Learning Linear Transformations for Fast Image and Video Style Transfer

Xueting Li*[1], Sifei Liu*[2], Jan Kautz[2], and Ming-Hsuan Yang[1,3]

[1]University of California, Merced, [2]NVIDIA, [3]Google Cloud

Abstract

Given a random pair of images, a universal style transfer method extracts the feel from a reference image to synthesize an output based on the look of a content image. Recent algorithms based on second-order statistics, however, are either computationally expensive or prone to generate artifacts due to the trade-off between image quality and run-time performance. In this work, we present an approach for universal style transfer that learns the transformation matrix in a data-driven fashion. Our algorithm is efficient yet flexible to transfer different levels of styles with the same auto-encoder network. It also produces stable video style transfer results due to the preservation of the content affinity. In addition, we propose a linear propagation module to enable a feed-forward network for photo-realistic style transfer. We demonstrate the effectiveness of our approach on three tasks: artistic style, photo-realistic and video style transfer, with comparisons to state-of-the-art methods. The project website can be found at https://sites.google.com/view/linear-style-transfer-cvpr19.

1. Introduction

A style transfer method takes a content image and a style image as inputs to synthesize an image with the look from the former and feel from the latter. In recent years, numerous style transfer methods have been developed. The method by Gatys et al. [8] iteratively minimizes content and style reconstruction losses between the target image and input images. To reduce the computational cost, a few approaches have since been developed based on feed-forward networks [13, 30]. However, these approaches do not generalize to universal style images with one single network.

For universal style transfer, a number of methods explore the second order statistical transformation from reference image onto content image via a linear multiplication between content image features and a transformation matrix [12, 18, 19]. The AdaIn method [12] matches the

*equal contribution

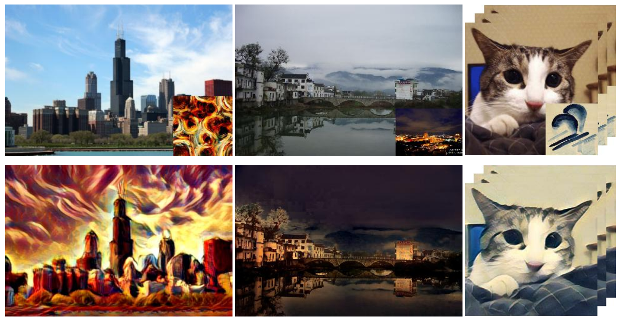
(a) artistic (b) photo-realistic (c) video

Figure 1. Applications of the proposed algorithm. (a) Artistic style transfer. (b) Photo-realistic style transfer. (c) Video style transfer (click on the image to see animations using Adobe Reader). The style thumbnails are shown in lower right corners.

means and variances of deep features between content and style images. The WCT [18] algorithm further exploits the feature covariance matrix instead of the variance, by embedding both whitening and coloring processes within a pre-trained encoder-decoder module. However, these approaches directly compute these matrices from deep feature vectors. As such, these methods do not present general solutions to this problem. Furthermore, such matrix-computation-based methods can be computationally expensive due to the high dimensionality of deep feature vectors.

In this work, we propose a learnable linear transformation matrix which is conditioned on an arbitrary pair of content and style images. We derive a linear transform and draw connections to the reconstruction objective (squared Frobenius norm of the difference between Gram matrices) widely used in style transfer [8, 13, 30, 12]. We learn the transformation matrix with two light-weighted CNNs, and show that this approach is significantly faster than computing the transforms from features (e.g., [18, 19]). Specifically, the learning-based transformation matrix can be controlled by different levels of style losses and is computationally efficient. In addition, we present a linear propagation module [22] that can correct distortions and artifacts in contours and textures, which are commonly observed in style transfer results. We then integrate this module into our style transfer network to generate undistorted results for photo-realistic style transfer.

