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

Xiaolei Wu[1*], Zhihao Hu[1*], Lu Sheng[1†], Dong Xu[2]
[1]College of Software, Beihang University, China [2]The 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. The early work from Gatys *et al*. [1] discovered that the features extracted from a well-trained deep convolutional neural network can indicate the content structures, and their statistical distributions capture the style patterns, which inspired a line of works with advanced style and content descriptions [2, 3, 4, 5, 6, 7]. Despite remarkable results have been achieved, these methods are usually formulated as a complex optimization problem, whereby the losses over a deep network must be minimized for every image pair, leading to high computational cost.

A large number of works [8, 9, 10, 11, 12, 13, 14, 15, 16] have been proposed to balance among stylization quality, generalization ability and execution efficiency. A common paradigm is to pretrain a feed-forward network with a *feature transfer* module to "universally" produce stylized results by using a single forward pass. This module should be able to simultaneously produce *diversified* style patterns redistributed from the style image and preserve *coherent* structures with the content image. The existing attempts either tried to adjust the holistic statistics of the content features with that of the style features [17, 18, 19, 20, 21, 22, 23], or non-locally swap the relevant style features in order to match the content features [24, 25, 26, 27, 28, 29, 30, 31]. While these methods are significantly faster than those optimization based works, they may not generalize well to unseen images, which inevitably degrades stylization quality or distorts content structures.

* 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 expressive 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.

