StyleFormer: Real-time Arbitrary Style Transfer via Parametric Style Composition

Xiaolei Wu, Zhihao Hu, Lu Sheng, Dong Xu
1College of Software, Beihang University, China  2The University of Sydney, Australia
{wuxiaolei, huzhihao, lsheng}.buaa.edu.cn, dong.xu@sydney.edu.au

Figure 1. Results of our proposed StyleFormer, which can faithfully transfer various styles to the content images.

Abstract

In this work, we propose a new feed-forward arbitrary style transfer method, referred to as StyleFormer, which can simultaneously fulfill fine-grained style diversity and semantic content coherency. Specifically, our transformer-inspired feature-level stylization method consists of three modules: (a) the style bank generation module for sparse but compact parametric style pattern extraction, (b) the transformer-driven style composition module for content-guided global style composition, and (c) the parametric content modulation module for flexible but faithful stylization. The output stylized images are impressively coherent with the content structure, sensitive to the detailed style variations, but still holistically adhere to the style distributions from the style images. Qualitative and quantitative comparisons as well as comprehensive user studies demonstrate that our StyleFormer outperforms the existing SOTA methods in generating visually plausible stylization results with real-time efficiency.

1. Introduction

Arbitrary style transfer aims to re-render the content of one natural image by using the style of an arbitrary artwork.

* First two authors contributed equally.
† Corresponding author: Lu Sheng.
In this work, we propose a new arbitrary style transfer method that follows a similar feed-forward paradigm, but here we formulate how to generate diversified and coherent stylization results as a process, which first finds global composition of a finite set of learnable style codes and then parametrically modulates the content features by the composed style codes. The whole network can be decomposed into three modules: style bank generation, transformer-driven style composition and parametric content modulation. The style bank generation module produces a finite set of style codes as a sparse and compact representation of the style patterns. The transformer-driven style composition module adopts the expressi multi-head attention strategy from the well-known transformer architecture [32] to globally compose these representative style codes, which aims at modeling new style distributions that are coherent with the content structures and sensitive to the detailed style variations, but still holistically belong to the style manifold spanned by the style images. The parametric content modulation module aligns each content feature into its stylized counterpart by viewing the composed style code as a set of content-conditioned group-wise affine transforms, which thus offers more flexibility to represent diverse style patterns but still adheres to the content. Based on these modules, our feed-forward arbitrary style transfer method, referred to as StyleFormer, can produce visually plausible stylization results for various artworks, while ensuring the style diversity with fine-grained style details, and the content coherence with the input content images (see our results in Fig. 1).

Our network is end-to-end trained on MS-COCO [33] and Wikiart [34] based on the common style and content losses [18, 3]. When compared with the prior optimization-based [1, 3] and feed-forward based approaches [18, 17, 25, 19, 23, 35], our method achieves the SOTA stylization results in terms of both visual quality and efficiency.

2. Related Works

Optimization-based Stylization. Gatys et al. [1] first formulated the style as multi-level feature correlation (e.g., Gram matrix) from a trained neural network such as VGG [36], and defined style transfer as an iterative optimization process that balances the content and style discrepancies. Thereafter a number of variants [3, 5, 37, 38, 7, 39, 40] have been developed based on new style and content losses, or by adopting this framework to different scenarios and requirements. The works [5] and [7] explored how to control the perceptual factors during the transfer process, and another two works [37] and [38] demystified the components proposed in [1]. Different from the above works, the recent work [3] firstly viewed the style as non-parametric feature distributions instead of parametric statistics and then defined the content by using self-similarity. Despite visually plausible results are reported in these works, the careful hyper-parameter tuning process is often required [39, 40] to better optimize the objective function for each style. Moreover, the optimization-based approaches are less efficient, making these approaches unsuitable for real-time applications.