The contributions of this work are summarized as follows. First, we propose an efficient (about 140 fps) and flexible style transfer model that preserves content affinity during the style transfer process. Second, we show that the proposed method can be flexibly applied to different tasks, including but not limited to artistic style and video style transfer by slightly modifying the network architecture (see Fig. 1(a) and (c)). Third, we develop a linear propagation module that can be equipped with a feed-forward neural network to generate high-quality undistorted results for photo-realistic style transfer (see Fig. 1(b)).

2. Related Work

Deep learning based style transfer has been intensively studied [15, 32, 33, 9, 31, 17, 1, 7] to match statistical information between content and style images based on features extracted from pre-trained convolutional neural networks. One main drawback with the method by Gatys et al. [8] is the heavy computational cost due to the iterative optimization process. Fast feed-forward approaches [13, 30, 16] address this issue by training feed-forward neural networks that minimize the same feature reconstruction loss and style reconstruction loss as [8]. However, each feed-forward network is trained to transfer one fixed style. Dumoulin et al. [4] introduce an instance normalization layer that allows 32 styles to be represented by one model, and Li et al. [17] encode 1,000 styles by using a binary selection unit for image synthesis. Nevertheless these models are not able to transfer an arbitrary style onto a content image.

Universal style transfer. Several methods [10, 12, 27, 34] have been proposed to match mean and variance of content and style feature vectors to transfer an arbitrary style onto a content image. However, these methods do not model the covariance of features and may not synthesize images well. Li et al. [18] resolve this problem by applying both whitening and coloring transforms with pre-trained image reconstruction auto-encoders. However, this approach is computationally expensive due to the need of large matrix decompositions at multiple levels. Shen et al.[6] propose to train a meta network that generates a 14 layer network for each content and style image pair. However, it requires extra memory for each content/style pair and does not explicitly model the second-order image statistics.

Photo-realistic style transfer. Photo-realistic style transfer methods [19, 23] aim to synthesize images without distorting geometric structures. Luan et al. [23] introduce a local-affinity based energy term as an extra loss function of the model by Gatys et al. [8], where stylized results are ob-

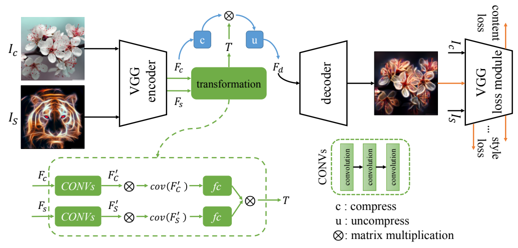

Figure 2. Overview of the proposed method. Our model contains a pre-trained encoder and a decoder, a loss module, a transformation module with the compress/uncompress blocks. Only the transformation module, as well as the pair of compress and uncompress blocks are learnable, while all the others are fixed (black). We use orange arrows to denote the losses and "T" to denote the point where a style is transformed (see Section 3.2 for technical details).

tained through time-consuming optimization. Li et al. [19] replace the computationally expensive transforms with a feed-forward network and solve the optimization problem with a closed-form solution to synthesize images. In this work, we introduce an efficient linear propagation module for photo-realistic style transfer. The whole pipeline can be implemented by a single feed-forward network, which is faster and GPU-friendly compared to existing methods.

3. Style Transfer by Linear Transformation

The proposed model contains two feed-forward networks, a symmetric encoder-decoder image reconstruction module and a transformation learning module, as shown in Fig. 2. The encoder-decoder is trained to reconstruct any input image faithfully. It is then fixed and serves as a reconstruction network in the remaining training procedures. Instead of computing the transformation of the 2^{nd} moment statistic in the intermediate layers of the auto-encoder, as being conducted in many previous work [18, 19, 29], we make the transformation matrix learnable by outputting it via a light-weighted CNN block. To learn the transformation via a feed-forward network, we need some supervision signal. We use a pre-trained VGG-19 network to compute style losses at multiple levels and one content loss in a way similar to the prior work [13, 12]. The proposed feed-forward convolutional neural network, which is able to transfer arbitrary styles efficiently at 140 fps, is much faster than the computation-based methods.