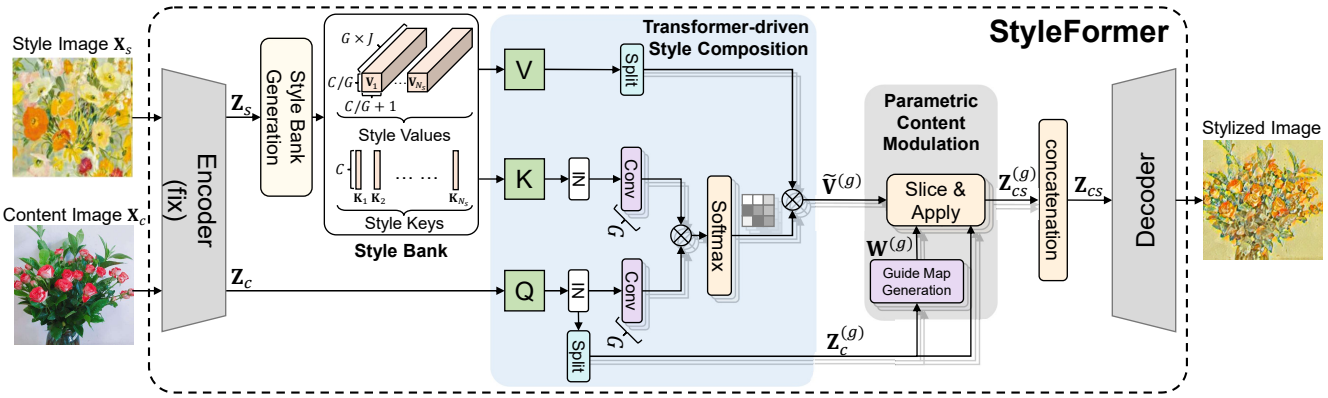


Figure 2. The overall framework of our proposed method. We first generate the content feature $\mathbf{Z}_c$ and the style feature $\mathbf{Z}_s$ from the content image $\mathbf{X}_c$ and the style image $\mathbf{X}_s$ by using the fixed encoder (VGG-16). Then we feed the style feature $\mathbf{Z}_s$ into the style bank generation module to produce the style codes. Each style code stores the affine coefficients $\mathbf{V}_i$ (*i.e.* the style value) and the corresponding style key $\mathbf{K}_i$, $i = \{1, \ldots, N_s\}$, where N_s is the number of style codes. Taking the content feature $\mathbf{Z}_c$ as the query (Q), the style keys $\mathbf{K}_i$ as the key (K), the style values $\mathbf{V}_i$ as the value (V), we employ the transformer-driven style composition module to produce the content-conditioned affine coefficients $\tilde{\mathbf{V}}^{(g)}$ for each group. The total number of group is G. After that, in the parametric content modulation module, we apply the normalized content features $\mathbf{Z}_c^{(g)}$ to predict a guide map $\mathbf{W}^{(g)}$ to slice the affine coefficient from $\tilde{\mathbf{V}}^{(g)}$, and then generate the final group-wise stylized feature $\mathbf{Z}_{cs}^{(g)}$. Finally, we concatenate $\mathbf{Z}_{cs}^{(g)}$ from all groups as the final stylized feature $\mathbf{Z}_{cs}$ and feed it into the decoder to produce the stylized image. "IN" denotes the instance normalization operation. Best viewed in color.

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 $\mathbf{X}_c$ and the style image $\mathbf{X}_s$ are fed into an encoder $E_{\boldsymbol{\theta}_{\text{enc}}}(\cdot)$ to obtain the content and style features $\mathbf{Z}_c \in \mathbb{R}^{H_c \times W_c \times C}$ and $\mathbf{Z}_s \in \mathbb{R}^{H_s \times W_s \times C}$, 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 $\mathbf{Z}_c$ and $\mathbf{Z}_s$ may be different in our work. Then the $\mathbf{Z}_c$ is transferred to the stylized feature $\mathbf{Z}_{cs}$ according to $\mathbf{Z}_s$, 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 $\mathbf{Z}_{cs}$ is fed into a learnable decoder $D_{\boldsymbol{\theta}_{\text{dec}}}(\cdot)$ to generate the stylized image.

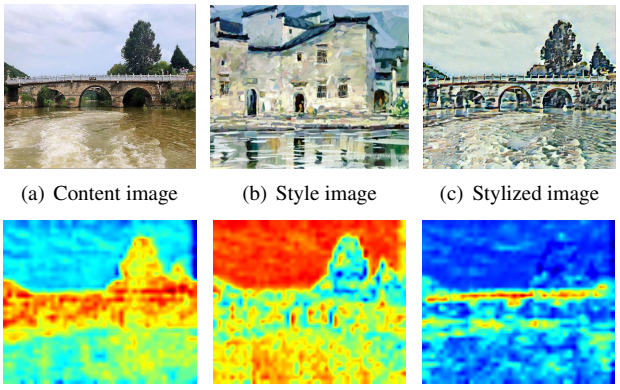
The style feature $\mathbf{Z}_s$ is fed into the **style bank generation** (elaborated in Sec. 3.2) module to produce the style codes, which include the style keys $\mathbf{K} \in \mathbb{R}^{N_s \times C}$ as a set of exemplar style patterns, and style values $\mathbf{V} \in \mathbb{R}^{N_s \times G \times J \times (C/G) \times (C/G+1)}$ 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, $\mathbf{V}$ 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 $\mathbf{Z}_c$ 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 $\tilde{\mathbf{V}}^{(g)} \in \mathbb{R}^{H_c \times W_c \times J \times (C/G) \times (C/G+1)}$, and additionally the group-wise normalized content feature $\mathbf{Z}_c^{(g)} \in \mathbb{R}^{H_c \times W_c \times (C/G) \times 1}, g \in \{1, \ldots, G\}$.

The **parametric content modulation** (in Sec. 3.4) module employs $\mathbf{Z}_c^{(g)}$ 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, $\mathbf{Z}_c^{(g)}$ is eventually transformed to produce $\mathbf{Z}_{cs}^{(g)}$ for each group. By concatenating $\{\mathbf{Z}_{cs}^{(g)}\}_{g=1}^{G}$ along the channel dimension, we produce the final stylized feature $\mathbf{Z}_{cs}$.

3.2. Style Bank Generation

The style bank generation process aims to discover a finite set of style codes from the style features $\mathbf{Z}_s$, which can



(a) Content image (b) Style image (c) Stylized image

(d) Three visualization results of the attention maps

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.

parametrically regenerate the encoded style patterns by using affine transforms upon the content features.

