StyleRemix: An Interpretable Representation for Neural Image Style Transfer

Hongmin Xu*, Qiang Li*, Wenbo Zhang, Wen Zheng
Y-tech, Kwai

Abstract

Multi-Style Transfer (MST) intents to capture the high-level visual vocabulary of different styles and expresses these vocabularies in a joint model to transfer each specific style. Recently, Style Embedding Learning (SEL) based methods represent each style with an explicit set of parameters to perform MST task. However, most existing SEL methods either learn explicit style representation with numerous independent parameters or learn a relatively black-box style representation, which makes them difficult to control the stylized results. In this paper, we outline a novel MST model, StyleRemix, to compactly and explicitly integrate multiple styles into one network. By decomposing diverse styles into the same basis, StyleRemix represents a specific style in a continuous vector space with 1-dimensional coefficients. With the interpretable style representation, StyleRemix not only enables the style visualization task but also allows several ways of remixing styles in the smooth style embedding space. Extensive experiments demonstrate the effectiveness of StyleRemix on various MST tasks compared to state-of-the-art SEL approaches.

1. Introduction

Style transfer aims to convincingly confer artist’s painting style, such as the shapes, lines, colors, tones, and textures created by the unique techniques of artist, to arbitrary images. With style transfer, people can produce Van Gogh’s The Starry Night style in their photographs. In the past, it may take much time of a well-trained artist to create such paintings. The critical issue of style transfer and its previous closely related work texture synthesis is how to model visual texture of target style image and generate a stylized image, which has similar visual fashion with the target style image while retaining the shape of input content image. Traditional methods [4, 39] usually focused on modeling texture with low-level features to handle this problem.

The seminal work of Gatys [5, 6], Neural Style (NS) transfer, first modeled texture and style by the summary statistics in the domain of Convolutional Neural Network (CNN). Specifically, style is defined by the covariance matrix of feature response computed by multi-layers of VGG-19 [34]. Based on online powerful iterative optimization, NS outputs a stylized image, whose style is close to the target image while retaining the shape of input image. Although NS produces better visually satisfied image than traditional methods, NS is time-consuming since its expensive computation cost. Therefore, Fast Neural Style (FNS) [14, 36, 38, 35] transfer approaches trained offline feed-forward style-specific model to produce stylized images, which is hundreds of times faster than the NS method. However, FNS loses the flexibility of NS: different generative networks have to be trained for each specific style image.

In order to improve the flexibility of FNS, Multi-Style Transfer (MST) methods aim to incorporate multiple styles into one single model. There are two paths to achieve this goal: 1) parametric Arbitrary Style Transfer (AST) approaches [41, 7, 32, 33, 20, 12, 37, 2, 9] and 2) non-parametric Style Embedding Learning (SEL) based style transfer approaches, such as [1, 3, 21, 40].
AST methods [41, 33, 20, 12, 37, 2, 9] often learn a parameter network to describe a large set of styles and manually design [2, 9, 20, 41, 33] or learn a network as a universal style transfer function [37, 12, 33]. Therefore, the scalability of AST is larger than SEL to some extent. However, AST suffers from two shortcomings: 1) They always produce compromised visual quality stylized images, as some new styles may be poorly represented by the parametric style network or the designed/learned style transfer function may not be suitable for the new styles. 2) The style representation learned by a black-box network is relatively hard to explain and difficult to control the stylization.

In parallel, SEL methods [1, 3, 21, 40] focused on learning a small set of parameters to explicitly define each style, which enable them to generate better visually impressive stylized results than AST. Dumoulin et al. [3] utilized the parameters of instance normalization layers to represent one specific style. Li et al. [21] defined a noise map to indicate one particular style. Yanai et al. [40] used a simple one-hot map to denote different styles. However, those style representations are relatively hard to interpret. As for Chen et al. [1], they proposed an explicit style representation method, StyleBank, which is more explainable whereas StyleBank need a layer of convolution filter banks to represent each style. Therefore, for all the state of the art SEL methods, either they [3, 21, 40] are hardly interpretable or [1, 40] inefficient to represent styles.

