multi-level feature embeddings enable our SANet and decoder to preserve the content structure as much as possible while enriching the style patterns. Experimental results demonstrate that our algorithm synthesizes stylized images in real-time that are higher in quality than those produced by the state-of-the-art algorithms.

1. Introduction

Artistic style transfer is a technique used to create art by synthesizing global and local style patterns from a given style image evenly over a content image while maintaining its original structure. Recently, the seminal work of Gatys et al. [5] showed that the correlation between features extracted from a pre-trained deep neural network can capture the style patterns well. The method by Gatys et al. [5] is

[*] indicates equal contribution

flexible enough to combine the content and style of arbitrary images, but is prohibitively slow due to the iterative optimization process.

Significant efforts have been made to reduce the computational cost of this process. Several approaches [1, 8, 12, 22, 3, 14, 19, 26, 29] have been developed based on feedforward networks. Feedforward methods can synthesize stylized images efficiently, but are limited to a fixed number of styles or provide insufficient visual quality.

For arbitrary style transfer, a few methods [13, 7, 20] holistically adjust the content features to match the second-order statistics of the style features. AdaIN [7] simply adjusts the mean and variance of the content image to match those of the style image. Although AdaIN effectively combines the structure of the content image and the style pattern by transferring feature statistics, its output suffers in quality due to the over-simplified nature of this method. WCT [13] transforms the content features into the style feature space through a whitening and coloring process with the covariance instead of the variance. By embedding these stylized features within a pre-trained encoder–decoder module, the style-free decoder synthesizes the stylized image. However, if the feature has a large number of dimensions, WCT will accordingly require computationally expensive operations. Avatar-Net [20] is a patch-based style decorator module that maps the content features with the characteristics of the style patterns while maintaining the content structure. Avatar-Net considers not only the holistic style distribution, but also the local style patterns. However, despite valuable efforts, these methods still do not reflect the detailed texture of the style image, distort content structures, or fail to balance the local and global style patterns.

In this work, we propose a novel arbitrary style transfer algorithm that synthesizes high-quality stylized images in real time while preserving the content structure. This is achieved by a new style-attentional network (SANet) and a novel identity loss function. For arbitrary style transfer, our feedforward network, composed of SANets and decoders, learns the semantic correlations between the content features and the style features by spatially rearranging the style features according to the content features.

Our proposed SANet is closely related to the style feature decorator of Avatar-Net [20]. There are, however, two main differences: The proposed model uses 1) a learned similarity kernel instead of a fixed one and 2) soft attention instead of hard attention. In other words, we changed the self-attention mechanism to a learnable soft-attention-based network for the purpose of style decoration. Our SANet uses the learnable similarity kernel to represent the content feature map as a weighted sum of style features that are similar to each of its positions. Using the identity loss during the training, the same image pair are input and our model is trained to restore the same result. At inference time, one of the input images is replaced with the style image, and the content image is restored as much as possible based on the style features. Identity loss, unlike the content–style trade-off, helps to maintain the content structure without losing the richness of the style because it helps restore the contents based on style features. The main contributions of our work are as follows:

- We propose a new SANet to flexibly match the semantically nearest style features onto the content features.
- We present a learning approach for a feedforward network composed of SANets and decoders that is optimized using a conventional style reconstruction loss and a new identity loss.
- Our experiments show that our method is highly efficient (about 18–24 frames per second (fps) at 512 pixels) at synthesizing high-quality stylized images while balancing the global and local style patterns and preserving content structure.

2. Related Work

Arbitrary Style Transfer. The ultimate goal of arbitrary style transfer is to simultaneously achieve and preserve generalization, quality, and efficiency. Despite recent advances, existing methods [5, 4, 1, 8, 12, 22, 3, 6, 10, 11, 23, 24, 28, 18] present a trade-off among generalization, quality, and efficiency. Recently, several methods [13, 20, 2, 7] have been proposed to achieve arbitrary style transfer. The AdaIN algorithm simply adjusts the mean and variance of the content image to match those of the style image by transferring global feature statistics. WCT performs a pair of feature transforms, whitening and coloring, for feature embedding within a pre-trained encoder-decoder module. Avatar-Net introduced the patch-based feature decorator, which transfers the content features to the semantically nearest style features while simultaneously minimizing the difference between their holistic feature distributions. In many cases, we observe that the results of WCT and Avatar-Net fail to sufficiently represent the detailed texture or maintain the content structure. We speculate that WCT and Avatar-Net could fail to synthesize the detailed texture style due to their pre-trained general encoder–decoder networks, which are learned from general images such as MS-COCO datasets [15] with large differences in style characteristics. As a result, these methods consider mapping the style feature onto the content feature in the feature space, but there is no way to control the global statistics or content structure of the style. Although Avatar-Net can obtain the local style patterns through a patch-based style decorator, the scale of style patterns in the style images depends on the patch size. Therefore, the global and local style patterns cannot both be taken into consideration. In contrast, AdaIN transforms texture and color distribution well, but does not represent local style patterns well. In this method,

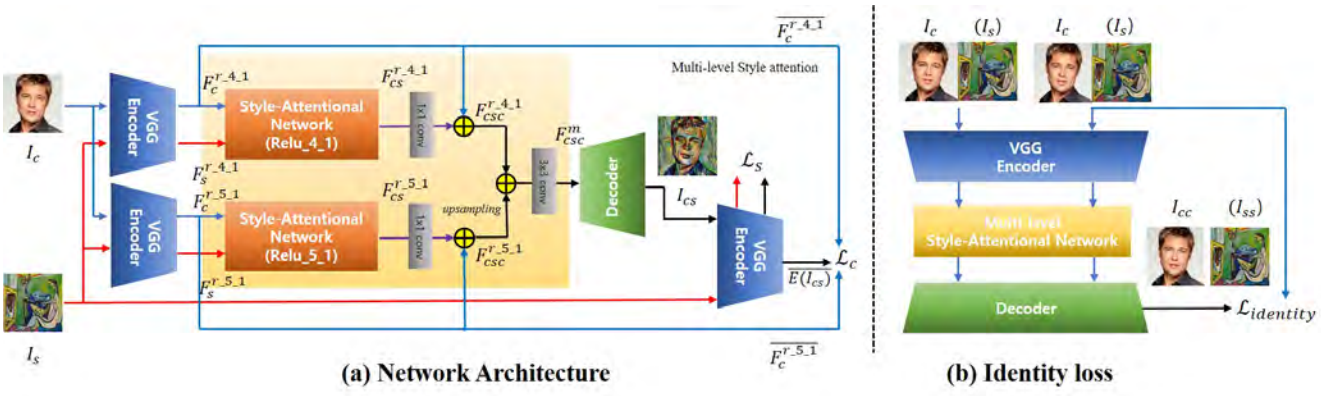

Figure 2: Overview of training flow. (a) Fixed VGG encoder encoding content and style images. Two SANets map features from `Relu_4_1` and `Relu_5_1` features respectively. The decoder transforms the combined SANet output features to I_{cs} (Eq. 4). The fixed VGG encoder is used to compute $\mathcal{L}_c$ (Eq. 7) and $\mathcal{L}_s$ (Eq. 8). (b) The identity loss $\mathcal{L}_{identity}$ (Eq. 9) quantifies the difference between I_c and I_{cc} or between I_s and I_{ss}, where I_c (I_s) is the original content (style) image and I_{cc} (I_{ss}) is the output image synthesized from the image pair (content or style).

there exists another trade-off between content and style due to a combination of scale-adapted content and style loss. In this paper, we try to solve these problems using the SANets and the proposed identity loss. In this way, the proposed style transfer network can represent global and local style patterns and maintain the content structure without losing the richness of the style.

Self-Attention Mechanism. Our style-attentional module is related to the recent self-attention methods [25, 30] for image generation and machine translation. These models calculate the response at a position in a sequence or an image by attending to all positions and taking their weighted average in an embedding space. The proposed SANet learns the mapping between the content features and the style features by slightly modifying the self-attention mechanism.

3. Method

The style transfer network proposed in this paper is composed of an encoder–decoder module and a style-attentional module, as shown in Fig. 2. The proposed feedforward network effectively generates high-quality stylized images that appropriately reflect global and local style patterns. Our new identity loss function helps to maintain the detailed structure of the content while reflecting the style sufficiently.

3.1. Network Architecture

Our style transfer network takes a content image I_c and an arbitrary style image I_s as inputs, and synthesizes a stylized image I_{cs} using the semantic structures from the former and characteristics from the latter. In this work, the pretrained VGG-19 network [21] is employed as encoder and a symmetric decoder and two SANets are jointly trained for arbitrary style transfer. Our decoder follows the settings of [7].

To combine global style patterns and local style patterns adequately, we integrate two SANets by taking the VGG feature maps encoded from different layers (`Relu_4_1` and `Relu_5_1`) as inputs and combining both output feature maps. From a content image I_c and style image I_s pair, we first extract their respective VGG feature maps $F_c = E(I_c)$ and $F_s = E(I_s)$ at a certain layer (e.g., `Relu_4_1`) of the encoder.