3.1. Learning Universal Style Transfer

We denote the feature map of the top-most encoder layer as $F(I) \in \mathcal{R}^{N \times C}$, where I is an input image, N is the number of pixels, and C is the number of channels. For presentation clarity, the feature maps of a content and style image pair are denoted as row vectors F_c and F_s. We formulate the image style transfer problem as a linear transformation between the content feature F_c and learned matrix T, with the transformed feature vector F_d. We use ϕ_s to denote a "virtual" feature map that provides the desired style. For several style transfer methods based on matrix transformation [18, 12], $\phi_s = F_s$. In this work, ϕ_s accommodates multiple forms based on different configurations of style losses. Thus, ϕ_s can be described as a nonlinear mapping of F_s as $\phi_s = \phi(F_s)$. We denote $\bar{F}$ as the vectorized feature map F with zero mean.

Our goal is to learn the optimal F_d such that the statistic of the transformed feature from the encoder matches that of the desired style , which is determined by the style losses in the loss network. It can be expressed as minimizing the difference of centered covariance between F_d and ϕ_s:

$$\begin{gathered} F_d^* = \arg\min_{F_d} \tfrac{1}{NC} \|\bar{F}_d \bar{F}_d^\top - \bar{\phi}_s \bar{\phi}_s^\top\|_F^2 \\ s.t. \bar{F}_d = T\bar{F}_c. \end{gathered} \tag{1}$$

By substituting the linear constraint into Eq. (1), the minima is obtained when

$$T\bar{F}_c\bar{F}_c^\top T^\top = \bar{\phi}_s\bar{\phi}_s^\top. \tag{2}$$

The centered covariance of $\bar{F}_c$ is $cov\,(F_c) = \bar{F}_c\bar{F}_c^\top = V_c D_c V_c^\top$, and the corresponding singular value decomposition (SVD) is $cov\,(\phi_s) = \bar{\phi}_s\bar{\phi}_s^\top = V_s D_s V_s^\top$. It is easy to show that

$$T = \left(V_s D_s^{\frac{1}{2}} V_s^\top\right) U \left(V_c D_c^{-\frac{1}{2}} V_c^\top\right), \tag{3}$$

is one set of solutions to Eq. (2) where $U \in \mathcal{R}^{\mathcal{C}\times\mathcal{C}}$ is a C-dimensional orthogonal group. In other words, T is solely determined by the covariance of the content and style image feature vectors. Given T, the transformed feature is obtained by $\bar{F}_d + mean(F_s)$, which simultaneously aligns to the mean and the covariance statistics of the target style, and thus describes the style reconstruction used by most existing approaches. In the following, we discuss how to select a proper model for learning T.

3.2. Learning Transformation T

Given that T is only conditioned on the content and style images, one feasible approach is to use a network that takes both images to directly output a $C \times C$ matrix. According to Eq. (3), the terms of content and style are decoupled, so we use two independent CNNs for the content/style inputs.

The formulation in Eq. (2) suggests three input options to the CNNs: (i) images (c and s), (ii) feature maps (F_c and F_s), and (iii) covariance matrices ($cov\,(F_c)$ and $cov\,(F_s)$). In this work, we use the third option where each CNN takes the covariance of feature vectors and outputs a $C \times C$ intermediate matrix. These two matrices are then multiplied to formulate the T as shown in Fig. 2. We explain the design options in the following paragraph.

First, we hope the module that outputs T should be flexibly adapted to both a full content image, and an arbitrarily-shaped region, e.g., for stylization within a segmentation mask, as in Section 5. This property does not hold when the model input is the image or feature map. For example, it is easy to show that $T = \bar{\phi}_s U \bar{F}_c^{-1}$ is one of the solutions for Eq. (2), which is based on feature inputs. However, it also requires that the content and style features inputing to the transformation module have the same dimensions. Second, since T describes the style transformation, it focus on describing the global statistics shift, rather than any image spatial information. Amazingly, using the covariance matrices as the inputs addresses both challenges. The ablation study in Section 5 shows that using a covariance matrix as model input leads to better stylized images (see Fig. 5).