Each style code includes a style key $\mathbf{K}_i$ and a style value $\mathbf{V}_i, i \in \{1, \dots, N_s\}$. $\mathbf{K}_i \in \mathbb{R}^C$ indicates the exemplar style feature and $\mathbf{V}_i \in \mathbb{R}^{G \times J \times (C/G) \times (C/G+1)}$ 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 $\mathbf{V} \in \mathbb{R}^{N_s \times G \times J \times (C/G) \times (C/G+1)}$, (2) extracting the style key tensor $\mathbf{K} \in \mathbb{R}^{N_s \times C}$. The network architecture is shown in Fig. 4, and we will discussed it in Sec.3.5.

3.3. Transformer-driven Style Composition

Taking the content feature $\mathbf{Z}_c$ as the query, the style key $\mathbf{K}$ as the key, the style value $\mathbf{V}$ 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 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 $\tilde{\mathbf{V}}^{(g)}$ to match the spatial resolution of $\mathbf{Z}_c^{(g)}$ 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 $\tilde{\mathbf{V}}^{(g)} \in \mathbb{R}^{H_c \times W_c \times J \times (C/G) \times (C/G+1)}$. Moreover, the transformer-driven style composition module also generates the group-wise normalized content features $\mathbf{Z}_c^{(g)} \in \mathbb{R}^{H_c \times W_c \times (C/G) \times 1}, g \in \{1, \dots, G\}$, which are the channel-wise splitted 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 $\tilde{\mathbf{V}}^{(g)}$, and then *apply* the sliced affine coefficients to transform the group-wise normalized content feature $\mathbf{Z}_c^{(g)}$ into $\mathbf{Z}_{cs}^{(g)}$. By concatenating $\{\mathbf{Z}_{cs}^{(g)}\}_{g=1}^{G}$ along the channel dimension , we eventually produce the stylized feature $\mathbf{Z}_{cs}$.

Slicing. The slicing operation is performed by first generating a guide map $\mathbf{W}^{(g)} \in \mathbb{R}^{H_c \times W_c \times 1}$ from $\mathbf{Z}_c^{(g)}$ 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 $\tilde{\mathbf{V}}^{(g)}$, 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 $\tilde{\mathbf{A}}^{(g)} \in \mathbb{R}^{H_c \times W_c \times (C/G) \times (C/G)}$ and $\tilde{\mathbf{b}}^{(g)} \in \mathbb{R}^{H_c \times W_c \times (C/G) \times 1}$. 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 obtain as

$$\mathbf{Z}_{cs} = \mathop{\Big\|}_{g=1}^{G} \mathbf{Z}_{cs}^{(g)} = \mathop{\Big\|}_{g=1}^{G} \tilde{\mathbf{A}}^{(g)} \otimes \mathbf{Z}_c^{(g)} + \tilde{\mathbf{b}}^{(g)}, \qquad (1)$$

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

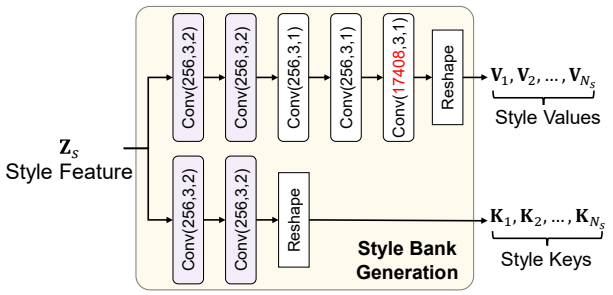


Figure 4. The network details of our style bank generation module. $\mathbf{Z}_s$ denotes the style feature. $\mathbf{V}_i$ and $\mathbf{K}_i, i = 1, \ldots, N_s$ are the output style values and keys, respectively.

Note that this module employs the instance-normalized content features as its input so that it would like to pay more attention to the structural clues and remove the textural patterns in the content image, which again leads to more reliable stylization results.