After encoding the content and style images, we feed both feature maps to a SANet module that maps the correspondences between the content feature map F_c and the style feature map F_s, producing following the output feature map:

$$F_{cs} = SANet\,(F_c, F_s) \tag{1}$$

After applying 1×1 convolution to F_{cs} and summing the two matrices element-wise as follows, we obtain F_{csc}:

$$F_{csc} = F_c + W_{cs}F_{cs}, \tag{2}$$

where "+" denotes element-wise summation.

We combine two the output feature maps from the two SANets as

$$F^m_{csc} = conv_{3\times3}(F^{\text{r\_4\_1}}_{csc} + upsampling(F^{\text{r\_5\_1}}_{csc})), \tag{3}$$

where $F^{\text{r\_4\_1}}_{csc}$ and $F^{\text{r\_5\_1}}_{csc}$ are the output feature maps obtained from the two SANets, $conv_{3\times3}$ denotes the 3×3 convolution used to combine the two feature maps, and $F^{\text{r\_5\_1}}_{csc}$ is added to $F^{\text{r\_4\_1}}_{csc}$ after upsampling.

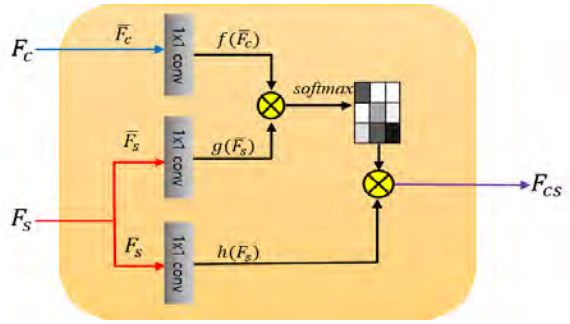

Figure 3: SANet.

Then, the stylized output image I_{cs} is synthesized by feeding F^m_{csc} into the decoder as follows:

$$I_{cs} = D\left(F^m_{csc}\right). \tag{4}$$

3.2. SANet for Style Feature Embedding

Figure 3 shows style feature embedding using the SANet module. Content feature maps F_c and style feature maps F_s from the encoder are normalized and then transformed into two feature spaces f and g to calculate the attention between $\overline{F^i_c}$ and $\overline{F^j_s}$ as follows:

$$F^i_{cs} = \frac{1}{C(F)} \sum_{\forall j} \exp(f(\overline{F^i_c})^T g(\overline{F^j_s}))h(F^j_s), \tag{5}$$

where $f(\overline{F_c}) = W_f\overline{F_c}$, $g(\overline{F_s}) = W_g\overline{F_s}$, and $h(F_s) = W_hF_s$. Further, $\overline{F}$ denotes a mean–variance channel-wise normalized version of F. The response is normalized by a factor $C(F) = \sum_{\forall j} \exp(f(\overline{F^i_c})^T g(\overline{F^j_s}))$. Here, i is the index of an output position and j is the index that enumerates all possible positions. In the above formulation, W_f, W_g, and W_h are the learned weight matrices, which are implemented as 1×1 convolutions as in [30].

Our SANet has a network structure similar to the existing non-local block structure [27], but the number of input data differ (the input of the SANet consists of F_c and F_s). The SANet module can appropriately embed a local style pattern in each position of the content feature maps by mapping a relationship (such as affinity) between the content and style feature maps through learning.

3.3. Full Systemq

As shown in Fig. 2, we use the encoder (a pre-trained VGG-19 [21]) to compute the loss function for training the SANet and decoder:

$$\mathcal{L} = \lambda_c\mathcal{L}_c + \lambda_s\mathcal{L}_s + \mathcal{L}_{identity}, \tag{6}$$

where the composers of content, style, and identity loss are $\mathcal{L}_c$, $\mathcal{L}_s$, and $\mathcal{L}_{identity}$, respectively, and λ_c and λ_s are the weights of different losses.

Similar to [7], the content loss is the Euclidean distance between the mean–variance channel-wise normalized target features, $\overline{F_c^{\texttt{r\_4\_1}}}$ and $\overline{F_c^{\texttt{r\_5\_1}}}$ and the mean–variance channel-wise normalized features of the output image VGG features, $\overline{E(I_{cs})^{\texttt{r\_4\_1}}}$ and $\overline{E(I_{cs})^{\texttt{r\_5\_1}}}$, as follows:

$$\mathcal{L}_c = ||\overline{E(I_{cs})^{\texttt{r\_4\_1}}} - \overline{F_c^{\texttt{r\_4\_1}}}||_2 + ||\overline{E(I_{cs})^{\texttt{r\_5\_1}}} - \overline{F_c^{\texttt{r\_5\_1}}}||_2. \tag{7}$$

The style loss is defined as follows:

$$\begin{aligned}\mathcal{L}_s = &\sum_{i=1}^{L} ||\mu(\phi_i(I_{cs})) - \mu(\phi_i(I_s))||_2 \\ &+ ||\sigma(\phi_i(I_{cs})) - \sigma(\phi_i(I_s))||_2,\end{aligned} \tag{8}$$

where each ϕ denotes a feature map of the layer in the encoder used to compute the style loss. We use `Relu_1_1`, `Relu_2_1`, `Relu_3_1`, `Relu_4_1`, and `Relu_5_1` layers with equal weights. We have applied both the Gram matrix loss [5] and the AdaIN style loss [7], but the results show that the AdaIN style loss is more satisfactory.

When W_f, W_g, and W_h are fixed as the identity matrices, each position in the content feature maps can be transformed into the semantically nearest feature in the style feature maps. In this case, the system cannot parse sufficient style features. In the SANet, although W_f, W_g, and W_h are learnable matrices, our style transfer model can be trained by considering only the global statistics of the style loss $\mathcal{L}_s$.

To consider both the global statistics and the semantically local mapping between the content features and the style features, we define a new identity loss function as

$$\begin{aligned}\mathcal{L}_{identity} = &\lambda_{identity1}(||(I_{cc} - I_c)||_2 + ||(I_{ss} - I_s)||_2) \\ &+ \lambda_{identity2}\sum_{i=1}^{L}(||\phi_i(I_{cc}) - \phi_i(I_c)||_2 \\ &+ ||\phi_i(I_{ss}) - \phi_i(I_s)||_2),\end{aligned} \tag{9}$$

where I_{cc}(or I_{ss}) denotes the output image synthesized from two same content (or style) images, each ϕ_i denotes a layer in the encoder, and $\lambda_{identity1}$ and $\lambda_{identity2}$ are identity loss weights. The weighting parameters are simply set as $\lambda_c = 1$, $\lambda_s = 3$, $\lambda_{identity1} = 1$, and $\lambda_{identity2} = 50$ in our experiments.

The content and style losses control the trade-off between the structure of the content image and the style patterns. Unlike the other two losses, the identity loss is calculated from the same input images with no gap in style characteristics. Therefore, the identity loss concentrates

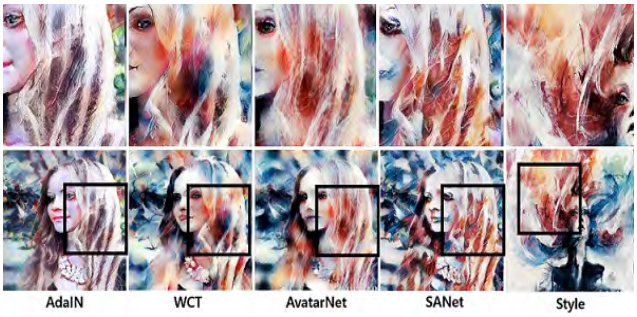

Figure 4: Result details. Regions marked by bounding boxes in the bottom row are enlarged in the top row for better visualization.

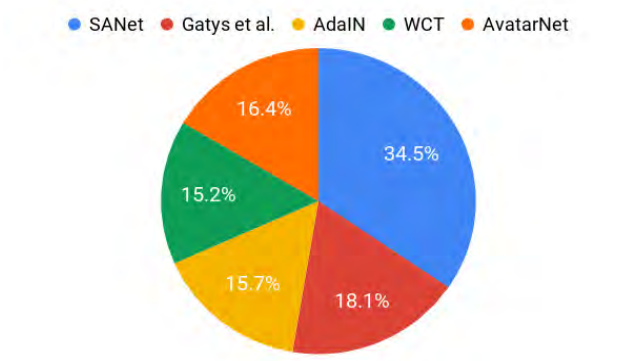

Figure 5: User preference result of five style transfer algorithms.

Method	Time (256 px)	Time (512 px)
Gatys et al. [5]	15.863	50.804
WCT [13]	0.689	0.997
Avatar-Net [20]	0.248	0.356
AdaIN [7]	0.011	0.039
ours (`Relu_4_1`)	0.012	0.042
ours (multi-level)	0.017	0.055

Table 1: Execution time comparison (in seconds).

on keeping the structure of the content image rather than changing the style statistics. As a result, the identity loss makes it possible to maintain the structure of the content image and style characteristics of the reference image simultaneously.

4. Experimental Results

Figure 2 shows an overview of our style transfer network based on the proposed SANets. The demo site will be release at https://dypark86.github.io/SANET/.

4.1. Experimental Settings

We trained the network using MS-COCO [15] for the content images and WikiArt [17] for the style images. Both datasets contain roughly 80,000 training images. We used the Adam optimizer [9] with a learning rate of 0.0001 and a batch size of five content–style image pairs. During training, we first rescaled the smaller dimension of both images to 512 while preserving the aspect ratio, then randomly cropped a region of size 256×256 pixels. In the testing phase, our network can handle any input size because it is fully convolutional.