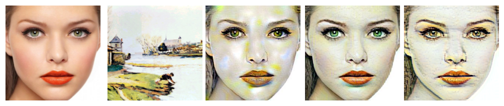

Figure 3. Style transfer using a shallow auto-encoder described in Section 3.3. Our method faithfully captures styles even when a shallow auto-encoder is used.

3.3. Learning Video Style Transfer

We show that since the transformation is learnable, the model flexibly accommodates numerous combinations of the auto-encoder and the loss modules. Here we show a practical solution for video style transfer. By using a shallower auto-encoder, the network can generate stabilized transferred videos as the proposed linear transformation method is able to preserve affinity across frames. Instead of enforcing temporal consistency through any cross-frame alignment strategy, e.g., optical flow warping [11], we ensure a transferred video to have similar affinity as the content video. Although the same principle could be applied to both WCT and Adain methods, it is difficult to preserve affinity as these algorithms are developed based on relatively deep auto-encoders (see Fig. 10) and perform poorly when shallower auto-encoders are used (see Fig. 3).

Affinity describes the pairwise relation of pixels, features, or other image elements. Preserving the affinity of content image/video, which indicates that the dense pairwise relations among pixels in the content image are preserved well in the stylized result, is a key solution for generating stabilized video. In principle, style transfer and affinity preservation belong to different forms of matrix multiplications: the former is pre-multiplication while the latter is post-multiplication to the feature vector (see Eq. (1)). Given Eq. (1) and the normalized affinity for a vectorized feature $F \in \mathcal{R}^{\mathcal{N}\times\mathcal{C}}$:

$$\mathit{aff}\,(F) = \bar{F}^\top cov\,(F)^{-1}\,\bar{F} \tag{4}$$

It is straightforward to show that the affinities of F_c and F_d are equal to each other by the following equations:

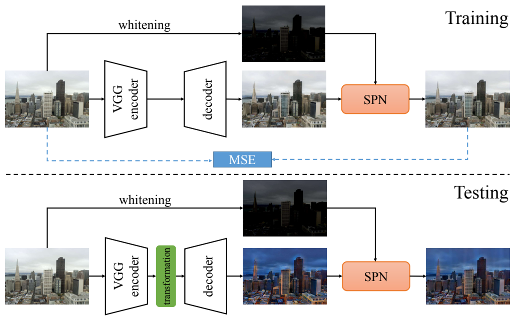

Figure 4. Photo-realistic style transfer with the linear propagation module. We first fix pretrained auto-encoder and train a SPN module on reconstructed images to minimize distortions caused by the auto-encoder using the whitened content image as guidance. Given a test image, we directly apply it on style transferred image.

$$\bar{F_d}^\top cov\left(\phi_s\right)^{-1} \bar{F_d} = \bar{F_d}^\top \left(T\bar{F_c}\bar{F_c}^\top T^\top\right)^{-1} \bar{F_d} \qquad (5)$$
$$= \bar{F_c}^\top T^\top \left(T\bar{F_c}\bar{F_c}^\top T^\top\right)^{-1} T\bar{F_c} = \bar{F_c}^\top cov\left(F_c\right)^{-1} \bar{F_c}.$$

Thus, the proposed linear transformation method is capable of preserving the feature affinity of the content image.

Although the affinity is preserved via the proposed linear transformation model, there are three more factors can cause temporal inconsistency: (i) the non-linear layers in the decoder; (ii) the trace $tr\left(\bar{\phi_s}\bar{\phi_s}^\top\right)$, determined by how the loss is enforced (see similar analysis in [11]), that affects F_d; and (iii) the spatial resolution of the top layer that determines the scale at which the affinity is preserved. In this work, we use a shallower model up to the *relu3_1* layer of the VGG-19 model to better preserve affinity. This shallower model can still express rich stylization when a relatively deep style loss network (e.g., using up to *relu4_1* in the loss module) is used. We show that the stylized results contain fewer distortions and are more stable than existing methods [8, 13] that are not based on the formulation in Eq. (1) (see Fig. 8 and 10). Note that for a shallower encoder, the corresponding F_c (e.g., *relu2_1*) is not expressive enough to transfer abstract styles.