Feed-forward Approximation. Recently, a number of works [8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23] approximated the iterative back-propagation procedure as the feed-forward networks for real-time style transfer. Some works [10, 12, 41, 42, 16] specified each trained network for a single style. Other approaches [8, 13, 9, 11, 43] attempted to incorporate multiple styles in one model, but they still cannot deal with the unseen styles. The recent methods focused on arbitrary style transfer by building a more flexible feed-forward architecture. For example, the first line of works [18, 17, 22, 19] proposed to directly align the holistic statistics of the content features to that of the style features, which unfortunately leads to distortion artifacts. The second line of works [44, 45, 46, 47] exploited the power of generative adversarial network to generate the stylized images. The third line of works [24, 25, 28, 29, 30] tried to swap the most relevant style patches to match the content patches at the feature level, which may lead to flaky results. SANet [23] proposed the style-attention network to integrate the style patterns according to the semantic spatial distribution of the content image. AAMS [27] used the self-attention mechanism and swapped the styles to directly match the content structures. MANet [35] used the self-attention module to disentangle the content and style features, and then used a cross-attention operation to reorganize the style distribution according to the content distribution. Instead of directly feeding the swapped style feature patches to the decoder, we produce a set of content-aware affine transforms to group-wisely transfer each content feature into its stylized counterpart, and thus provide more flexibility in representing diverse style patterns.

Transformer. Transformer [32] is originally applied to the NLP tasks and has achieved significant improvements, which leverages the attention mechanisms to encode long-range dependencies. Recently, researchers used transformer for various CV tasks like image classification [48, 49], object detection [50, 51], video grounding [52, 53, 54] and low-level vision tasks including image translation [55], super-resolution [56] and de-noising [48]. Our proposed StyleFormer inherits the impressive relation-modeling capability of transformer to learn content-consistent style composition for the style transfer task, but we improve the naïve transformer-based style composition in a parametric way, leading to more flexible rendering results with diverse style patterns that adhere to the content semantics.
3. Approach

Following the feed-forward style transfer paradigm, we propose a new arbitrary style transfer method (referred to as StyleFormer) to learn the global composition of styles based on the well-known transformer-like architectures [32], and generate globally content-consistent and locally realistic style patterns in a parametric manner.

3.1. Overview

The framework of our proposed StyleFormer is summarized in Fig. 2. At first, the content image \( X_c \) and the style image \( X_s \) are fed into an encoder \( E_{\theta_1}(\cdot) \) to obtain the content and style features \( Z_c \in \mathbb{R}^{H_c \times W_c \times C} \) and \( Z_s \in \mathbb{R}^{H_s \times W_s \times C} \), where \( H_c \) and \( W_c \) are the height and the width of the content feature, \( H_s \) and \( W_s \) are those of the style feature and \( C \) is the number of channels. Note the sizes of \( Z_c \) and \( Z_s \) may be different in our work. Then, the \( Z_c \) is transferred to the stylized feature \( Z_{cs} \) according to \( Z_s \), based on a newly designed feature transfer module including the proposed style bank generation, transformer-driven style composition and parametric content modulation modules. In the end, the stylized feature \( Z_{cs} \) is fed into a learnable decoder \( D_{\theta_2}(\cdot) \) to generate the stylized image.

The style feature \( Z_s \) is fed into the style bank generation (elaborated in Sec. 3.2) module to produce the style codes, which include the style keys \( K \in \mathbb{R}^{N_s \times C} \) as a set of exemplar style patterns, and style values \( V \in \mathbb{R}^{N_s \times G \times J \times (C/G) \times (C/G + 1)} \) as a set of affine transformation matrices corresponding to the style keys, which can parametrically transform content features into the stylized features based on appropriate reorganization and slicing operations. \( N_s \) is the number of style codes decided based on the spatial size (see Sec. 3.5). Specifically, \( V \) contains \( N_s \) style values and each one has \( G \times J \) affine transformation matrices with the size of \( (C/G) \times (C/G + 1) \). Note we divide each style code into \( G \) groups to ensure the efficiency, and also enrich the affine transformations in each group by using \( J \) variants for producing more diverse style patterns.