4.2. Comparison with Prior Work

To evaluate the our method, we compared it with three types of arbitrary style transform methods: the iterative optimization method proposed by Gatys et al. [5], two feature transformation-based methods (WCT [13] and AdaIN [7]), and the patch-based method Avatar-Net [20].

Qualitative examples. In Fig. 11, we show examples of style transfer results synthesized by the state-of-the-art methods. Additional results are provided in the supplementary materials. Note that none of the test style images were observed during the training of our model.

The optimization-based method [5] allows arbitrary style transfer, but is likely to encounter a bad local minimum (e.g., rows 2 and 4 in Fig. 11). AdaIN [7] simply adjusts the mean and variance of the content features to synthesize the stylized image. However, its results are less appealing and often retain some of the color distribution of the content due to the trade-off between content and style (e.g., rows 1, 2, and 8 in Fig. 11). In addition, both AdaIN [7] and WCT [13] sometimes yield distorted local style patterns because of the holistic adjustment of the content features to match the second-order statistics of the style features, as shown in Fig. 11. Although Avatar-Net [20] decorates the image with the style patterns according to the semantic spatial distribution of the content image and applies a multi-scale style transfer, it frequently cannot represent the local and global style patterns at the same time due to its dependency on the patch size. Moreover, it cannot keep the content structure in most cases (column 4 in Fig. 11). In contrast, our method can parse diverse style patterns such as global color distribution, texture, and local style patterns while maintaining the structure of the content in most examples, as shown in Fig. 11.

Unlike other algorithms, our learnable SANets can flexibly parse a sufficient level of style features without maximally aligning the content and style features, regardless a large domain gap (rows 1 and 6 in Fig. 11). The proposed SANet semantically distinguishes the content structure and transfers similar style patterns onto the regions with

the same semantic meaning. Our method transfers different styles for each type of semantic content. In Fig. 11 (row 3), the sky and buildings in our stylized image are stylized using different style patterns, whereas the results of other methods have ambiguous style boundaries between the sky and buildings.

We also provide details of the results in Fig. 4. Our results exhibit multi-scale style patterns (e.g., color distribution, bush strokes, and the white and red patterns of rough textures in the style image). Avatar-Net and WCT distort the brush strokes, output blurry hair texture, and do not preserve the appearance of the face. AdaIN cannot even preserve the color distribution.

User study. We used 14 content images and 70 style images to synthesize 980 images in total. We randomly selected 30 content and style combinations for each subject and showed them the stylized images obtained by the five comparison methods side-by-side in a random order. We then asked the subject to indicate his/her favorite result for each style. We collect 2,400 votes from 80 users and show the percentage of votes for each method in Fig. 5. The result shows that the stylized results obtained by our method are preferred more often than those of other methods.

Efficiency. Table 1 shows the run time performance of the proposed method and other methods at two image scales: 256 and 512 pixels. We measured the runtime performance, including the time for style encoding. The optimization-based method [5] is impractically computationally expensive because of its iterative optimization process. In contrast, our multi-scale models (`Relu_4_1` and `Relu_5_1`) algorithms run at 59 fps and 18 fps for 256- and 512-pixel images respectively, and the single-scale (only `Relu_4_1`) algorithms runs at 83 fps and 24 fps for 256- and 512-pixel images respectively. Therefore, our method could feasibly process style transfer in real time. Our model is 7–20 times faster than the matrix computation-based methods (WCT [13] and Avatar-Net [20]).

4.3. Ablation Studies

Loss analysis. In this section, we show the influence of content-style loss and identity loss. Figure 6 (a) shows the results obtained by fixing $\lambda_{identity1}$, $\lambda_{identity2}$, and λ_s at 0, 0, and 5, respectively, while increasing λ_c from 1 to 50. Figure 6 (b) shows the results obtained by fixing λ_c and λ_s at 0 and 5, respectively, and increasing $\lambda_{identity1}$ and $\lambda_{identity2}$ from 1 to 100 and from 50 to 5,000, respectively. Without the identity loss, if we increase the weight of the content loss, the content structure is preserved, but the characteristics of the style patterns disappear, because of the trade-off between the content loss and the style loss. In contrast, increasing the weights of identity loss without content loss preserves the content structure as much as possible while maintaining style patterns. However, distortion

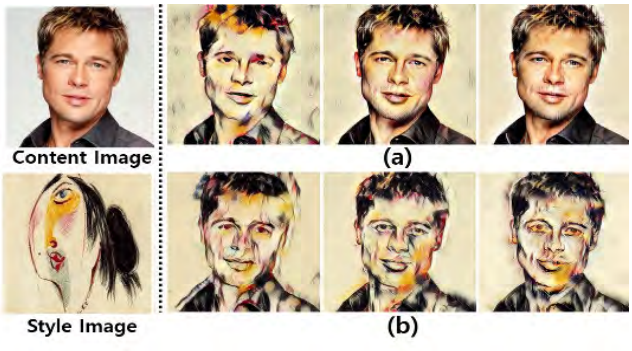

Figure 6: Content-style loss vs. identity loss. (a) Results obtained by fixing $\lambda_{identity1}$, $\lambda_{identity2}$, and λ_s at 0, 0, and 5, respectively, and increasing λ_c from 1 to 50. (b) Results obtained by fixing λ_c and λ_s at 0 and 5, respectively, and increasing $\lambda_{identity1}$ and $\lambda_{identity2}$ from 1 to 100 and from 50 to 5,000, respectively.

Figure 7: Multi-level feature embedding. By embedding features at multiple levels, we can enrich the local and global patterns for the stylized images.

of the content structure cannot be avoided. We hence applied a combination of content-style loss and identity loss to maintain the content structure while enriching style patterns.

Multi-level feature embedding. Figure 7 shows two stylized outputs obtained from `Relu_4_1` and `Relu_5_1`, respectively. When only `Relu_4_1` is used for style transfer, the global statistics of the style features and the content structure are maintained well. However, the local style patterns do not appear well. In contrast, `Relu_5_1` helps add the local style patterns such as circle patterns because the receptive field is wider. However, the content structures are distorted and textures such as brush strokes disappear. In our work, to enrich the style patterns, we integrated two SANets by taking VGG feature maps encoded from the different (`Relu_4_1` and `Relu_5_1`) layers as inputs and combining both output feature maps

4.4. Runtime Controls

In this section, we present the flexibility of our method through several applications.

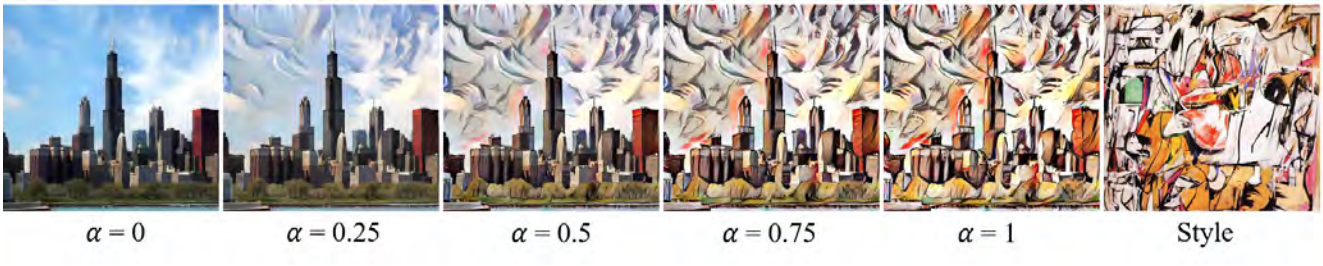

Figure 8: Content–style trade-off during runtime. Our algorithm allows this trade-off to be adjusted at runtime by interpolating between feature maps F^m_{ccc} and F^m_{csc}.

Figure 9: Style interpolation with four different styles.

Figure 10: Example of spatial control. Left: content image. Middle: style images and masks. Right: stylized image from two different style images.

Content–style trade-off. The degree of stylization can be controlled during training by adjusting the style weight λ_s in Eq. 6 or during test time by interpolating between feature maps that are fed to the decoder. For runtime control, we adjust the stylized features $F^m_{csc} \leftarrow \alpha F^m_{csc} + (1-\alpha)F^m_{ccc}$ and $\forall \alpha \in [0, 1]$. Map F^m_{ccc} is obtained by taking two content images as input for our model. The network tries to reconstruct the content image when $\alpha = 0$, and to synthesize the most stylized image when $\alpha = 1$ (as shown in Fig. 8).

Style interpolation. To interpolate between several style images, a convex combination of feature maps F^m_{csc} from different styles can be fed into the decoder (as shown in Fig. 9).

Spatial control. Figure 10 shows an example of spatially controlling the stylization. A set of masks M (Fig. 10 column 3) is additionally required as input to map the spatial correspondence between content regions and styles. We can assign the different styles in each spatial region by replacing F^m_{csc} with $M \bigodot F^m_{csc}$, where $\bigodot$ is a simple mask-out operation.