In this paper, we propose a SEL method StyleRemix that brings interpretability and efficiency to the style representation in MST task. Specifically, StyleRemix separates content (of input image) and style (of target image) representation. Content is transformed into multi-layers activations of an autoencoder. Style is described as weighted convolution filters, which is convolved with the intermediate feature embedding (content) to get the final stylized image. Style representation consists of two parts: style basis and style weights. Style basis correspond to interpretability and their channels describe the fundamental style elements. Style weights correspond to efficiency and a specific style is modeled as a 1-dimensional continuous vector, which is more efficient than the block-wise representation method in [1].

Our method learns the style embedding space for a set of styles. Thus, MST task can be performed by providing the inference network with the style weights of a specific style. With the learned style representations, StyleRemix makes it possible for the task of style visualization. Besides, thanks to the interpretability and efficiency of the style representation, our method offers several ways to manipulate styles, such as convex combination, noise perturbation, and even Collection Style Transfer (CST), as shown in Fig. 1. Furthermore, StyleRemix can handle a certain amount of diverse styles with consistent visual quality according to a scalability analysis on the number of styles.

2. Related Works

Paired Image-to-Image Translation. Image-to-Image Translation can be dated back to image analogy [10]. The goal of image analogy is modeling the transfer for a single paired input and output image. A recent work [22] investigated the image analogy problem in deep feature level. On the other hand, the goal of image-to-image translation is learning the transfer function between the input image set and output image set. For paired image-to-image translation, recent approaches used a paired dataset of input and output images to learn a parametric transfer function by using CNNs [26]. Isola et al. [13] used conditional generative adversarial network (GAN) [8] to learn the transfer function between two sets. With the similar idea, Sangkloy et al. [31] generated images conditioned on both sketched boundaries and sparse color strokes. Karacan et al. [15] synthesized images from the semantic layout. However, how to get the paired dataset is a tough problem in practice.

Unpaired Image-to-Image Translation. To handle the issue of obtaining paired dataset, unpaired image-to-image translation [17, 24, 42] methods have been proposed. The goal is tackling the unpaired data setting and learn the transfer function between two domains. To achieve this goal, Liu et al. [25] shared the weights of the generators of different GANs to learn the joint representation in cross domains. Later, Liu et al. [24] extended the framework of [25] by combining the variational autoencoders [19] with generative adversarial network. Another line of works, Zhu et al. [42] and Kim et al. [17], applied a cycle consistency loss to preserve the key attributes between two domains. Concurrently, Collection Style Transfer (CST) is another approach to perform unpaired image-to-image translation task. The goal of CST is to compose the stylized image by taking advantage of multiple style images. Sanakoyeu et al. [30] handled the CST task with the help of a style-aware loss and an adversarial loss to improve the stylization results. The difference between MST and CST is that MST can transfer any element in the content image set into a specific style. However, the style of CST is defined by all style images in the style image set, and CST usually doesn’t offer a method that can transfer the input content image into a particular style.

3. StyleRemix Network

The goal of style transfer can be defined as finding a stylized image o, which contains the visual fashion of target image s while retaining the content of the input image c. For MST task, recent methods often use two sets: content image set C = {c1, c2, ..., cN} and style image set S = {s1, s2, ..., sM}, to perform style transfer task. More specifically, MST methods aim to offer a model, which can
transfer any element of the content set $C$ to a specific style $s_j$ in the style set $S$.

### 3.1. Overview

StyleRemix aims to learn a compact and interpretable continuous vector representation for each style. The framework of StyleRemix is shown in Fig. 2, which consists of three parts: an autoencoder (i.e., encoder $E$ and decoder $D$), a style basis layer $B$, and style weights layers $W$. In StyleRemix, all styles in style set are decomposed and stored in a shared convolution layer and each particular style is constructed by a weighted style basis. As a result, we can simply represent each specific style with its style weights given the shared style basis. Finally, the weighted style basis is convolved with the intermediate feature maps [11] of input content image to obtain the stylized image $o$.

### 3.2. Encoder and Decoder

We do not set any limitation on the architecture of encoder and decoder as shown in Fig. 2. Since the stylized process in StyleRemix is the weighted style basis directly operating on the intermediate feature maps of the input content image given by autoencoder. Any symmetry or asymmetry autoencoder architecture can be used to produce the intermediate feature maps $F = E(c)$. For simplicity, we use a 3 layers symmetry encoder and decoder, which is similar to [14, 1].