Given the aforementioned two quantities as the key and the value, respectively, taking the content feature \( Z_c \) as the query, the transformer-driven style composition (in Sec. 3.3) module applies multi-head attention mechanism to generate the group-wise content-consistent affine coefficients \( \hat{V}(g) \in \mathbb{R}^{H_c \times W_c \times J \times (C/G) \times (C/G + 1)} \) and additionally the group-wise normalized content feature \( Z_{cs}^{(g)} \in \mathbb{R}^{H_c \times W_c \times (C/G) \times 1}, g \in \{1, \ldots, G\} \).

The parametric content modulation (in Sec. 3.4) module employs \( Z_{cs}^{(g)} \) to generate a guide map for each group, based on which we first sample two corresponding affine transformation matrices from all \( J \) affine transformation matrices at each spatial position and then we generate the interpolated affine transformation matrix at each spatial position. Based on the interpolated affine transformation matrices, \( Z_{cs}^{(g)} \) is eventually transformed to produce \( Z_{cs}^{(g)} \) for each group. By concatenating \( \{Z_{cs}^{(g)}\}^{G}_{g=1} \) along the channel dimension, we produce the final stylized feature \( Z_{cs} \).

3.2. Style Bank Generation

The style bank generation process aims to discover a finite set of style codes from the style features \( Z_s \), which can
parametrically regenerate the encoded style patterns by using affine transforms upon the content features.

Each style code includes a style key $K_i$ and a style value $V_i$, $i \in \{1, \ldots, N_i\}$. $K_i \in \mathbb{R}^C$ indicates the exemplar style feature and $V_i \in \mathbb{R}^{G \times J \times (C/G) \times (C/G+1)}$ integrates a rich set of affine coefficients associated to this exemplar feature, which includes $G \times J$ affine transformation matrices with the size of $(C/G) \times (C/G+1)$. Dividing the style codes into $G$ groups can improve the efficiency while using $J$ slices will bring more flexibility. Specifically, the grouping operation reduces the costs for storing and learning the affine coefficients, since an affine transformation with $C \times (C+1)$ coefficients can be approximated by grouped affine transformations with $C \times (C/G+1)$ coefficients in total. Moreover, using $J$ candidate affine transformations in each group facilitates more flexible style representations when we interpolate them with the additional guidance signals.

Therefore, the style bank generation module includes two parallel branches for: (1) extracting the style value tensor $V \in \mathbb{R}^{N_s \times G \times J \times (C/G) \times (C/G+1)}$, (2) extracting the style key tensor $K \in \mathbb{R}^{N_s \times C}$. The network architecture is shown in Fig. 4, and we will discuss it in Sec.3.5.

3.3. Transformer-driven Style Composition

Taking the content feature $Z_c$ as the query, the style key $K$ as the key, the style value $V$ as the value, we employ the multi-head attention mechanism to compose the styles conditioned on the content structures. Note that instance normalization [42] is required before feeding the key and value into the multi-head attention module and thus we can enrich the interactions between the style and content patterns, as indicated in [25]. Here the number of heads is the same as the number of group $G$ in the style bank. Note that two additional convolutions with stride 2 are performed on the instance normalized content feature in order to reduce its spatial size and thus speed up the multi-head attention process, and we will also upsample the output affine transformation matrices $\hat{V}^{(g)}$ to match the spatial resolution of $Z_c^{(g)}$ before the parametric content modulation module. As shown in Fig. 3, this multi-head attention operation indicates that different groups focus on distinctive style patterns.

Therefore, the output of the attention module in each head will be the group-wise content-consistent affine coefficients $\hat{V}^{(g)} \in \mathbb{R}^{H_c \times W_c \times J \times (C/G) \times (C/G+1)}$. Moreover, the transformer-driven style composition module also generates the group-wise normalized content features $Z_c^{(g)} \in \mathbb{R}^{H_c \times W_c \times (C/G) \times 1}$, which are the channel-wise split content feature after the instance normalization operation, as shown in Fig. 2.

3.4. Parametric Content Modulation

In this module, we would like to slice the group-wise content-consistent affine coefficients $\hat{V}^{(g)}$, and then apply the sliced affine coefficients to transform the group-wise normalized content feature $Z_c^{(g)}$ into $Z_{cs}^{(g)}$. By concatenating $\{Z_{cs}^{(g)}\}_{g=1}^G$ along the channel dimension, we eventually produce the styled feature $Z_{cs}$.