5. Conclusions

In this work, we proposed a new arbitrary style transform algorithm that consists of style-attentional networks and decoders. Our algorithm is effective and efficient. Unlike the patch-based style decorator in [20], our proposed SANet can flexibly decorate the style features through learning using a conventional style reconstruction loss and identity loss. Furthermore, the proposed identity loss helps the SANet maintain the content structure, enriching the local and global style patterns. Experimental results demonstrate that the proposed method synthesizes images that are preferred over other state-of-the-art arbitrary style transfer algorithms.

Acknowledgments. This research is supported by the Ministry of Culture, Sports, and Tourism (MCST) and Korea Creative Content Agency (KOCCA) in the Culture Technology (CT) Research and Development Program 2019

References

[1] D. Chen, L. Yuan, J. Liao, N. Yu, and G. Hua. StyleBank: An explicit representation for neural image style transfer. In *Proc. CVPR*, volume 1, page 4, 2017.

[2] T. Q. Chen and M. Schmidt. Fast patch-based style transfer of arbitrary style. *arXiv preprint arXiv:1612.04337*, 2016.

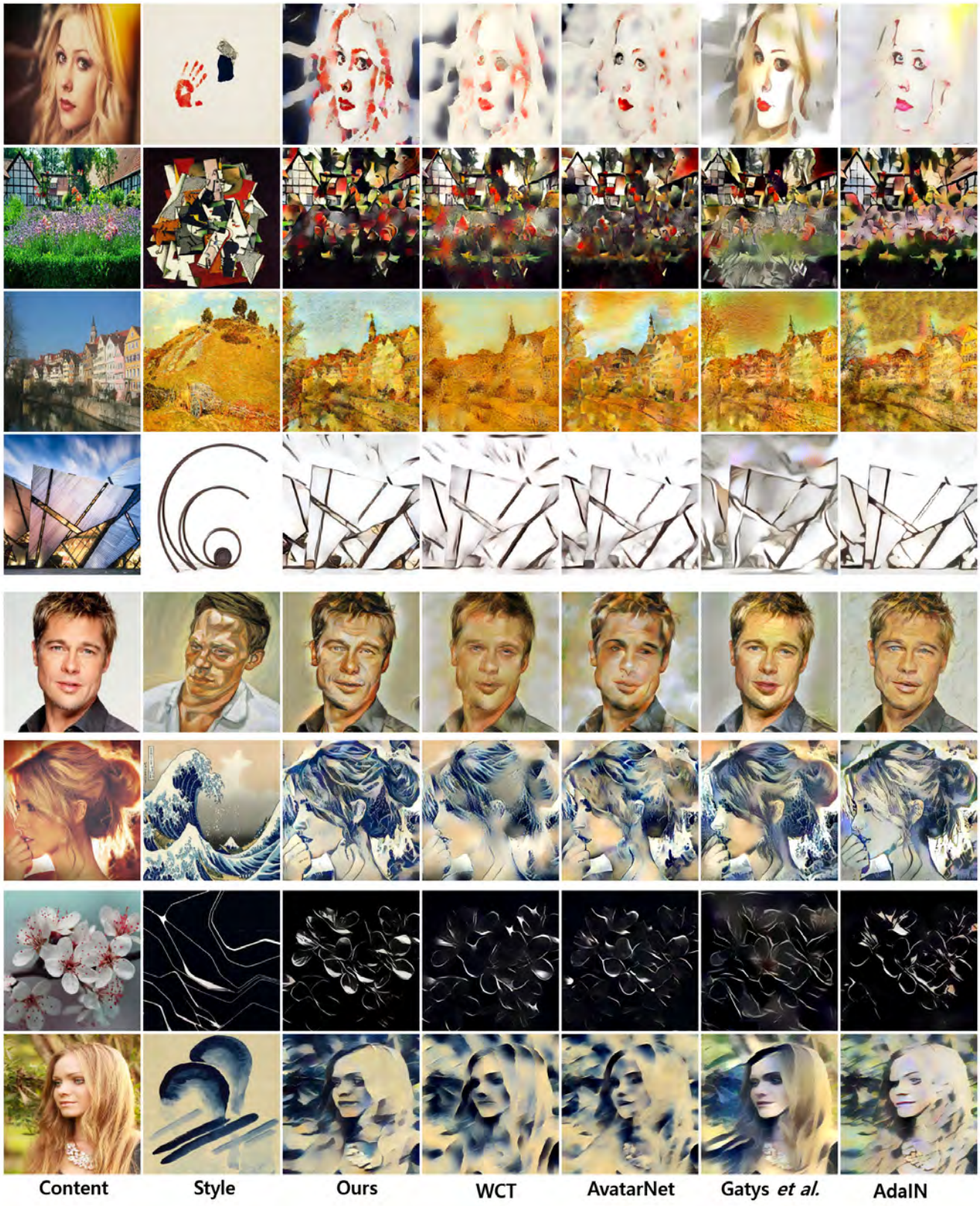

Figure 11: Example results for comparisons.

[3] V. Dumoulin, J. Shlens, and M. Kudlur. A learned representation for artistic style. In *Proc. ICLR*, 2017.

[4] L. Gatys, A. S. Ecker, and M. Bethge. Texture synthesis using convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 262–270, 2015.

[5] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *Proc. CVPR*, pages 2414–2423, 2016.

[6] L. A. Gatys, A. S. Ecker, M. Bethge, A. Hertzmann, and E. Shechtman. Controlling perceptual factors in neural style transfer. In *Proc. CVPR*, 2017.

[7] X. Huang and S. J. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proc. ICCV*, pages 1510–1519, 2017.

[8] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *Proc. ECCV*, pages 694–711. Springer, 2016.

[9] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[10] C. Li and M. Wand. Combining Markov random fields and convolutional neural networks for image synthesis. In *Proc. CVPR*, pages 2479–2486, 2016.

[11] C. Li and M. Wand. Precomputed real-time texture synthesis with Markovian generative adversarial networks. In *Proc. ECCV*, pages 702–716. Springer, 2016.

[12] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang. Diversified texture synthesis with feed-forward networks. In *Proc. CVPR*, 2017.

[13] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang. Universal style transfer via feature transforms. In *Advances in Neural Information Processing Systems*, pages 386–396, 2017.

[14] Y. Li, N. Wang, J. Liu, and X. Hou. Demystifying neural style transfer. *arXiv preprint arXiv:1701.01036*, 2017.

[15] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *Proc. ECCV*, pages 740–755. Springer, 2014.

[16] A. Paszke, S. Chintala, R. Collobert, K. Kavukcuoglu, C. Farabet, S. Bengio, I. Melvin, J. Weston, and J. Mariethoz. PyTorch: Tensors and dynamic neural networks in Python with strong GPU acceleration, Available: https://github.com/pytorch/pytorch, May 2017.

[17] F. Phillips and B. Mackintosh. Wiki Art Gallery, Inc.: A case for critical thinking. *Issues in Accounting Education*, 26(3):593–608, 2011.

[18] E. Risser, P. Wilmot, and C. Barnes. Stable and controllable neural texture synthesis and style transfer using histogram losses. *arXiv preprint arXiv:1701.08893*, 2017.

[19] F. Shen, S. Yan, and G. Zeng. Meta networks for neural style transfer. *arXiv preprint arXiv:1709.04111*, 2017.

[20] L. Sheng, Z. Lin, J. Shao, and X. Wang. Avatar-Net: Multi-scale zero-shot style transfer by feature decoration. In *Proc. CVPR*, pages 8242–8250, 2018.

[21] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[22] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. S. Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *Proc. ICML*, pages 1349–1357, 2016.

[23] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, (2016).

[24] D. Ulyanov, A. Vedaldi, and V. S. Lempitsky. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In *Proc. CVPR*, volume 1, page 3, 2017.

[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[26] H. Wang, X. Liang, H. Zhang, D.-Y. Yeung, and E. P. Xing. ZM-Net: Real-time zero-shot image manipulation network. *arXiv preprint arXiv:1703.07255*, 2017.

[27] X. Wang, R. Girshick, A. Gupta, and K. He. Non-local neural networks. *arXiv preprint arXiv:1711.07971*, 2017.

[28] X. Wang, G. Oxholm, D. Zhang, and Y.-F. Wang. Multimodal transfer: A hierarchical deep convolutional neural network for fast artistic style transfer. In *Proc. CVPR*, volume 2, page 7, 2017.

[29] H. Zhang and K. Dana. Multi-style generative network for real-time transfer. *arXiv preprint arXiv:1703.06953*, 2017.

[30] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena. Self-attention generative adversarial networks. *arXiv preprint arXiv:1805.08318*, 2018.