3.4. Learning Photo-realistic Style Transfer

Photo-realistic style transfer is another important application which aims to synthesize images without structural distortions. Although using a shallower network described in Section 3.3 can generate results that better align with the content images, the stylized results still contain significant artifacts due to the distortion caused by the deep auto-encoder. To address this issue, we exploit the spatial propagation network (SPN) proposed by Liu et al. [22]. The SPN is a generic framework that learns to model pixel pairwise relations. We insert a SPN as a "anti-distortion filter" on top of the auto-encoder, while the coefficients of the filter are learned through a CNN guidance network in a data-driven manner (See [22] for more details). Specifically, we train a SPN using the reconstructed and content images since no photo-realistic stylized ground truth is available. The content image also serves as the input to the guidance network. Then we apply the SPN on the stylized image to minimize distortions caused by the auto-encoder. However, affinity learned directly from the original content image will include color information and affect the stylized image. Instead of using the original content image as guidance, we use the whitened content image to remove color information and encode only affinity information. Fig. 4 illustrates the training and testing process of our method.

4. Model Analysis

In this section, we discuss the advantages of the linear transformation method for style transfer in terms of run-time efficiency and model size. The analysis of the propagation module for photo-realistic style transfer is presented in Section 5.

Run-time efficiency. Our method performs efficiently (see Table 1) since it does not involve time-consuming matrix computation, e.g., SVD in WCT [18]. On the other hand, the AdaIn [12] method performs efficiently as only first-order statistics (i.e., variance) are involved but at the expense of image quality. We show that while the AdaIn method can generate favorable results with a deeper auto-encoder, the proposed model performs well since the covariance shift is accurately modeled with a much shallower auto-encoder (which brings further speedup).

Model size. Our model can transfer rich styles using a single auto-encoder, in contrast to the cascade networks in the WCT method [18], thanks to the learnable transformation. As analyzed in [8], Gram matrices of features from different layers capture different details for style transfer, e.g., features from lower layers usually capture color and textures while those based on features from higher layers capture common patterns. Since the transformation in the WCT method is not learnable, it has to present multi-level styles by cascading several auto-encoders with different layers at the expense of additional computational loads. In contrast, a single transformation T in our model can express rich styles by using a combination of multiple style reconstruction losses in the loss module, e.g., via $\{$*relu1_1*, *relu1_2*, *relu1_3*, *relu1_4*$\}$ in a way similar to [12, 13] as shown in Fig. 6. Thus, the proposed method achieves the same goal without additional computational overhead during the inference stage.

5. Experimental Results

We discuss the experimental settings and present ablation studies to understand how the main modules of the proposed algorithm contribute. We evaluate the proposed algorithm against the state-of-the-art methods on artistic, video and photo-realistic style transfer.

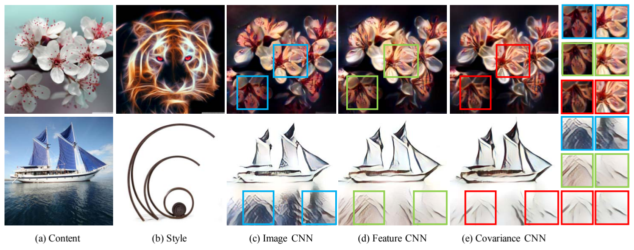

Figure 5. Stylized images using different model inputs. The CNN in (c) takes content/style images as input, the CNN in (d) takes content/style features as input, and the CNN in (e) takes content/style features as input but feeds the covariance matrix of encoded features into the final fully connected layer (see Fig. 2 and Section 5 for details).

5.1. Experimental Settings

Encoder-decoder. Our method contains an encoder with the first few layers from the VGG-19 model [28] pre-trained on the ImageNet dataset [3] and a symmetric decoder. We train the decoder on the MS-COCO dataset [21] from scratch to reconstruct images. This module is then fixed throughout the remaining network training procedure. We show various design options of the encoder-decoder with varying model depth for different applications.