### 3.3. Style Basis and Style Weights

#### Style Basis.

By sharing the redundant convolution operators, which may be used to produce similar texture patterns, coarsening or softening strokes among different styles, style basis $B$ is introduced to describe an embedding space of the style set. By doing so, a specific style $s_j$ in StyleRemix is described as a layer of weighted convolution filters, which map the high-level features of target style $s_j$ to input image $c$. The transferred feature maps $\tilde{F}_j$ given by style $s_j$ can be achieved by the convolution operation of the weighted style basis $B_j$ over the intermediate feature maps $F$:

$$\tilde{F}_j = B_j \odot F,$$

where $F \in \mathbb{R}^{c_{in} \times h \times w}$, $B_j \in \mathbb{R}^{c_{out} \times c_{in} \times k_h \times k_w}$, $\tilde{F}_j \in \mathbb{R}^{c_{out} \times h \times w}$, $c_{in}$ and $c_{out}$ are numbers of feature channels for $F$ and $\tilde{F}_j$ respectively, $(h, w)$ is feature map size, and $(k_h, k_w)$ is the kernel size. The weighted style basis $B_j$ is a point in learned style embedding space. In another word, $B_j$ is determined by its corresponding coordinates, i.e. style weights $\mathbf{w}_j$. Therefore, $B_j$ is defined as:

$$B_j = (1 \mathbf{w}_j^T) \odot B,$$

where $1 \in \mathbb{R}^{c_{out} \times k_h \times k_w \times 1}$, $\mathbf{w}_j \in \mathbb{R}^{c_{in} \times 1}$, and $B \in \mathbb{R}^{c_{out} \times k_h \times k_w \times c_{in}}$, and $\odot$ is element-wise product. $B_j$ in equation 1 and 2 are equivalent by reshaping operation.

In StyleRemix, style basis $B$ is implemented as a single convolution layer $B$, which locates at the end of encoder $E$ and the beginning of decoder $D$ as shown in Fig. 2.

#### Style Weights.

As we mentioned above, one purpose of style basis is compressing the redundancy of different styles. On the other hand, we hope that style basis itself should be interpretable. In another word, if the style base has no contribution to a specific style, its coefficient should be close to zero. To deliver this intuition, all weighted style basis are forced to be a convex combination of style basis $B$. That is to say, style weights $\mathbf{w}_j$ lies in the simplex:

$$\Delta_{c_{in}} = \{ \mathbf{w}_j \in \mathbb{R}^{c_{in} \times 1} \},$$

s.t. $\mathbf{w}_j[i] \geq 0$ and $\sum_i \mathbf{w}_j[i] = 1$. (3)

Style basis and style weights are jointly trained during one optimization iteration step. There are many ways to
learn style weights while meeting the constraint in Eq. 3. For simplicity, we implement style weights as a learnable layer \( \mathcal{W} \), which consists of two sublayers: a linear layer without bias \( \mathcal{L} \) and a softmax layer \( \mathcal{S} \). The function of linear layer is defined as:

\[
y = \alpha x,
\]

where \( \alpha \) is a scalar, \( x, y \in \mathbb{R}^{c_n \times 1} \). The function of softmax layer is defined as:

\[
y_i = \frac{\exp(y_i)}{\sum_j \exp(y_j)},
\]

where \( y_i \) is \( i \)th element in \( y \). By combining Eq. 4 and 5 and setting \( \alpha = 1 \), style weights can be achieve by:

\[
\omega \rightarrow \mathcal{W} \rightarrow \mathcal{S} \rightarrow \mathcal{L} \rightarrow \mathcal{S}.
\]

It is worth noticing that one of the state of the art \( \mathcal{L} \rightarrow \mathcal{S} \) setting where \( s \) is the goal of CST.

### 3.4. Loss Function.

StyleRemix includes two training branches: autoencoder branch \( (\mathcal{E} \rightarrow \mathcal{D}) \) and stylizing branch \( (\mathcal{E} \rightarrow \mathcal{W} \rightarrow \mathcal{D}) \). For the autoencoder training branch, the Euclidean Distance between the input content image \( c \) and the reconstructed content image \( \hat{c} \) is adopted to measure the reconstruction error, which can be achieved by:

\[
\mathcal{L}_r(\hat{c} , c) = ||\hat{c} - c||^2_2.
\]

For stylizing training branch, the perceptual loss defined in [14] is used to learn style representation. The perceptual loss can be achieved by:

\[
\mathcal{L}_p(\hat{c}, s_j) = \alpha \mathcal{L}_c(\hat{c}, c) + \beta \mathcal{L}_s(\hat{c}, s_j),
\]

where \( s_j \) is a specific style belonging to the style set \( S \).