Slicing. The slicing operation is performed by first generating a guide map $W^{(g)} \in \mathbb{R}^{H_c \times W_c \times 1}$ from $Z_c^{(g)}$ based on a stack of convolutional layers. At each spatial location, we then use the corresponding value in the guide map to decide the two nearest indices and then sample two corresponding affine transformation matrices from $J$ affine transformation matrices in $\hat{V}^{(g)}$, and eventually linearly interpolate these two affine transformation matrices to generate the sliced affine transformation matrix. Since then, the group-wise sliced affine transformation matrices become a pair of $\tilde{A}^{(g)} \in \mathbb{R}^{H_c \times W_c \times (C/G) \times 1}$ and $\tilde{b}^{(g)} \in \mathbb{R}^{H_c \times W_c \times (C/G) \times 1}$. In addition to the transformer-driven style composition module, the slicing operation also makes the final affine coefficients rely more on the content structure. Moreover, other than the softmax-based interpolation, the slicing operation is more computationally efficient.

Applying. After obtaining the group-wise sliced affine coefficients, the final stylized features can be obtained as

$$Z_{cs} = \bigoplus_{g=1}^{G} Z_{cs}^{(g)} = \bigoplus_{g=1}^{G} \tilde{A}^{(g)} \otimes Z_c^{(g)} + \tilde{b}^{(g)},$$  

where $\bigoplus_{g=1}^{G}$ means channel-wise concatenation among $G$ affine-transformed group-wise features, $\otimes$ indicates the matrix multiplication operation at every spatial position. $Z_{cs}$ has the same size as that of the input content feature $Z_c$. 

Figure 3. Visualization of the attention maps from different groups. Different groups concentrate on different spatial locations of the content image, e.g., the left attention map concentrates on the bridge and the tree, the middle one concentrates on the sky and river and the right one only concentrates on the bridge’s deck.

Figure 3. Visualization of the attention maps from different groups. Different groups concentrate on different spatial locations of the content image, e.g., the left attention map concentrates on the bridge and the tree, the middle one concentrates on the sky and river and the right one only concentrates on the bridge’s deck.
where the style loss \( L_s \) matches the mean and standard deviations of the VGG-16 features between the generated output and the input style image, in the same way as used in AdaIN [18]. Following STROTSS [3], the content loss \( L_c \) matches the self-similarity patterns extracted from the VGG-16 features between the generated output and the input content image, which loosely preserves the semantics and spatial layout rather than the rigorous pixel values.

**3.5. Loss and Implementation Details**

**Loss.** In our design, the proposed network is fully differentiable, which enables back-propagation throughout the whole network except the fixed encoder. It is trained in a supervised fashion similarly as the recent feed-forward arbitrary style transfer methods [18, 10], where the content images are gathered from MS-COCO [33], and style images are from WikiArts [34]. The training losses consist of a style loss and a content loss, namely,

\[
L = \alpha L_c + \beta L_s, \tag{2}
\]

where the style loss \( L_s \) matches the mean and standard deviations of the VGG-16 features between the generated output and the input style image, in the same way as used in AdaIN [18]. Following STROTSS [3], the content loss \( L_c \) matches the self-similarity patterns extracted from the VGG-16 features between the generated output and the input content image, which loosely preserves the semantics and spatial layout rather than the rigorous pixel values.

**Implementation Details.** The number of heads and the number of group \( G \) in the style bank is set as 16. We set the number of affine coefficients \( J \) in each group as 4. The number of channel \( C = 256 \). The weighting parameters in the loss functions are set as \( \alpha = 60 \) and \( \beta = 1 \).