Transformation module. Our transformation module consists of two CNNs where each takes either content or style features as input and outputs a transformation matrix, respectively. These two transformation matrices are then multiplied to generate the final transformation matrix T. We compute the transferred feature vector F_d by multiplying a content feature vector F_c with T, and feed F_d into the decoder to generate the stylized image. Within each CNN in the transformation module, as discussed in Section 3, we compute the transformation matrix from feature covariance to handle images of any size during inference. In order to learn a nonlinear mapping from feature covariance to the transformation matrix, a number of fully-connected layers are used. However, this leads to a large memory requirement and model size since the covariance matrix usually has high dimensions, i.e., 512×512 in *relu4_1*. Thus, we factorize the model by first encoding the input features to the reduced dimensions (e.g., $512 \rightarrow 32$) via three consecutive convolutional units (denoted as "CONVs" in Fig. 2), where each is equipped with a 3×3 *conv* and a *relu* layer. The covariance matrix of the encoded feature is then fed into one *fc* unit to compute the transformation matrix. We further use a pair of convolutional layers to "compress" the content features, and "uncompress" the transformed features to the corresponding dimensions. Overall, our transformation module is compact, efficient and can be easily trained for any combination of styles.

Dataset and training settings. We use the MS-COCO dataset [21] as our content images and the WikiArt database [24] as our styles. Both datasets contain roughly 80,000 images. We keep the image ratio and re-scale the smaller dimension of each training image to 300 pixels. We then apply patch-based training by randomly cropping a region of 256×256 pixels from it as one training sample with randomly flip at the probability of 0.5. During the inference stage, our model is able to handle any input size for both content and style images, as discussed in Section 3. We train our network using the Adam solver [14] with a learning rate of 10^{-4} and a batch size of 8 for 10^5 iterations. The training roughly takes 5 hours on a single Titan XP GPU for each model in our Pytorch [25] implementation.

The source code, trained models and real-time demos will be made available to the public.

5.2. Ablation Studies for Transformation Modules

One single network? As discussed in Section 3.1, using two separate CNNs for learning T is more suitable than sharing a single CNN for content and style images. To verify this, we train a single CNN to learn the transformation matrix, which takes both content and style images as inputs. However, this model does not converge during training, which is consistent with our discussion.

Inputs to the transformation module. We discuss in Section 3 that the formulation in Eq. (2) suggests three input options to the CNNs in the transformation module. We implement three models with the same architecture except for the image CNN, which uses five convolutional units instead of three. Note that neither image nor feature CNN uses matrix multiplication between the "CONVs" and "fc" modules, as shown in Fig. 2. We show stylized images in Fig. 5. In general, the image CNN does not generate faithful stylized results, e.g., the patches in the second row of Fig. 5(c) still retain color pixels from the original content image. On the other hand, the proposed covariance CNN generates results more similar to the styles (e.g., abstract lines in the close-ups of Fig. 5(e)) than the other methods.

Combining multi-level style losses. Features from different layers capture different style details. We show our algorithm allows a flexible combination of multiple levels of styles within a single transformation matrix T by making use of different style reconstruction losses in the loss module. We apply two types of auto-encoders – encoders up to $relu3\_1$ and $relu4\_1$ in the VGG-19 model, and use different style reconstruction losses – single loss layer on $relu1\_1$, $relu2\_1$ or $relu3\_1$, and multiple loss layers $\{relu1\_1, \cdots, relu4\_1\}$, as shown in Fig. 6. The results show that both transferring content features of lower layers (*relu3_1* in row (i)) and using one single style reconstruction loss from lower layers (*relu1_1, relu2_1* in column (a) and (b)) lead to more photo-realistic visual effects. On the other hand, more stylized images can be generated using the style reconstruction loss from higher layers (e.g., Fig. 6