Content loss \( \mathcal{L}_c \) and style loss \( \mathcal{L}_s \) are respectively defined to measure the difference of feature maps and gram matrix [6] between different images. Both feature maps and gram matrix are computed from VGG-16 network [34]. Content loss \( \mathcal{L}_c \) and style loss \( \mathcal{L}_s \) can be respectively achieved by:

\[
\mathcal{L}_c(\hat{c}, c) = \sum_{l \in \{l_c\}} ||F^l(\hat{c}) - F^l(c)||^2_2,
\]

\[
\mathcal{L}_s(\hat{c}, s_j) = \sum_{l \in \{l_s\}} \frac{1}{n_l} ||G^l(\hat{c}) - G^l(s_j)||^2_2,
\]

where \( F^l \) and \( G^l \) are feature maps and gram matrix calculated from \( l \)th layer of VGG-16 network [34], \( n_l \) is the total number of units of layer \( l \), \( \{l_c\} \) and \( \{l_s\} \) are layers which deliberately chosen to obtain \( F^l \) and \( G^l \).

### 3.5. Training Strategy.

StyleRemix is difficult to train because the model has to convincingly express all the styles in the style set while...
comprehensively describing the content information. Inspired by [1] and [21], our training strategy includes two phases: a warming-up stage and a finetuning stage.

The detailed training steps of the warming-up and the finetuning stages are shown in Fig. 4. The warming-up stage is trying to make the network get a general picture of the content and style information to be learned gradually. For this purpose, in the warming-up stage, each new style will be incrementally added to the current style set, i.e. from \( S_1 = \{s_1\}, S_2 = \{s_1, s_2\} \) to \( S_m = \{s_1, s_2, \ldots, s_m\} \). In the first \( K \) iterations, only the autoencoder is trained. After the following each \( K \) iterations, a new style set \( S_j \) will be trained until no more new style is available.

After the warming-up stage, we conduct the finetuning phase to enforce the network to strengthen the learned content and style characteristics. In particular, we employ the \( T + 1 \) training strategy, namely 1 round training on content images and \( T \) rounds training on the style set \( S_m \) to capture the content and style characteristics simultaneously.

4. Experiment

Implementation details. We employ a simple symmetry autoencoder same to [1] but with 256 channels style basis. The detailed architecture of network is shown in Table 1. The linear layer has 1 input channel and 256 output channels and the softmax layer has 256 input channels and 256 output channels. We employed VGG-16 [34], pre-trained on the ImageNet dataset [29], as perceptual feature extractor. We respectively chose layer Conv2_2 and layers Conv1_2, Conv2_2, Conv3_2, and Conv4_2 for content and style losses. The parameters \( \alpha \) and \( \beta \) in Eq. 7 are respectively set to: \( \alpha = 1 \) and \( \beta = 3 \times 4 \) for all results of StyleRemix in MST task except for specific statements. We randomly sample 1000 images from Microsoft COCO dataset [23] as content dataset. For style dataset, \( S_{50} \) contains 50 style images which are widely used in other MST papers. Note that another style dataset, which contains 50 style images which are widely used in other MST datasets, is employed for CST task.

During the training process, content images are randomly cropped to \( 512 \times 512 \), style images are scaled to 600 along large side. StyleRemix is trained with batch size 4, style weights initialized with the same value, parameters \( T \) in style learning stage is set to 2, learning rate 0.001 with decayed 0.8 at every 30K iterations, and Adam optimizer [18] for 300K iterations. Our implementation is based on PyTorch [28].

4.1. Multi-Style Transfer

We compare StyleRemix against several baselines to measure the performance of MST task. Baselines include: Neural style (NS) [6], Fast Neural Style (FNS) [14], StyleBank [1], Multi-Style Generative Network (MST-Net) [41], and Conditional Instance normalization (CIN) [3]. StyleBank is trained using the same network and parameters as StyleRemix. Other baselines are trained with default parameters from their open source projects, except that NS is optimized for 4000 iterations, MSG-Net is trained for 5 epochs. Besides, all MST baselines are trained with the same style set \( S_{50} \) as StyleRemix.