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,

$$\mathcal{L} = \alpha \mathcal{L}_c + \beta \mathcal{L}_s, \tag{2}$$

where the style loss $\mathcal{L}_s$ 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 $\mathcal{L}_c$ 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 $E_{\boldsymbol{\theta}_{\text{enc}}}(\cdot)$ is a pretrained and fixed VGG-16 [36] (up to `ReLU3_1` layer), and the decoder $D_{\boldsymbol{\theta}_{\text{dec}}}(\cdot)$ 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 512×512, then randomly crop a 256×256 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 2^{nd}, 6^{th}, 7^{th} 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 1^{st} and 5^{th} 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 5^{th} 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. 256×256 denotes the resolution of each test image is 256×256.

	Gatys *et al*.	STROTSS	AdaIN	WCT	Avatar-Net	LST	SANet	MANet	Ours
Style Loss (*i.e.*, L_s)	3.20	1.22	1.77	3.09	5.91	2.67	1.41	1.83	**1.14**
Preference(%)	9.89	9.32	5.37	11.02	4.24	10.73	10.17	8.47	**30.79**
A/B Test(%)	25.00	24.69	22.83	32.91	11.69	33.33	32.31	27.16	-
Time (sec) for 256×256	39.017	92.724	0.004	0.531	0.884	0.004	0.011	0.009	0.013
Time (sec) for 512×512	40.341	112.53	0.005	1.298	0.971	0.005	0.038	0.011	0.026
Time (sec) for 1024×1024	42.319	170.201	0.007	4.077	1.174	0.024	0.169	0.014	0.071

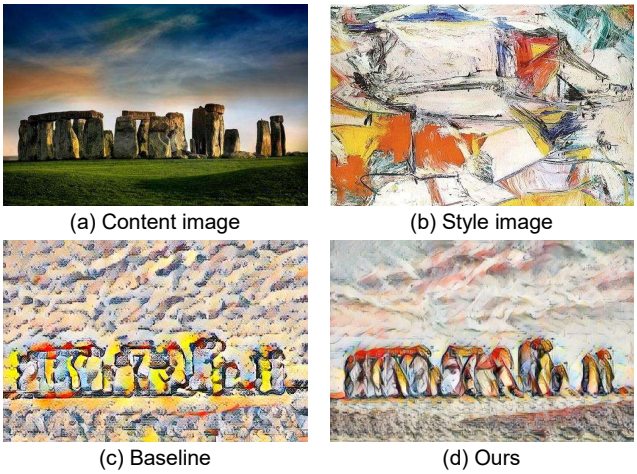


Figure 6. Comparison between our proposed method and the simple transformer [32] based baseline method.

methods are more likely to generate the repeated style patterns in the background area (see the 5^{th}, 6^{th} and 7^{th} 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: 256×256, 512×512 and 1024×1024. 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 [11] and LST [19] 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 1024×1024 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-

Figure 7. The exchange of the strokes and background of two style images. $\mathbf{Z}_{cs}$ denotes the stylized feature, $\mathbf{A}$ and $\mathbf{b}$ denote the affine transformation matrix and bias from style 1 or style 2 and $\mathbf{Z}_c$ denotes the content feature.

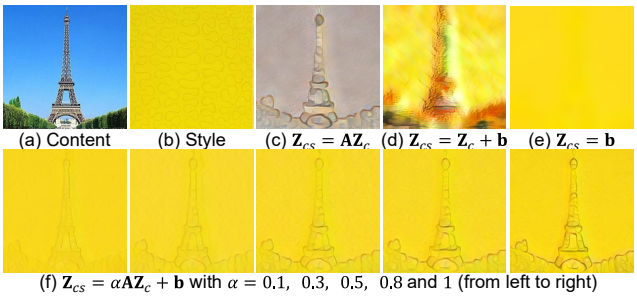


Figure 8. Ablation study of the affine transformation. We change the notations in Eq. (1) as $\mathbf{Z}_{cs} = \mathbf{A}\mathbf{Z}_c + \mathbf{b}$, in which $\mathbf{Z}_c$ is the content feature, $\mathbf{A}$ and $\mathbf{b}$ are the affine transformation matrix and the bias vector, and $\mathbf{Z}_{cs}$ is the stylized feature. The results in (c, d, e, f) are the decoded stylized images.