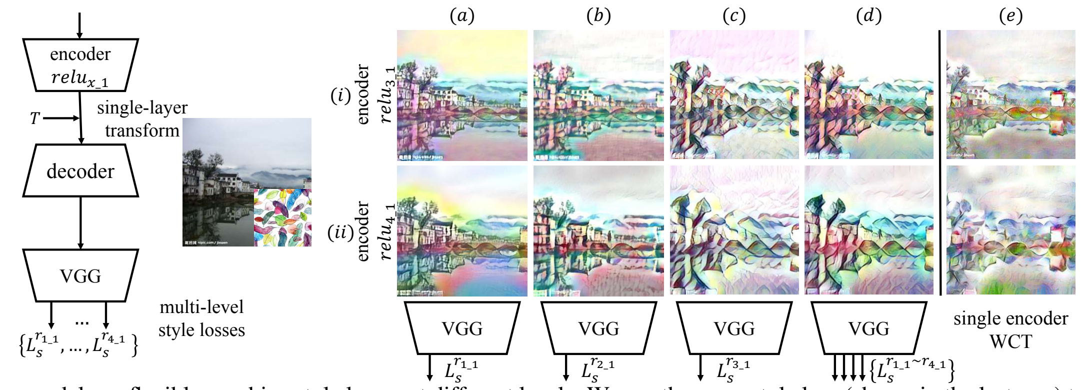

Figure 6. Our model can flexibly combine style losses at different levels. We use the same style loss (shown in the last row) to train models in each column. Row 1 and 2 show stylized images by transferring content features from *relu3_1* and *relu4_1*, respectively. The last column shows results by one single layer WCT (*relu3_1* on the top and *relu4_1* at the bottom).

Image Size	256	512	1024
Ulyanov et al. [30]	0.013	0.028	0.092
Gatys et al. [9]	16.51	59.45	N/A
Huang et al. [12]	0.019	0.071	N/A
Li et al. [18]	0.922	1.080	N/A
Ours (*relu3_1*)	0.007	0.025	0.100
Ours (*relu4_1*)	0.010	0.036	0.146

Table 1. Runtime performance. "N/A" indicates the input cannot fit 12GB GPU. Runtime is measured in seconds using the source code on a single Titan XP GPU. For WCT [18], we use the version that cascades four different encoder-decoder modules because of its best performance and faster speed.

column (c) and (d)). Specifically, the auto-encoder up to $relu3\_1$ (row (i) in Fig. 6) generates more undistorted results for all types of style losses than using auto-encoder up to $relu4\_1$ (row (ii) of Fig. 6).

We also show stylized results by the single-encoder WCT in column (e) of Fig. 6 (*relu3_1* on top and *relu4_1* on bottom), where neither model performs well (e.g., neither color nor texture is well aligned, and edges are blurry). In contrast, our method generates visually pleasing results for richer styles, as shown in column (d) in Fig. 6, where no cascade modules are required. Due to the flexibility of our algorithm, we are able to choose different settings for different tasks without any computational overheads. Specifically, we transfer content image features from *relu4_1* layer for artistic style transfer. For video style transfer, we transfer content frame features from *relu3_1* to minimize distortions. For all tasks, we compute style reconstruction losses using features from *relu1_1*, *relu2_1*, *relu3_1* and *relu4_1* to capture different style details.

5.3. Artistic Style Transfer

We evaluate the proposed algorithm with three state-of-the-art methods for artistic style transfer: optimization based method [8], fast feed-forward network [13] and feature transformation based approaches [12, 18].

Qualitative results. We present stylized results of the evaluated methods in Fig. 7 and more results in the appendix. The proposed algorithm performs favorably against the state-of-the-art methods. Although the optimization based method [8] allows universal style transfer, it is computationally expensive due to the adopted iterative optimization process (see Table 1). The fast feed-forward methods [13] performs more efficiently than the optimization based scheme, but it requires training one network for each style and adjusting style weights for best performance.
The AdaIn method presents an efficient solution for universal style transfer, but it generates less appealing results (e.g., row 2, 3 in Fig. 7) as only the first-order image statistics are used. The WCT method performs well (see column 5 in Fig 7) by modeling the second-order image statistics but at the expense of runtime (see Table 1). In contrast, our method learns the covariance-based transformation and performs favorably for arbitrary style (see the last column in Fig 7) and efficiently (see Table 1). Besides, it can be applied to other applications flexibly.