**Qualitative Result.** The stylized images \((512 \times 512)\) are shown in Fig. 5. In general, our method has achieved competitive stylization results compared to other baselines.

More specifically, for Single-Style Transfer (SST) methods, since plenty of parameters can be used to describe a single style, SST methods often yield better results than other baselines. However, NS produces many artifacts result in the details of stylized results looking unnatural, e.g. rows 1, 2, and 4. It may be caused by that the optimization process is based on the representation of the whole image. FNS yields larger texture pattern, which even larger than textures in target style image, e.g. rows 2, 3, and 4. This issue is possibly due to the reason that FNS uses \( 9 \times 9 \) kernels and deep architecture which has a larger receptive field.

For MST methods, since the diversity of style set has a significant influence on the stylized results, they have to compromise with different styles. For CIN, it can be observed the color histogram shift only, and some weird textures scattered inconsistently throughout, e.g. rows 1 and 3. Besides, CIN fails to reproduce texture patterns in style images. One possible reason is CIN’s style representation come from all the instance normalization layers, which do not explicitly separate content and style. For the same reason, CIN may be incapable of explicitly describing the textures of style. For MSG-Net, it seems to transfer the same structure but different color textures, e.g. rows 1, 2, and 3, and contain some large artifacts, e.g. rows 1, 3, and 5. The root cause may be that MSG-Net explicitly model the universal transfer function by a set of learnable weights. Likewise, MSG-Net suffers from some interferences among dif-

| Layer Type | Activation Dimensions |
|------------|-----------------------|
| Input      | \( H \times W \times 3 \) |
| Conv-IN-ReLU-S\(_1\) | \( H \times W \times 32 \) |
| Conv-IN-ReLU-S\(_2\) | \( \frac{1}{2}H \times \frac{1}{2}W \times 64 \) |
| Conv-IN-ReLU-S\(_2\) | \( \frac{1}{4}H \times \frac{1}{4}W \times 256 \) |
| Conv-IN-ReLU-S\(_2\) (StyleBasis) | \( \frac{1}{4}H \times \frac{1}{4}W \times 256 \) |
| Conv-IN-ReLU-S\(_{1/2}\) | \( \frac{1}{2}H \times \frac{1}{2}W \times 64 \) |
| Conv-IN-ReLU-S\(_{1/2}\) | \( H \times W \times 32 \) |
| Conv-S\(_1\) | \( H \times W \times 3 \) |

Table 1. The symmetry architecture used in StyleRemix. Conv: Convolution layer. IN: Instance Normalization layer. ReLU: ReLU layer. \( S_n \): stride = \( n \).
Comparing with SST methods, StyleRemix produces appealing stylized results in a different way. To be specific, StyleRemix creates the fundamental elements of style, such as texture patterns, softening strokes, instead of the abstract color area produced by NS or FNS, e.g. rows 2 and 5. Besides, thanks to the separated representations of content and style, StyleRemix yields more visually pleasing results than CIN and MSG-Net. For example, StyleRemix has captured textures in style images, e.g. rows 1, 3, and 6. Since StyleBank is a special case of StyleRemix, their stylized results are visually similar.

**Quantitative Results.** We compare the quantitative results of StyleRemix with other baselines in three aspects: Deception Rate [30] (DR), the number of float-point operations (FLOPs), and the cost of adding a new style (Cost/Style). The DR is calculated as the fraction of generated images which were classified by a network as the artworks of an artist for which the stylization was produced. The network is pre-trained on Wikiart to classify 624 artists. For DR, we use the available 14 artists from the open source project of Sanakoyeu et al. [30]. Since DR test mainly focus on paintings, we set the $\beta = 6e4$ for paintings stylization. And we generate 4200 stylized images (14 styles, 300 per style) for every method. FLOPs is calculated by counting the number of all convolution operations for single style. Since MSG-Net is based on the architecture of FNS [14] and need an extra parametric network to obtain style representation, we didn’t list its FLOPs. Cost/Style is defined by the parameters cost of adding a new particular style. Results in Tab. 2 show that our method has competitive performance compared to other baselines. Worth mentioning that, StyleRemix requires the lightest cost when adding a