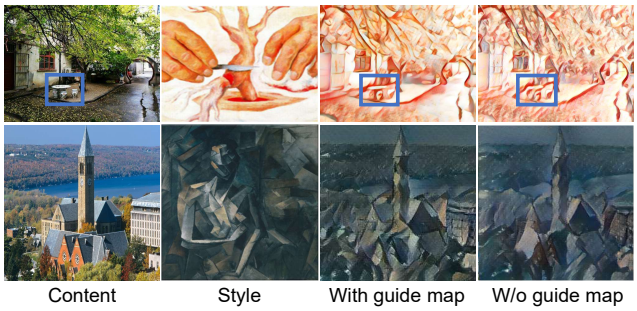


Figure 9. Ablation study of the guide map. It is observed that our method with the guide map preserves better content consistency (see the blue box in the first row) and also generates richer style patterns (see the second row).

sults not only indicate that we may not solve the image stylization problem by simply applying the transformer-like architecture for style composition, but also validate the effectiveness 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 $\mathbf{Z}_{cs} = \mathbf{A}\mathbf{Z}_c + \mathbf{b}$, in which $\mathbf{A}$ and $\mathbf{b}$ denote the content-conditioned affine transformation matrix and bias vector, $\mathbf{Z}_c$ is the normalized content feature. We also modify it as $\mathbf{Z}_{cs} = \alpha\mathbf{A}\mathbf{Z}_c + \mathbf{b}$ by controlling the weight α of $\mathbf{A}$. 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 $\mathbf{b}$ is removed (see whether the yellow background exists or not when setting $\mathbf{Z}_{cs} = \mathbf{A}\mathbf{Z}_c$ in Fig. 8 (c) and $\mathbf{Z}_{cs} = \mathbf{b}$ in Fig. 8 (e)). When $\mathbf{A}$ is replaced by an identity matrix (*i.e.*, $\mathbf{Z}_{cs} = \mathbf{Z}_c + \mathbf{b}$ in Fig. 8 (d)), the detailed strokes from the style image no longer exist in the output image. When α 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 $\mathbf{A}$ and the holistic style pattern is mainly stored in $\mathbf{b}$, which provides more flexibility to multi-style applications. In Fig. 7, benefiting from the disentangled functionality of these coefficents, 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 modoule 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

[1] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *CVPR*, pages 2414–2423, 2016. 1, 2, 5, 6, 7

[2] Chuan Li and Michael Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *ECCV*, pages 702–716. Springer, 2016. 1

[3] Nicholas Kolkin, Jason Salavon, and Gregory Shakhnarovich. Style transfer by relaxed optimal transport and self-similarity. In *CVPR*, pages 10051–10060, 2019. 1, 2, 5, 6, 7

[4] Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. Singan: Learning a generative model from a single natural image. In *ICCV*, pages 4570–4580, 2019. 1

[5] Leon A Gatys, Alexander S Ecker, Matthias Bethge, Aaron Hertzmann, and Eli Shechtman. Controlling perceptual factors in neural style transfer. In *CVPR*, pages 3985–3993, 2017. 1, 2

[6] Mao-Chuang Yeh, Shuai Tang, Anand Bhattad, Chuhang Zou, and David Forsyth. Improving style transfer with calibrated metrics. In *WACV*, pages 3160–3168, 2020. 1

[7] Alex J Champandard. Semantic style transfer and turning two-bit doodles into fine artworks. *arXiv preprint arXiv:1603.01768*, 2016. 1, 2

[8] Dongdong Chen, Lu Yuan, Jing Liao, Nenghai Yu, and Gang Hua. Stylebank: An explicit representation for neural image style transfer. In *CVPR*, pages 1897–1906, 2017. 1, 2