CVPR2019 Paper Translation

姓名：王思睿

学号：2016300155

班号：HC001606

带有风格注意网络的任意风格转换

Dae Young Park
Artificial Intelligence Research Institute, Korea
likebullet86@gmail.com

Kwang Hee Lee
Artificial Intelligence Research Institute, Korea
lkwanghee@gmail.com

摘要

任意风格转换是利用一个图像的风格合成一个图像的内容来创建具有从未见过的第三个图像。近期的任意风格转换算法研究认为平衡内容结构与风格是一件具有挑战的事情。此外，由于基于块的机制，同时保障全局和局部的风格十分困难。在本篇文章中，我们提出了一种新型的风格注意网络(SANet)，能够高效灵活的根据内容图像的语义空间分布集成局部风格模式。一种新的恒等损失函数和多层特征嵌入使我们的SANet和生成层?(decorder)能够在丰富样式模式的同时尽可能的保留内容结构。实验结果表明我们的算法实时将图片风格化的质量要好于目前最先进的算法。

1. 介绍

艺术风格转换是一种通过在保持原有结构的同时，综合一个给定风格图片的全局和局部模式并将其均匀的叠加在一个内容图像上的艺术创作技术。最近，Gatys等人开创性的工作表明，从预先训练的深度神经网络中提取的特征之间的相关性可以很好地捕捉样式模式。Gatys等人的方法足够灵活，可以将任意图像的内容和样式结合起来，但是由于迭代优化过程，速度慢得令人望而却步。

为降低这一过程的成本已做出巨大努力。有很多研究方法[1, 8, 12, 22, 3, 14, 19, 26, 29]是基于前馈网络的。前馈神经网络可以有效综合风格图片，但限于固定数量的样式或提供的视觉质量不够。

对于任意样式的转换，有一些方法[13, 7, 20]对内容特性进行整体调整以匹配样式特征的二阶统计量。AdaIN [7]简单地调整内容图像的均值和方差，以匹配那些风格图像。虽然AdaIN通过传递特征统计信息，有效地结合了内容图像的结构和样式模式，但由于该方法过于简化，其输出质量受到影响。WCT[13]通过协方差而不是方差的增白着色过程将内容特征转化为风格特征空间。通过将这些风格化特征嵌入预训练的编解码器模块中，无风格化解码器合成风格化图像。然而，如果该特性具有大量维度，那么WCT将相应地需要在计算上花费大量的操作。Avatar-Net[20]是一个基于补丁的风格装饰器模块，它在维护内容结构的同时，将内容特性映射到风格模式的特征。Avatar-Net不仅考虑整体风格分布，而且考虑局部风格模式。然而，尽管付出了大量的努力，这些方法仍然不能反映风格图像的细节纹理，扭曲内容结构，或者不能平衡局部和全局风格模式。

在这项工作中，我们提出了一种新的任意风格传输算法，它在保留内容结构的同时实时合成高质量的风格化图像。这是通过一个新的风格注意网络(SANet)和一个新的恒等误差函数函数实现的。对于任意的风格转换，由SANet和解码器组成的前馈网络根据内容特征对风格特征进行空间重组，学习内容特征和风格特征之间的语义关联。

我们提出的SANet与Avatar-Net的风格特征装饰器[20]十分相近。然而，有两个主要的区别:提出的模型1)采用一个学习的相似核而不是一个固定的2)采用软注意而不是硬注意。换句话说，我们将自我注意力机制转变为一个可学习的、基于软注意的网络，以达到转换风格的目的。我们的SANet使用可学习的相似性内核将内容特性图表示为与它的每个位置相近的风格特征的加权和。在训练过程中使用固定误差，输入相同图像对，训练模型恢复相同的结果。在推理时，将其中一个输入图像替换为样式图像，并根据样式特征尽可能多地恢复内容图像。固定误差与内容样式权重

Figure 1: 我们提出的SANet的成果. 我们可以在保留图像内容基础上将其高质量转换成多种风格.

不同，它有助于维护内容结构而不丢失风格的丰富性，因为它有助于基于风格特征恢复内容。我们工作的主要贡献如下：

•我们提出了一个新的SANet来灵活地将语义上最接近的风格特征匹配到内容特征上。

•提出了一种前馈网络的学习方法，该网络由sanet和解码器组成，利用传统的重建误差和新的固定误差进行优化。

•我们的实验表明，我们的方法是高效的(约18-24帧每秒(fps)在512像素)合成高质量的程式化图像，同时平衡全局和局部样式模式并保留内容结构。

2. 相关工作

任意风格转移.任意样式转换的最终目标是同时实现和保持泛化、质量和效率。尽管最近取得了一些进展，但是现有的方法[5, 4, 1, 8, 12, 22, 3, 6, 10, 11, 23, 24, 28, 18]在泛化、质量和效率之间存在权衡。近年来，提出了几种实现任意风格转换的方法[13, 20, 2, 7]。AdaIN算法通过传递全局特征统计量，简单地调整内容图像的均值和方差，使其与样式图像的均值和方差匹配。WCT执行一对增白和着色的特征变换部分，用于将特征嵌入到预先训练的编解码器模块中。Avatar-Net引入了基于补丁的特性修饰器，它将内容特性转换为语义上最接近的样式特性，同时最小化它们的整体特性分布之间的差异。在很多情况下，我们观察到WCT和AvatarNet的结果并不能充分体现详细的纹理或保持内容结构。我们推测，WCT和Avatar-Net由于事先训练好的通用编解码器网络从风格特征差异较大的MS-COCO数据集[15]等通用图像中学习，可能无法综合出详细的纹理风格。因此，这些方法考虑将风格特征映射到特征空间中的内容特征上，但是无法控制风格的全局统计信息或内容结构。虽然Avatar-Net可以通过基于补丁的样式装饰器获得局部风格模式，但是风格图像中的风格模式的规模取决于补丁的大小。因此，不能同时考虑全局和局部风格模式。相比之下，AdaIN很好地转换了纹理和颜色分布，但不能很好地表达局部风格模式。在这种方法中，由于内容对尺度的适应调整和风格误差的联系，在内容和样式之间存在另一种权衡。在本文中，我们尝试使用SANets和提出的固定误差解决这些问题。这样，所提出的风格转移网络可以表示全局和局部风格模式，并在保持内容结构的同时又不失风格的丰富性。

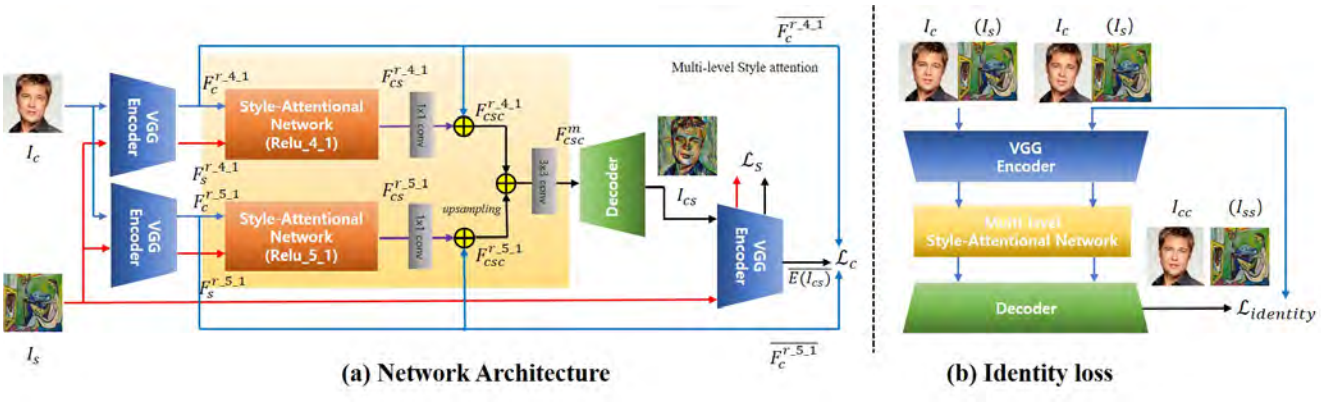

Figure 2: 训练流程概述. (a) 固定VGG编码器编码输入的内容和风格图像. 两个SANets特征图分别来自Relu_4_1和Relu_5_1的特征图. 解码器将组合的SANet输出特性转换为I_{cs} (Eq. 4). 固定的VGG编码器用于计算$\mathcal{L}_c$ (Eq. 7)和$\mathcal{L}_s$ (Eq. 8). (b) 固定损失$\mathcal{L}_{identity}$ (Eq. 9)量化I_c 和I_{cc}之间的差异或I_s 和I_{ss}之间的差异, 其中I_c (I_s) is the 原始内容(风格)图像且I_{cc} (I_{ss})是图像组合成的输出图像(内容和风格)。

注意力机制。我们的风格注意算法与最近的图像生成和机器翻译的注意力算法[25, 30]有关。这些模型通过关注序列或图像的所有位置并在嵌入空间中取其加权平均值来计算序列或图像中某一位置的响应。提出的SANet通过稍微修改自我注意机制来学习内容特征和风格特征之间的映射。

3. 模型

本文提出的风格转换网络由输入-输出模块和风格注意模块组成，如Fig. 2所示。该前馈网络能够有效地生成高质量的风格化图像，能够很好地反映全局和局部的风格模式。我们新的固定误差函数有助于维护内容的详细结构，同时充分反映风格。

3.1. 网络体系结构

我们的风格传递网络以内容图像Ic和任意风格图像Is为输入，利用前者的语义结构和后者的特征综合出一个风格化的图像Ics。采用预先训练好的VGG-19网络[21]作为编码层和对称解码层层，联合训练两个SANet进行任意风格的转换。我们的解码层模仿文章[7]中的结构。