| Style | Content | NS | FNS | CIN | MSG-Net | StyleBank | Ours |
|-------|---------|----|-----|-----|---------|-----------|------|
| ![Image](tree.png) | ![Image](woman.png) | ![Image](tree.png) | ![Image](woman.png) | ![Image](tree.png) | ![Image](woman.png) | ![Image](tree.png) | ![Image](woman.png) |
| ![Image](man.png) | ![Image](building.png) | ![Image](man.png) | ![Image](building.png) | ![Image](man.png) | ![Image](building.png) | ![Image](man.png) | ![Image](building.png) |
| ![Image](abstract.png) | ![Image](blue.png) | ![Image](abstract.png) | ![Image](blue.png) | ![Image](abstract.png) | ![Image](blue.png) | ![Image](abstract.png) | ![Image](blue.png) |
| ![Image](city.png) | ![Image](bridge.png) | ![Image](city.png) | ![Image](bridge.png) | ![Image](city.png) | ![Image](bridge.png) | ![Image](city.png) | ![Image](bridge.png) |
| ![Image](street.png) | ![Image](street.png) | ![Image](street.png) | ![Image](street.png) | ![Image](street.png) | ![Image](street.png) | ![Image](street.png) | ![Image](street.png) |

Figure 5. Some example results for qualitative evaluation.
4.2. Style Visualization

According to our best knowledge, only CIN [3] claims that they learn the embedding of style and more experiment results are presented in [7] to support this opinion. We compare the style embedding learned by StyleRemix and CIN. The style representation of StyleRemix is $S_{sr} = \{\gamma_s, \beta_s\} \in \mathbb{R}^{256 \times 1}$, and the style representation of CIN is $S_{cin} = \{\gamma_{cin}, \beta_{cin}\} \in \mathbb{R}^{3206 \times 1}$, where $\gamma_s$ and $\beta_s$ are respectively the learned mean and standard deviation of instance normalization layers.

We examine the low dimensional style representations of both CIN and StyleRemix. The style parameter of StyleRemix, in Eq. 7, is set to $\beta = 6e4$, to capture more information about style. Also, in order to achieve low dimensional display, we first employ Principal Component Analysis to reduce both style representation to 10 dimensions and then t-SNE [27] dimensional reduction technique to further reduce the 10 dimensions style representation to 2 dimensions. Notice t-SNE will necessarily distort the representation significantly in order to compress the representation to small dimensionality. Therefore, we restrict our analysis to the qualitative description. The t-SNE results are shown in Fig. 6.

Comparing the t-SNE results of CIN in Fig. 6 (left) and StyleRemix in Fig. 6 (right), one may notice StyleRemix has better locality than CIN. As shown in Fig. 6, we use different color boxes to identify different class styles. For instance, for the t-SNE results of StyleRemix, in the yellow box, the styles contain only black, white color and both of them look like Sketch.

We also show the correlation matrix of 10 selected styles, which is directly calculated by original high dimensional style weights, in Fig. 7. The correlation matrix provides strong evidence that the style embedding learned by StyleRemix can distinguish styles in semantic similarity. The visually similar styles have high correlation coefficients, and the visually different styles have low correlation coefficients. Therefore, the correlation matrix of StyleRemix (Fig. 7 right) shows Blocking Effect, which is not observed in the correlation matrix of CIN (Fig. 7 left).

4.3. Style Manipulation

We conduct the Style Remaking manipulation by decomposing all styles into the same style embedding space and recombining them to create a large number of new style effects. We show some heuristic examples to show how to remix styles via simply operating the style weights.

**Convex Combination.** Convex combination offers a smooth transition from one style to the other, which can be achieved by simply apply the convex combination in style weights of different styles. Supposing there are two different styles $s_l$ and $s_k$ with style weights $w_l$ and $w_k$, the convex combination style $s_{new}$ can be achieved by:

$$w_{new} = \alpha w_l + (1 - \alpha) w_k,$$  \hspace{1cm} (10)

where $0 \leq \alpha \leq 1$, $w_{new}$ is the style weights of $s_{new}$. Fig. 8 presents the convex combination results, of which one can observe the smooth transition between the two styles.

**Perturbation Study.** We show an example how to remix styles by adding Gaussian noise to a particular style. The results are shown in Fig. 9. We add Gaussian noise $\varepsilon \sim \mathcal{N}(\mu, \sigma^2)$, where $\mu = 1/256, \sigma = 0.005$, to the style weights $w_j$. Then normalized the vector $\mathbf{v} = w_j + \varepsilon$ by $\mathbf{v} = \mathbf{v}/\sum_{i=1}^{256} v_i$. One may notice that parts of the
Figure 8. The convex combination results of two different styles. \( \alpha = 1.0, 0.8, 0.6, 0.4, 0.2, 0.0 \) respectively.