[9] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. In *ICLR*, 2017. 1, 2

[10] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, pages 694–711. Springer, 2016. 1, 2, 5

[11] Falong Shen, Shuicheng Yan, and Gang Zeng. Neural style transfer via meta networks. In *CVPR*, pages 8061–8069, 2018. 1, 2, 5, 6, 7

[12] Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor S Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *ICML*, volume 1, page 4, 2016. 1, 2

[13] Hang Zhang and Kristin Dana. Multi-style generative network for real-time transfer. In *ECCVW*, 2018. 1, 2

[14] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In *CVPR*, pages 6924–6932, 2017. 1, 2

[15] Yongcheng Jing, Yang Liu, Yezhou Yang, Zunlei Feng, Yizhou Yu, Dacheng Tao, and Mingli Song. Stroke controllable fast style transfer with adaptive receptive fields. In *ECCV*, pages 238–254, 2018. 1, 2

[16] Gilles Puy and Patrick Pérez. A flexible convolutional solver for fast style transfers. In *CVPR*, pages 8963–8972, 2019. 1, 2

[17] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. Universal style transfer via feature transforms. In *NIPS*, pages 386–396, 2017. 1, 2, 5, 6, 7

[18] Xun 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

[19] Xueting Li, Sifei Liu, Jan Kautz, and Ming-Hsuan Yang. Learning linear transformations for fast image and video style transfer. In *CVPR*, pages 3809–3817, 2019. 1, 2, 5, 6, 7

[20] Hao Wang, Xiaodan Liang, Hao Zhang, Dit-Yan Yeung, and Eric P Xing. Zm-net: Real-time zero-shot image manipulation network. *arXiv preprint arXiv:1703.07255*, 2017. 1, 2

[21] Chunjin Song, Zhijie Wu, Yang Zhou, Minglun Gong, and Hui Huang. Etnet: Error transition network for arbitrary style transfer. In *NIPS*, pages 668–677, 2019. 1, 2

[22] Ming Lu, Hao Zhao, Anbang Yao, Yurong Chen, Feng Xu, and Li Zhang. A closed-form solution to universal style transfer. In *ICCV*, pages 5952–5961, 2019. 1, 2

[23] Dae Young Park and Kwang Hee Lee. Arbitrary style transfer with style-attentional networks. In *CVPR*, pages 5880–5888, 2019. 1, 2, 5, 6, 7

[24] Tian Qi Chen and Mark Schmidt. Fast patch-based style transfer of arbitrary style. *arXiv preprint arXiv:1612.04337*, 2016. 1, 2

[25] Lu Sheng, Ziyi Lin, Jing Shao, and Xiaogang Wang. Avatar-net: Multi-scale zero-shot style transfer by feature decoration. In *CVPR*, pages 8242–8250, 2018. 1, 2, 4, 5, 7

[26] Shuyang Gu, Congliang Chen, Jing Liao, and Lu Yuan. Arbitrary style transfer with deep feature reshuffle. In *CVPR*, pages 8222–8231, 2018. 1

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

[28] Yulun Zhang, Chen Fang, Yilin Wang, Zhaowen Wang, Zhe Lin, Yun Fu, and Jimei Yang. Multimodal style transfer via graph cuts. In *ICCV*, pages 5943–5951, 2019. 1, 2

[29] Jing Liao, Yuan Yao, Lu Yuan, Gang Hua, and Sing Bing Kang. Visual attribute transfer through deep image analogy. *ACM Transactions on Graphics (TOG)*, 36(4):1–15, 2017. 1, 2

[30] Roey Mechrez, Itamar Talmi, and Lihi Zelnik-Manor. The contextual loss for image transformation with non-aligned data. In *ECCV*, pages 768–783, 2018. 1, 2

[31] Zhizhong Wang, Lei Zhao, Sihuan Lin, Qihang Mo, Huiming Zhang, Wei Xing, and Dongming Lu. Glstylenet: exquisite style transfer combining global and local pyramid features. *IET Computer Vision*, 14(8):575–586, 2020. 1