为了充分结合全局样式模式和局部样式模式，我们通过将不同层(Relu_4_1 and ReLu_5_1)编码的VGG特征图作为输入，并结合两个输出特征图，集成了两个SANet。从一对内容图像I_c和风格图像I_s中，我们首先在输入中某一层(e.g.,ReLu_4_1)提取它们各自的VGG特征图$F_c = E(I_c)$和$F_s = E(I_s)$。

在对内容和样式图像进行编码之后，我们将这两个特性映射提供给一个SANet模块，该模块映射内容特性映射Fc和样式特性映射Fs之间的对应关系，生成输出特性映射:

$$F_{cs} = SANet\,(F_c, F_s) \tag{1}$$

对FCS应用1x1的卷积，将两个矩阵元素按如下方式求和，得到Fcsc:

$$F_{csc} = F_c + W_{cs}F_{cs}, \tag{2}$$

其中'+'代表元素对应求和。

我们将来自两个sanet的两个输出特性映射以如下形式合并:

$$F^m_{csc} = conv_{3\times3}(F^{r\_4\_1}_{csc} + upsampling(F^{r\_5\_1}_{csc})), \tag{3}$$

其中$F^{r_4_1}_{csc}$和$F^{r_5_1}_{csc}$是从两个SANet获得的输出特征映射，$conv_{3\times3}$表示将两个特征图结合所使用的3×3 卷积，$F^{r\_5\_1}_{csc}$ 向上采样后加到$F^{r_4_1}_{csc}$ 上。

然后，将F输入解码器合成风格化输出图像，如下所示:

$$I_{cs} = D\,(F^m_{csc})\,. \tag{4}$$

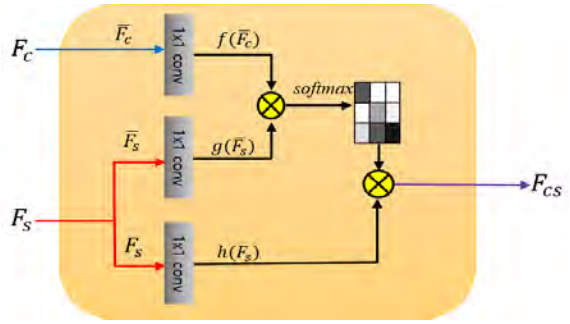

Figure 3: SANet.

3.2. 用于风格特性嵌入的SANet

图3显示了使用SANet算法嵌入的风格特性。将编码器的内容特征图F_c和风格特征图F_s进行归一化，然后转化为两个特征空间f和g，以下列方式计算$\overline{F_c^i}$与$\overline{F_s^j}$之间的注意程度:

$$F_{cs}^i = \frac{1}{C(F)} \sum_{\forall j} \exp(f(\overline{F_c^i})^T g(\overline{F_s^j})) h(F_s^j), \tag{5}$$

其中，$f(\overline{F_c}) = W_f \overline{F_c}, g(\overline{F_s}) = W_g \overline{F_s}$且$h(F_s) = W_h F_s$。此外，$\overline{F}$表示$F$在信道方向上的均值方差归一化版本。响应由一个因子$C(F) = \sum_{\forall j} \exp(f(\overline{F_c^i})^T g(\overline{F_s^j}))$进行标准化。其中，这里，$i$是输出位置的指数，$j$是枚举所有可能位置的指数。在上面的公式中，$W_f, W_g$, 和$W_h$ 是学习的权值矩阵，其实现为1 × 1个卷积，如[30]所示.

我们的SANet具有与现有的非局部块结构[27]类似的网络结构，但是输入数据的数量（由F_c和F_s组成的SANet的输入）不同。SANet算法可以通过学习映射内容和风格特征图之间的关系(例如亲缘关系)，在内容特征图的每个位置适当地嵌入一个风格样式模式。

3.3. 整体系统

如Fig. 2所示，我们使用编码器(预训练的VGG-19[21])计算训练SANet和解码器的误差函数:

$$\mathcal{L} = \lambda_c \mathcal{L}_c + \lambda_s \mathcal{L}_s + \mathcal{L}_{identity}, \tag{6}$$

其中内容、风格和固定误差分别是$\mathcal{L}_c$, $\mathcal{L}_s$, 和$\mathcal{L}_{identity}$。λ_c 和λ_s是不同误差的权重。

与[7]相似，内容误差是平均方差信道方向归一化目标特征之间，$\overline{F_c^{\text{r_4_1}}}$和$\overline{F_c^{\text{r_5_1}}}$和输出图像VGG特征的平均方差信道归一化特征，$\overline{E(I_{cs})^{\text{r_4_1}}}$ 和$\overline{E(I_{cs})^{\text{r_5_1}}}$的欧氏距离，如下：

$$\mathcal{L}_c = ||\overline{E(I_{cs})^{\text{r\_4\_1}}} - \overline{F_c^{\text{r\_4\_1}}}||_2 + ||\overline{E(I_{cs})^{\text{r\_5\_1}}} - \overline{F_c^{\text{r\_5\_1}}}||_2. \tag{7}$$

风格误差定义如下：

$$\begin{aligned} \mathcal{L}_s = &\sum_{i=1}^{L} ||\mu(\phi_i(I_{cs})) - \mu(\phi_i(I_s))||_2 \\ &+ ||\sigma(\phi_i(I_{cs})) - \sigma(\phi_i(I_s))||_2, \end{aligned} \tag{8}$$

其中任意ϕ表示用于计算风格误差的编码器中该层的特征图。我们使用Relu_1_1, Relu_2_1, Relu_3_1, Relu_4_1, 和Relu_5_1基层并赋予他们相同权重。我们同时运用Gram matrix误差[5]和AdaIN风格误差[7]，结果表明，AdaIN风格误差效果更好。

当W_f, W_g, 和W_h 是固定的单位矩阵,内容特征图中每个元素都可以转化为最近的语义特性风格特征图。在这种情况下，系统无法解析足够的风格特征。在SANet中，虽然W_f, W_g, 和W_h是可学习的矩阵，但是我们的风格转移模型只需要考虑风格误差$\mathcal{L}_s$.的全局统计量就可以训练。

为了同时考虑全局统计信息和内容特征与样式特征之间的语义局部映射，我们定义了一个新的固定误差函数如下:

$$\begin{aligned} \mathcal{L}_{identity} = &\lambda_{identity1}(||(I_{cc} - I_c)||_2 + ||(I_{ss} - I_s)||_2) \\ &+ \lambda_{identity2} \sum_{i=1}^{L} (||\phi_i(I_{cc}) - \phi_i(I_c)||_2 \\ &+ ||\phi_i(I_{ss}) - \phi_i(I_s)||_2), \end{aligned} \tag{9}$$

其中I_{cc}(or I_{ss})表示由两个相同内容(或风格)的图像合成的输出图像，任意ϕ_i 表示编码器中的一层，且$\lambda_{identity1}$和$\lambda_{identity2}$ 都为固定误差权重。在我们的实验中，权重参数简单地设置为$\lambda_c = 1$, $\lambda_s = 3$, $\lambda_{identity1} = 1$, and $\lambda_{identity2} = 50$。

内容和样式误差控制内容图像结构和样式模式之间的权衡。与其他两种损失不同的是，固定误差是从相同的输入图像中计算出来的，在样式特征上没有间隙。因此，固定误差集中于保持内容图像的结构，而不是更改风格统计信息。因此，特征损失使得同时保持参考图像的内容图像结构和风格特征成为可能。

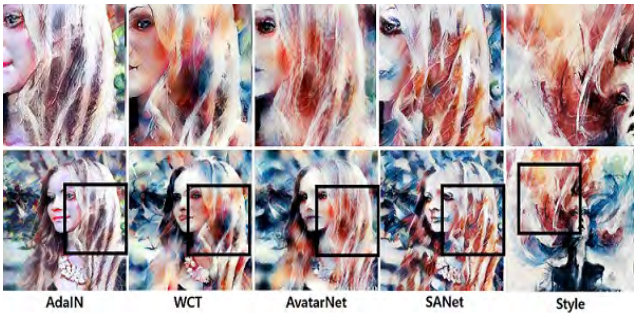

Figure 4: 成果细节. 为了更好地显示，底部行中由边框标记的区域在顶部行中被放大。

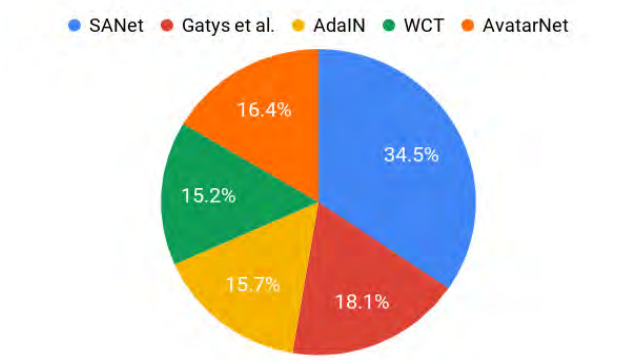

Figure 5: 五种风格传输算法的用户偏好结果。

Method	时间(256 px)	时间(512 px)
Gatys等人提出的模型[5]	15.863	50.804
WCT [13]	0.689	0.997
Avatar-Net [20]	0.248	0.356
AdaIN [7]	0.011	0.039
我们的(`Relu_4_1`)模型	0.012	0.042
我们的(多层次模型)	0.017	0.055

Table 1: 执行时间比较(以秒为单位).