The network structure of our style bank generation module is shown in Fig. 4. The style values and style keys are generated in two parallel branches. In the style value generation branch, there are 2 strided convolutional layers, three convolutional layers, and a reshape operation. In the style key generation branch, there are 2 strided convolutional layers followed by a reshape operation. Each convolutional layer is written as Conv(\( C_{out}, K, S \)), where \( C_{out} \), \( K \) and \( S \) denotes the number of output channels, the kernel size and the stride parameter, respectively. Note that the output channel 17408 of the last convolution layer before generating the style values is calculated by \( G \times J \times (C/G) \times (C/G + 1) \times J \), where \( C = 256 \), \( G = 16 \) and \( J = 4 \). The number of style codes generated in this module (i.e., \( N_s = \frac{H_s}{4} \times \frac{W_s}{4} \)) is decided by the spatial resolution of the input style feature, but is downsampled due to 2 strided convolution layers.

The encoder \( Enc(\theta) \) is a pretrained and fixed VGG-16 [36] (up to ReLU3_1 layer), and the decoder \( Dec(\theta) \) mirrors the encoder with all padding layers replaced by reflection padding, which is randomly initialized at the training stage. There is no normalization layer (if not mentioned specifically) in our models as suggested by [18].

## 4. Results and Evaluation

### 4.1. Dataset And Training Details

We use the content images from MS-COCO [33] and the style images from WikiArt [34] as the training data. Each dataset contains about 80,000 samples. We set the batch size as 16 and use the Adam optimizer with a fixed learning rate of 1e-4. For each content/style image, we first resize the image to the resolution of 512 \( \times \) 512, then randomly crop a 256 \( \times \) 256 region as the training samples. Since our network is fully convolutional, it can be applied to the images of any resolution during testing. We train our model for 800,000 steps, which takes five days on the machine with a single NVIDIA Tesla V100 GPU.

### 4.2. Comparison with Prior Arts

We quantitatively and qualitatively compare our approach with a set of baseline methods including AdaIN [18], WCT [17], Avatar-Net [11], LST [19], two iterative optimization methods (Gatys [1] and STROTSS [3]) and two attention-based methods (SANet [23] and MANet [27]).

**Qualitative Evaluations.** The representative style transfer results generated by our approach and the baseline methods are provided in Fig. 5. AdaIN [18] simply adjusts the mean and variance in a channel-wise manner and thus provides a sub-optimal solution and often retains repeated texture patterns (see the 5\(^{th}\), 6\(^{th}\) and 7\(^{th}\) columns in Fig. 5). Although WCT optimally matches the second-order statistics, it may also yield distorted patterns as it cannot always recover the original style patterns. For example, the background clutters and unwanted spiral patterns appear in the 1\(^{st}\), 5\(^{th}\), 6\(^{th}\) and 7\(^{th}\) columns in Fig. 5. It is also observed that WCT has bad performance when the style pattern is simply made up of lines (see the 2\(^{nd}\) column in Fig. 5). The results of Avatar-Net [25] are flaky and vague in most cases (see the 1\(^{st}\), 3\(^{rd}\), 4\(^{th}\), 6\(^{th}\) and 7\(^{th}\) columns in Fig. 5) since it directly swaps the style patches to the corresponding content patches, which makes it hard to keep the content structures. Different from the above methods, LST [19]
Figure 5. Comparison of the stylization results. The first and second rows show the content and style images, respectively. The rest rows are the stylized results by using AdaIN [18], WCT [17], Avatar-Net [11], LST [19], Gatys [1], STROTSS [3], SANet [23], MANet [35] and our StyleFormer, respectively. Best viewed in color.

aims to transfer the lower-level style patterns (e.g., colors) while the strokes of the style are usually ignored (see the 2nd, 6th, 7th columns in Fig. 5). The optimization-based methods Gatys et al. [1] and STROTSS [3] allow arbitrary style transfer but it is hard to tune the optimal results because the weights balancing the content and style losses are sensitive, and it is likely to encounter a bad local minimum as shown in the 1st and 5th columns in Fig. 5. In addition, STROTSS [3] tends to copy the contents from the style images, which makes the results look weird (see the 5th column in the Fig. 5). It should be mentioned that the running speed of these two optimization-based methods is more than 100 times slower than our proposed feed-forward method (see Table 1). We further provide the results from two attention-based methods SANet [23] and MANet [35]. From the results, we observe that these two
Table 1. Following WCT [17], we perform quantitative comparisons between our proposed method and the baseline methods (i.e., Gatys et al. [1], STROTSS [3], AdaIN [18], WCT [17], Avatar-Net [25], LST [19], MANet [23] and SANet [35]) in terms of covariance matrix difference (the style loss $L_s$), user preference and execution time. $256 \times 256$ denotes the resolution of each test image is $256 \times 256$.