[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, pages 5998–6008, 2017. 2, 3, 7

[33] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, pages 740–755. Springer, 2014. 2, 5

[34] K. Nichol. Painter by numbers. *https://www.kaggle.com/c/painter-by-numbers*, 2016. 2, 5

[35] Yingying Deng, Fan Tang, Weiming Dong, Wen Sun, Feiyue Huang, and Changsheng Xu. Arbitrary style transfer via multi-adaptation network. In *ACM Multimedia*, pages 2719–2727, 2020. 2, 6, 7

[36] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 2, 5

[37] Len Du. How much deep learning does neural style transfer really need? an ablation study. In *WACV*, pages 3150–3159, 2020. 2

[38] Yanghao Li, Naiyan Wang, Jiaying Liu, and Xiaodi Hou. Demystifying neural style transfer. In *IJCAI*, pages 2230–2236, 2017. 2

[39] Manuel Ruder, Alexey Dosovitskiy, and Thomas Brox. Artistic style transfer for videos. In *German conference on pattern recognition*, pages 26–36. Springer, 2016. 2

[40] Eric Risser, Pierre Wilmot, and Connelly Barnes. Stable and controllable neural texture synthesis and style transfer using histogram losses. *arXiv preprint arXiv:1701.08893*, 2017. 2

[41] Wei Gao, Yijun Li, Yihang Yin, and Ming-Hsuan Yang. Fast video multi-style transfer. In *WACV*, pages 3222–3230, 2020. 2

[42] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016. 2, 4

[43] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. Diversified texture synthesis with feed-forward networks. In *CVPR*, pages 3920–3928, 2017. 2

[44] Jan Svoboda, Asha Anoosheh, Christian Osendorfer, and Jonathan Masci. Two-stage peer-regularized feature recombination for arbitrary image style transfer. In *CVPR*, pages 13816–13825, 2020. 2

[45] Dmytro Kotovenko, Artsiom Sanakoyeu, Pingchuan Ma, Sabine Lang, and Bjorn Ommer. A content transformation block for image style transfer. In *CVPR*, pages 10032–10041, 2019. 2

[46] Dmytro Kotovenko, Artsiom Sanakoyeu, Sabine Lang, and Bjorn Ommer. Content and style disentanglement for artistic style transfer. In *ICCV*, pages 4422–4431, 2019. 2

[47] Artsiom Sanakoyeu, Dmytro Kotovenko, Sabine Lang, and Bjorn Ommer. A style-aware content loss for real-time hd style transfer. In *ECCV*, pages 698–714, 2018. 2

[48] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *ICML*, pages 1691–1703. PMLR, 2020. 2

[49] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv*, 2020. 2

[50] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, pages 213–229. Springer, 2020. 2

[51] Lichen Zhao, Jinyang Guo, Dong Xu, and Lu Sheng. Transformer3D-Det: Improving 3D object detection by vote refinement. *IEEE T-CSVT*, 2021. 2

[52] Zongheng Tang, Yue Liao, Si Liu, Guanbin Li, Xiaojie Jin, Hongxu Jiang, Qian Yu, and Dong Xu. Human-centric spatio-temporal video grounding with visual transformers. *IEEE T-CSVT*, 2021. 2

[53] Rui Su, Qian Yu, and Dong Xu. STVGBert: A visual-linguistic transformer based framework for spatio-temporal video grounding. In *ICCV*, pages 4343–4351, 2021. 2

[54] Lichen Zhao, Daigang Cai, Lu Sheng, and Dong Xu. 3DVG-Transformer: Relation modeling for visual grounding on point clouds. In *ICCV*, pages 4883–4891, 2021. 2

[55] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *ICML*, pages 4055–4064. PMLR, 2018. 2

[56] Fuzhi Yang, Huan Yang, Jianlong Fu, Hongtao Lu, and Baining Guo. Learning texture transformer network for image super-resolution. In *CVPR*, pages 5791–5800, 2020. 2