Figure 9. The perturbation study results of two different styles. The first column are style images. The second column are the original stylized results and its corresponding style weights. The following columns are obtained by adding Gaussian noise.

Style weights of perturbed style are negative. Interestingly, it is difficult to differentiate the perturbed stylized results from the original results. These results may suggest that StyleRemix has captured a smooth embedding space, which is robust to noise to some extend.

Collection Style Transfer. StyleRemix may offer a way to perform Collection Style Transfer (CST) task. As we mentioned before, the style of CST is defined by all styles in the style set. As the style elements of the style set are fully represented by the style basis in StyleRemix, it is possible to define the style of CST by manipulating the style weights. We train StyleRemix with 50 random paintings of Vincent Van Gogh from Wikiart [16]. The parameters \( \alpha \) and \( \beta \) in Eq. 7 are respectively set to: \( \alpha = 1 \) and \( \beta = 6e4 \). We show CST results in Fig. 10 with two simple manipulating strategies. One is to use the average vector of all 50 styles (row 1) as style weights, another is to set the same value \( 1/256 \) (row 2) for all weights. They yield similar results as both are capturing all the style elements of Van Gogh paintings. On the other hand, they have subtle difference on chrominance, luminance and local texture as the style weights varied.

4.4. Style Scalability

Style scalability is critical to SEL methods. However, it is difficult to measure since the maximum capabilities of a single model is highly related to the particular set of styles. For example, a SEL method is likely to have good style scalability if all styles come from one domain instead of different sources. To evaluate the capacity of StyleRemix, we experiment with the same diverse style set \( S_{50} \) as Section 4.1 and study how varying the number of styles affects the stylization. We train StyleRemix with subsets of \( S_{50} \), i.e. \( S_5 \subset S_{10} \subset S_{15} \), with the same parameters. According to Fig. 11, as the number of styles increases, the result shows visually consistent stylization effect though some details of style may fade progressively. This phenomena indicates that StyleRemix can handle a certain number of diverse styles without significant visual degradation.

5. Conclusion

In this paper, we have proposed a style embedding learning approach, StyleRemix, for multi-style transfer task. Combining interpretability with efficiency, our method effectively perform multi-style transfer and produce visually appealing stylization. In addition, the proposed method can be extended style visualization task. More importantly, StyleRemix provides several ways to remix styles in the style embedding space. Experimental results demonstrate the effectiveness of StyleRemix on many multi-style transfer tasks. As future direction, one may extend StyleRemix to handle arbitrary style transfer.
References

[1] D. Chen, L. Yuan, J. Liao, N. Yu, and G. Hua. Stylebank: An explicit representation for neural image style transfer. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), volume 1, page 4, 2017. 1, 2, 3, 4, 5

[2] T. Q. Chen and M. Schmidt. Fast patch-based style transfer of arbitrary style. Advances in Neural Information Processing Systems (NIPS), Constructive Machine Learning Workshop, 2016. 1, 2

[3] V. Dumoulin, J. Shlens, and M. Kudlur. A learned representation for artistic style. Proceedings of International Conference on Learning Representations (ICLR), 2017. 1, 2, 5, 7

[4] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In IEEE International Conference on Computer Vision (ICCV), page 1033. IEEE, 1999. 1

[5] L. Gatys, A. S. Ecker, and M. Bethge. Texture synthesis using convolutional neural networks. In Advances in Neural Information Processing Systems (NIPS), pages 262–270, 2015. 1

[6] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2414–2423. IEEE, 2016. 1, 4, 5

[7] G. Ghiasi, H. Lee, M. Kudlur, V. Dumoulin, and J. Shlens. Exploring the structure of a real-time, arbitrary neural artistic stylization network. arXiv preprint arXiv:1705.06830, 2017. 1, 7

[8] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Advances in neural information processing systems (NIPS), pages 2672–2680, 2014. 2

[9] S. Gu, C. Chen, J. Liao, and L. Yuan. Arbitrary style transfer with deep feature reshuffle. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 8222–8231, 2018. 1, 2

[10] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin. Image analogies. In Proceedings of the 28th annual conference on Computer graphics and interactive techniques, pages 327–340. ACM, 2001. 2