4. 实验结果

Figure 2显示了基于所提议的SANet的风格转换网络的概述。演示结果将在https://dypark86.github.io/SANET/发布。

4.1. 实验设置

我们使用MS-COCO[15]训练网络的内容图像，WikiArt [17]训练网络的样式图像。这两个数据集都包含大约80,000张训练图像。我们使用Adam优化器[9]，学习率为0.0001，批处理参数大小为5对内容-样式的图像。在训练过程中，我们在保持长宽比的前提下，先将两幅图像的较小尺寸重新调整为512，然后随机裁剪出大小为256×256像素的区域。在测试阶段，我们的网络可以处理任何输入大小，因为它是完全卷积的。

4.2. 与以往工作的对比

评估我们的方法,我们比较了三种类型的任意风格变换方法:Gatys等人提出的迭代优化方法[5].,以两个特征转换为基础的方法（WCT[13]和AdaIN [7]),和基于补丁的方法Avatar-Net [20]。

定性的例子。在Fig. 11中，我们展示了用最先进的方法合成的风格转换结果的例子。补充材料中提供了其他结果。注意，在我们的模型的训练过程中没有观察到任何测试中风格的图像。

基于优化的方法[5]允许任意风格的转换，但是很可能会遇到糟糕的局部最小值(例如Fig. 11)中的第2行和第4行)。AdaIN[7]简单地调整内容特征的均值和方差来合成风格化的图像。但是，由于内容和样式之间的权衡，它的结果不太吸引人，并且常常保留内容的一些颜色分布(例如Fig. 11中的第1、2和8行)。此外，AdaIN[7]和WCT[13]有时都会产生扭曲的局部样式模式，因为它们对内容特征进行整体调整，以匹配风格特征的二阶统计量，如Fig. 11所示。虽然Avatar-Net[20]根据内容图像的语义空间分布对图像进行风格模式的装饰，并应用多尺度的风格转换，但由于依赖于patch的大小，常常不能同时表示局部和全局的样式模式。而且，在大多数情况下，它不能保持内容结构(Fig. 11中的第4列)。相比之下，我们的方法可以解析不同的风格模式，如全局颜色分布、纹理和局部样式模式，同时维护大多数示例中的内容结构，如Fig. 11所示。

与其他算法不同，我们的可学习sanet可以灵活地解析足够的水平的风格特征，而不需要最大限度地对齐内容和样式特征，不需要考虑很大的域差距(Fig. 111中的第1行和第6行)。提出的SANet在语义上区分了内容结构，并将相似的样式模式转移到具有相同语义

的区域。我们的方法为每种类型的语义内容传输不同的风格。在Fig. 11(第3行)中，我们风格化图像中的天空和建筑使用不同的风格模式进行风格化，而其他方法的结果在天空和建筑之间存在模糊的风格边界。

我们还在Fig. 4中提供了结果的详细信息。我们的结果显示了多尺度的风格模式(例如，颜色分布，灌木笔触，以及风格图像中粗糙纹理的白色和红色模式)。Avatar-Net和WCT扭曲了画笔的笔触，输出模糊的头发纹理，并没有保留面部的外观。AdaIN甚至不能保持颜色的分布。

用户调查。用14幅内容图和70幅风格图合成了980幅图像。我们为每个受试者随机选择30种内容和风格组合，并将五种对比方法得到的风格化图像并排展示给他们看。然后，我们要求受试者指出他/她对每种风格最喜欢的结果。我们从80个用户中收集2400个投票，并在Fig. 5中显示每种方法的投票百分比。结果表明，与其它方法相比，本文方法得到的风格化结果更具有优越性。

效率。Table 1给出了该方法和其他方法在256和512像素两个图像尺度下的运行时性能。我们测量了运行时性能，包括风格输入的时间。基于优化的方法[5]由于其迭代优化过程还需要大量的计算。相比之下，我们的多尺度模型算法(`Relu_4_1`和`Relu_5_1`)在256和512像素图像上分别以59帧和18帧的速度运行，而单尺度算法(只有`Relu_4_1`)在256和512像素图像上分别以83帧和24帧的速度运行。因此，我们的方法可以有效地实时处理风格转换。我们的模型比基于矩阵计算的方法(WCT[13]和avar - net[20])快7-20倍。

4.3. 模型简化测试

误差分析。在本节中，我们将展示内容风格误差和固定误差的影响。Figure 6 (a)显示了将$\lambda_{identity1}$, $\lambda_{identity2}$, and λ_s分别固定在0,0,5处，同时将λ_c从1增加到50所得到的结果。Figure 6 (b)显示了将λ_c和λ_s分别固定在0和5，将$\lambda_{identity1}$和$\lambda_{identity2}$分别从1增加到100和从50增加到5000得到的结果。在没有固定误差的情况下，如果我们增加内容误差的权重，内容结构就会保留下来，但是风格模式的特征会消失，因为内容误差和风格误差之间存在权衡。相反，在不丢失内容的情况下增加固定误差的权重，可以在维护样式模式的同时尽可能保留内容结构。然而，内容结构的扭曲是无法避免的。因此，我们在丰富风格模式的同时，应用了内容样式损失和固定误差的组合来维护内容结构。

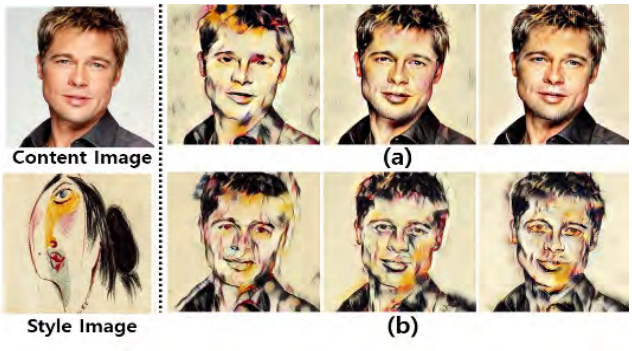

Figure 6: 内容-风格误差vs. 固定误差. (a) 将参数$\lambda_{identity1}$, $\lambda_{identity2}$, 和λ_s 固定为0, 0, and 5, 同时将λ_c从1 增加至50的结果。(b) 将参数λ_c 和λ_s 固定为0 and 5, respectively, 同时将$\lambda_{identity1}$ 和$\lambda_{identity2}$ 从1增加到100和从50增加到5,000。

Figure 7: 多层次特征潜入。通过在多个层次嵌入特征，可以丰富风格化图像的局部和全局模式。

多层次特征嵌入。Figure 7显示了分别从`Relu_4_1`和`Relu_5_1`得到的两个风格化输出。当只使用`Relu_4_1`进行风格转换时，可以很好地维护风格特征和内容结构的全局统计信息。但是，局部风格模式表达的不太好。相反，`Relu_5_1`有利于增添局部风格模式，比如圆圈模式，因为其接受域更宽。然而，其结果内容结构是扭曲的，画笔等纹理会消失。在我们的工作中，为了丰富样式模式，我们集成了两个SANet，将不同层(`Relu_4_1` 和`Relu_5_1`)的VGG特征图作为输入，并将两个输出的特征图结合起来。

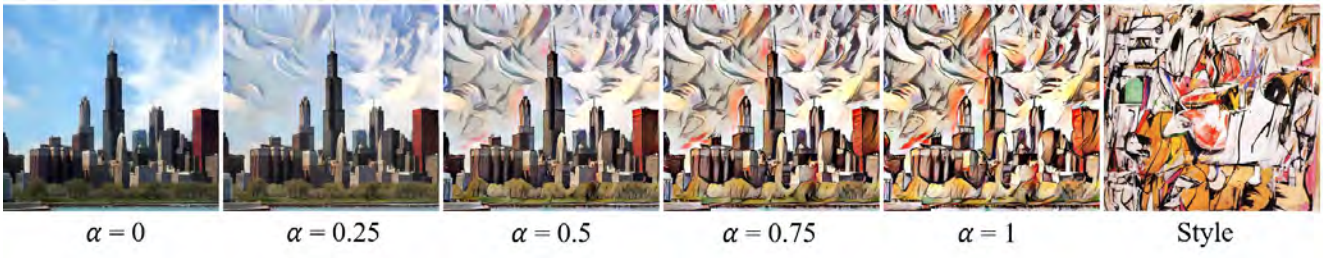

Figure 8: 运行时的内容-风格权衡。我们的算法允许在运行时通过在特征图F^m_{ccc}和F^m_{csc}间插值来调整这种平衡。

Figure 9: 风格插值与四种不同的风格。

Figure 10: 空间控制的例子。左:内容的形象。中间:样式图像和蒙版。右图:来自两个不同风格图像的风格化图像。

4.4. 运行时控制

在本节中，我们将通过几个应用程序展示方法的灵活性。

内容风格的权衡。在训练过程中，可以通过调整Eq. 6中的风格权值λ_s来控制样式化的程度，或者在测试期间，通过在输入解码器的特征映射之间插入来控制风格化的程度。在运行时控制,我们调整风格化的特征：$F^m_{csc} \leftarrow \alpha F^m_{csc} + (1-\alpha)F^m_{ccc}$和$\forall\alpha \in [0,1]$。将两幅内容图像作为模型的输入，得到映射$F^m_{ccc}$。网络试图在$\alpha = 1$时重建内容图像，在$\alpha = 1$时合成最具风格化的图像(如Fig. 8所示)。