<table>
<thead>
<tr>
<th>Method</th>
<th>Style Loss ($L_s$)</th>
<th>Preference (A/B Test)</th>
<th>Time (sec) for $256 \times 256$</th>
<th>Time (sec) for $512 \times 512$</th>
<th>Time (sec) for $1024 \times 1024$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gatys et al.</td>
<td>3.20</td>
<td>9.89</td>
<td>39.017</td>
<td>40.341</td>
<td>42.319</td>
</tr>
<tr>
<td>STROTSS</td>
<td>1.22</td>
<td>9.32</td>
<td>24.69</td>
<td>112.53</td>
<td>170.201</td>
</tr>
<tr>
<td>AdaIN</td>
<td>1.77</td>
<td>5.37</td>
<td>22.83</td>
<td>32.31</td>
<td>51.53</td>
</tr>
<tr>
<td>WCT</td>
<td>3.09</td>
<td>11.02</td>
<td>32.91</td>
<td>1.83</td>
<td>33.33</td>
</tr>
<tr>
<td>Avatar-Net</td>
<td>5.91</td>
<td>4.24</td>
<td>11.69</td>
<td>32.31</td>
<td>32.31</td>
</tr>
<tr>
<td>LST</td>
<td>2.67</td>
<td>10.73</td>
<td>33.33</td>
<td>32.31</td>
<td>27.16</td>
</tr>
<tr>
<td>SANet</td>
<td>1.41</td>
<td>10.17</td>
<td>0.004</td>
<td>0.005</td>
<td>0.024</td>
</tr>
<tr>
<td>MANet</td>
<td>1.14</td>
<td>8.47</td>
<td>0.011</td>
<td>0.013</td>
<td>0.026</td>
</tr>
<tr>
<td>Ours</td>
<td>114.36</td>
<td>30.79</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Figure 6. Comparison between our proposed method and the simple transformer [32] based baseline method.

methods are more likely to generate the repeated style patterns in the background area (see the 5th, 6th and 7th columns in Fig. 5).

In contrast to these approaches, as shown in Fig. 5, our method achieves the best performance that balances the style pattern richness and content structure consistency. Instead of holistically aligning the second-order statistics or directly rearranging the style features, our StyleFormer uses the transformer-driven style composition module with the aid of the group-wise content-conditioned parametric transfer strategy, and thus our method can more flexibly represent the diverse style patterns, while still preserving the content structure, which can also be observed in Fig. 6.

Quantitative Results. We further conduct quantitative comparison in terms of the style loss, the user preference and execution efficiency.

- **Style Loss (i.e., $L_s$):** Following WCT [17], we compare different methods in terms of the co-variance matrix difference and report the results in the first row in Table 1. It is observed that our proposed method StyleFormer achieves the lowest style loss among all approaches.

- **User Preference:** Additionally, we conduct a user study to compare our methods against eight state-of-the-art style transfer methods [1, 3, 18, 17, 19, 25, 23, 35] in terms of user preference (i.e., the result from which method is the most favored by humans). We selected 100 content-style pairs for evaluation. We use the official code and the default parameters for each method. For each user, 20 content-style combinations are selected and the stylized results from all methods are displayed in a random order. Finally, we collected 1140 votes from 57 users. As shown in Table 1, our method receives 30.79% of the total votes, which is much higher than the baseline methods. Considering that it is often a challenging task to select the best one from the results of nine methods, we also conduct the multiple A/B test. As also shown in Table 1, our method wins 77%, 88%, 75%, 67%, 73%, 68%, 75% and 67% votes when compared with AdaIN [18], Avatar-Net [25], Gatys [1], LST [19], MANet [23], SANet [35], STROTSS [3] and WCT [17], respectively. These results demonstrate that our method achieves better results.