[11] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. science, 313(5786):504–507, 2006. 3

[12] X. Huang and S. J. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In IEEE International Conference on Computer Vision (ICCV), pages 1510–1519, 2017. 1, 2

[13] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017. 2

[14] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In European Conference on Computer Vision (ECCV), pages 694–711. Springer, 2016. 1, 3, 4, 5, 6

[15] L. Karacan, Z. Akata, A. Erdem, and E. Erdem. Learning to generate images of outdoor scenes from attributes and semantic layouts. arXiv preprint arXiv:1612.00215, 2016. 2

[16] S. Karayev, M. Trentacoste, H. Han, A. Agarwala, T. Darrell, A. Hertzmann, and H. Winnemöller. Recognizing image style. arXiv preprint arXiv:1311.3715, 2013. 8

[17] T. Kim, M. Cha, H. Kim, J. K. Lee, and J. Kim. Learning to discover cross-domain relations with generative adversarial networks. arXiv preprint arXiv:1703.05192, 2017. 2

[18] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. Proceedings of the International Conference on Learning Representations (ICLR), 2014. 5

[19] D. P. Kingma and M. Welling. Auto-encoding variational bayes. Proceedings of the International Conference on Learning Representations (ICLR), 2013. 2

[20] X. Li, S. Liu, J. Kautz, and M.-H. Yang. Learning linear transformations for fast arbitrary style transfer. arXiv preprint arXiv:1808.04537, 2018. 1, 2

[21] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang. Diversified texture synthesis with feed-forward networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017. 1, 2, 5

[22] J. Liao, Y. Yao, L. Yuan, G. Hua, and S. B. Kang. Visual attribute transfer through deep image analogy. ACM Transactions on Graphics (TOG), 2017. 2

[23] 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 European Conference on Computer Vision (ECCV), pages 740–755. Springer, 2014. 5

[24] M.-Y. Liu, T. Breuel, and J. Kautz. Unsupervised image-to-image translation networks. In Advances in Neural Information Processing Systems (NIPS), pages 700–708, 2017. 2

[25] M.-Y. Liu and O. Tuzel. Coupled generative adversarial networks. In Advances in neural information processing systems (NIPS), pages 469–477, 2016. 2

[26] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 3431–3440, 2015. 2

[27] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. Journal of machine learning research (JMLR), 9(Nov):2579–2605, 2008. 7

[28] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017. 5

[29] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. International Journal of Computer Vision (IJCV), 115(3):211–252, 2015. 5

[30] A. Sanakoyeu, D. Kotovenko, S. Lang, and B. Ommer. A style-aware content loss for real-time hd style transfer. European Conference on Computer Vision (ECCV), 2018. 2, 6

[31] P. Sangkloy, J. Lu, C. Fang, F. Yu, and J. Hays. Scribbler: Controlling deep image synthesis with sketch and color. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), volume 2, 2017. 2
[32] F. Shen, S. Yan, and G. Zeng. Neural style transfer via meta networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 8061–8069, 2018.

[33] L. Sheng, Z. Lin, J. Shao, and X. Wang. Avatar-net: Multi-scale zero-shot style transfer by feature decoration. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 8242–8250, 2018.

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

[35] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. S. Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In International Conference on Machine Learning (ICML), pages 1349–1357, 2016.

[36] D. Ulyanov, A. Vedaldi, and V. S. Lempitsky. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), volume 1, page 3, 2017.

[37] 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.

[38] X. Wang, G. Oxholm, D. Zhang, and Y.-F. Wang. Multimodal transfer: A hierarchical deep convolutional neural network for fast artistic style transfer. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), volume 2, page 7, 2017.

[39] L.-Y. Wei and M. Levoy. Fast texture synthesis using tree-structured vector quantization. In Proceedings of the 27th annual conference on Computer Graphics and Interactive Techniques, pages 479–488. ACM Press/Addison-Wesley Publishing Co., 2000.

[40] K. Yanai and R. Tanno. Conditional fast style transfer network. In Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval (ICMR), pages 434–437. ACM, 2017.

[41] H. Zhang and K. Dana. Multi-style generative network for real-time transfer. European Conference on Computer Vision (ECCV) Workshops, 2018.

[42] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. Proceedings of the IEEE International Conference on Computer Vision (CVPR), 2017.