风格插值。为了在多个风格图像之间进行插值，可以将不同风格的特征映射F^m_{csc}的凸组合输入解码器(如Fig. 9所示)。

空间控制。Figure 10 显示了一个空间控制风格化的示例。此外，还需要一组蒙版M(Fig. 10 column 3)作为输入，以映射内容区域和风格之间的空间对应关系。我们可以通过将F^m_{csc}替换为$M \bigodot F^m_{csc}$(其中。$\bigodot$为一个简单的掩蔽操作)在每个空间区域分配不同的样式。

5. 总结

在本文中，我们提出了一种新的任意风格转换算法，它由风格注意网络和解码器组成。我们的算法是有效且高效的。与[20]中基于补丁的风格装饰器不同，我们提出的SANet可以通过学习使用传统的风格重建误差和固定误差来灵活地添加风格特征。此外，提出的固定误差有助于SANet维护内容结构，丰富局部和全局样式模式。实验结果表明，该方法综合的图像优于其他最先进的任意风格转换算法。

致谢。本研究由韩国文化、体育和旅游部(MCST)和韩国创意内容机构(KOCCA)在2019年文化技术(CT)研发计划中支持。

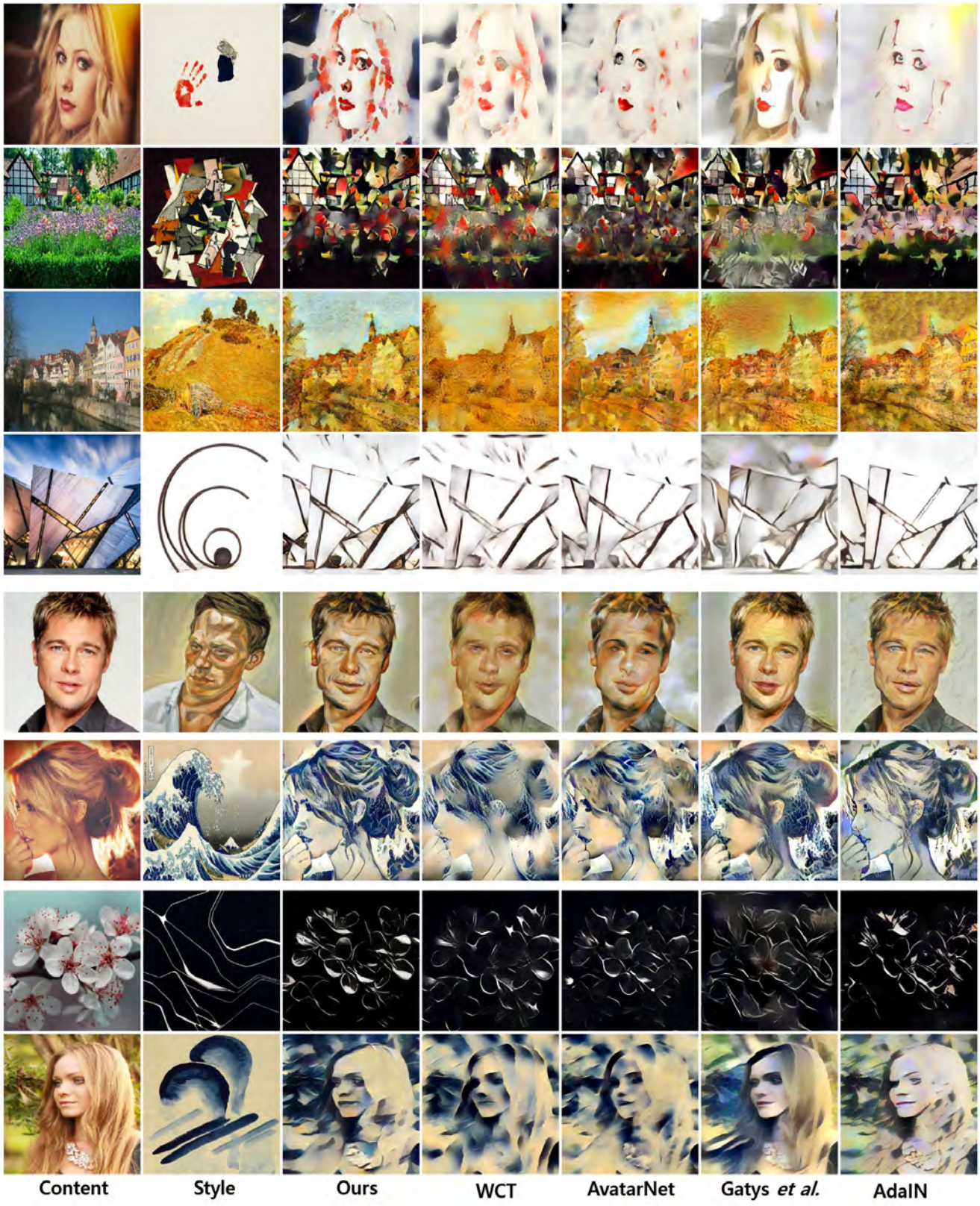

Figure 11: 用于比较的示例结果。

References

[1] D. Chen, L. Yuan, J. Liao, N. Yu, and G. Hua. StyleBank: An explicit representation for neural image style transfer. In *Proc. CVPR*, volume 1, page 4, 2017.

[2] T. Q. Chen and M. Schmidt. Fast patch-based style transfer of arbitrary style. *arXiv preprint arXiv:1612.04337*, 2016.

[3] V. Dumoulin, J. Shlens, and M. Kudlur. A learned representation for artistic style. In *Proc. ICLR*, 2017.

[4] L. Gatys, A. S. Ecker, and M. Bethge. Texture synthesis using convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 262–270, 2015.

[5] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *Proc. CVPR*, pages 2414–2423, 2016.

[6] L. A. Gatys, A. S. Ecker, M. Bethge, A. Hertzmann, and E. Shechtman. Controlling perceptual factors in neural style transfer. In *Proc. CVPR*, 2017.

[7] X. Huang and S. J. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proc. ICCV*, pages 1510–1519, 2017.

[8] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *Proc. ECCV*, pages 694–711. Springer, 2016.

[9] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[10] C. Li and M. Wand. Combining Markov random fields and convolutional neural networks for image synthesis. In *Proc. CVPR*, pages 2479–2486, 2016.

[11] C. Li and M. Wand. Precomputed real-time texture synthesis with Markovian generative adversarial networks. In *Proc. ECCV*, pages 702–716. Springer, 2016.

[12] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang. Diversified texture synthesis with feed-forward networks. In *Proc. CVPR*, 2017.

[13] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang. Universal style transfer via feature transforms. In *Advances in Neural Information Processing Systems*, pages 386–396, 2017.

[14] Y. Li, N. Wang, J. Liu, and X. Hou. Demystifying neural style transfer. *arXiv preprint arXiv:1701.01036*, 2017.

[15] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *Proc. ECCV*, pages 740–755. Springer, 2014.

[16] A. Paszke, S. Chintala, R. Collobert, K. Kavukcuoglu, C. Farabet, S. Bengio, I. Melvin, J. Weston, and J. Marietho z. PyTorch: Tensors and dynamic neural networks in Python with strong GPU acceleration, Available: https://github.com/pytorch/pytorch, May 2017.

[17] F. Phillips and B. Mackintosh. Wiki Art Gallery, Inc.: A case for critical thinking. *Issues in Accounting Education*, 26(3):593–608, 2011.

[18] E. Risser, P. Wilmot, and C. Barnes. Stable and controllable neural texture synthesis and style transfer using histogram losses. *arXiv preprint arXiv:1701.08893*, 2017.

[19] F. Shen, S. Yan, and G. Zeng. Meta networks for neural style transfer. *arXiv preprint arXiv:1709.04111*, 2017.

[20] L. Sheng, Z. Lin, J. Shao, and X. Wang. Avatar-Net: Multi-scale zero-shot style transfer by feature decoration. In *Proc. CVPR*, pages 8242–8250, 2018.

[21] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[22] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. S. Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *Proc. ICML*, pages 1349–1357, 2016.

[23] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, (2016).

[24] D. Ulyanov, A. Vedaldi, and V. S. Lempitsky. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In *Proc. CVPR*, volume 1, page 3, 2017.

[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[26] H. Wang, X. Liang, H. Zhang, D.-Y. Yeung, and E. P. Xing. ZM-Net: Real-time zero-shot image manipulation network. *arXiv preprint arXiv:1703.07255*, 2017.

[27] X. Wang, R. Girshick, A. Gupta, and K. He. Non-local neural networks. *arXiv preprint arXiv:1711.07971*, 2017.

[28] X. Wang, G. Oxholm, D. Zhang, and Y.-F. Wang. Multimodal transfer: A hierarchical deep convolutional neural network for fast artistic style transfer. In *Proc. CVPR*, volume 2, page 7, 2017.

[29] H. Zhang and K. Dana. Multi-style generative network for real-time transfer. *arXiv preprint arXiv:1703.06953*, 2017.

[30] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena. Self-attention generative adversarial networks. *arXiv preprint arXiv:1805.08318*, 2018.