- **Efficiency:** In Table 1, we compare the running time of our method with other methods at three image resolutions: $256 \times 256$, $512 \times 512$ and $1024 \times 1024$. The optimization-based methods [1, 3] are computationally expensive since it requires hundreds of forward and backward passes to generate the final stylized results. Other works including WCT [17], AdaIN [18], Avatar-Net [25], Gatys [1], LST [19], MANet [23], SANet [35], STROTSS [3] and WCT [17], are all based on the feed-forward networks, in which WCT is relatively slow as it requires several feed-forward passes and the SVD operation has to be executed in CPU. Benefited from the down-sampling and group-wise operations, the running time of our method for each image pair with the resolution of $1024 \times 1024$ is still within 0.1s.

4.3. More Discussions

Is naïve Transformer-driven style composition enough for image stylization? In Fig. 6, we compare our StyleFormer with a simple baseline method without involving the style bank generation and parametric content modulation modules (namely, it only use the transformer-driven style composition module), where the affine coefficients are directly replaced with the style features. We observe that our StyleFormer (Fig. 6 (d)) achieves better results than this baseline method (Fig. 6 (c)) in terms of preservation of the content structures and expression of style patterns. The re-
coefficients. For simplicity, we change the notations (for Disentangled functionalities of each terms in the affine architecture for style composition, but also validate the effect- tiveness of our parametric style composition module.

Disentangled functionalities of each terms in the affine coefficients. For simplicity, we change the notations (for the operation at each position) in Eq. (1) as $Z_{cs} = AZ_c + b$, in which $A$ and $b$ denote the content-conditioned affine transformation matrix and bias vector, $Z_c$ is the normalized content feature. We also modify it as $Z_{cs} = \alpha AZ_c + b$ by controlling the weight $\alpha$ of $A$. In Fig. 8, we provide the results when different sets of parameters are used. We observe that the style background (or called holistic style patterns) tends to disappear when $b$ is removed (see whether the yellow background exists or not when setting $Z_{cs} = AZ_c$ in Fig. 8 (c) and $Z_{cs} = b$ in Fig. 8 (e)). When $A$ is replaced by an identity matrix (i.e., $Z_{cs} = Z_c + b$ in Fig. 8 (d)), the detailed strokes from the style image no longer exist in the output image. When $\alpha$ grows, the strokes of the output images around the edges of the Eiffel Tower become stronger and meanwhile more curved lines appear (see the second row in Fig. 8). These observations demonstrate that the detailed style patterns (e.g., the strokes) are mainly stored in $A$ and the holistic style pattern is mainly stored in $b$, which provides more flexibility to multi-style applications. In Fig. 7, benefiting from the disentangled functionality of these coefficients, our StyleFormer exchanges the strokes and background of the two style images, enabling more flexible control on the stylization results.

**Should affine coefficients be content-conditioned?** In Fig. 9, we provide the results of our full method and our method without using the content-conditioned affine coefficients from the slicing operation in Sec. 3.4. It is observed that our StyleFormer generates richer style patterns by comparing the two results in the 2nd row and preserves better content consistency (see the stone bench in the 1st row), which validates that the proposed content-conditioned affine coefficients can bring more flexible style representations while preserving the content structures.

**5. Conclusion**

In this work, we have proposed a visually plausible real-time style transfer method (referred to as StyleFormer), which consists of three newly proposed modules including the style bank generation module, the transformer-driven style composition module and the parametric content modulation module. With these new modules, our StyleFormer can generate the results with fine-grained style details and coherent content structures. Extensive experimental results demonstrate the effectiveness of our method and its potential to combine multiple style patterns.

**Acknowledgment.** This work was supported by the National Key Research and Development Project of China (No. 2018AAA0101900), and the National Natural Science Foundation of China (No. 61906012, No. 62132001).
References


[18] Xin Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In ICCV, pages 1501–1510, 2017. 1, 2, 5, 6, 7


[27] Yuan Yao, Jianqiang Ren, Xuan song Xie, Weidong Liu, Yong-Jin Liu, and Jun Wang. Attention-aware multi-stroke style transfer. In CVPR, pages 1467–1475, 2019. 1, 2, 5