User study. We conduct a user study to evaluate the proposed algorithm against the state-of-the-art style transfer methods [8, 31, 12, 18]. We use 6 content and 40 style images to synthesize 240 stylized results, and show 15 randomly chosen content and style combinations to each subject. For each combination, we present 5 synthesized images by each method mentioned above in a random order and ask the subject to select the most visually pleasant one. We collect 540 votes from 36 users and present the percentage of votes for each method in Fig. 9(a). Overall, the proposed algorithm is favored among all evaluated methods.

Efficiency. Table 1 shows the runtime performance of all evaluated algorithms at three input image scales: 256×256, 512×512, 1024×1024. All methods listed in this table allow universal style transfer except the algorithm by Ulyanov et al. [30] (row 1). Even the slower variant of the proposed algorithm (trained to transfer content features from *relu4_1*) runs at 100 FPS and 27 FPS for 256×256 and 512×512 images, thereby making real-time style trans-

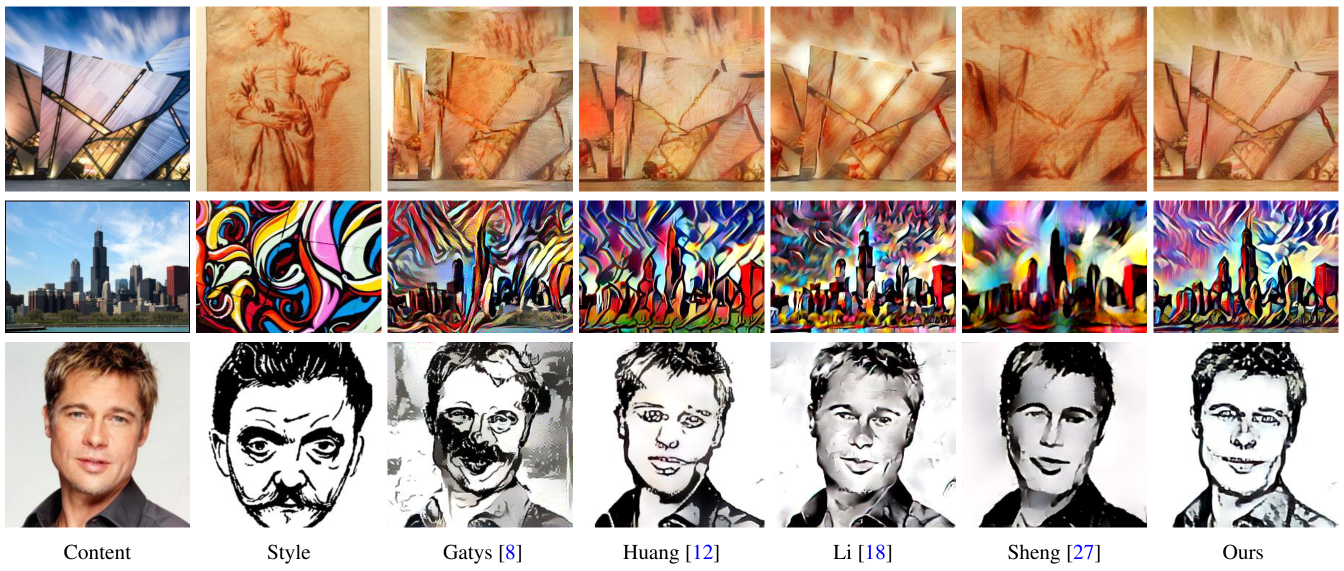

Figure 7. Stylized images by the evaluated methods. Our model is trained to transfer content features from *relu4_1* with style losses computed on *relu1_1*, *relu2_1*, *relu3_1*, *relu4_1* layer of VGG 19. All content images, as well as style images have never been seen by our model during the training process. More examples are available in the appendix.

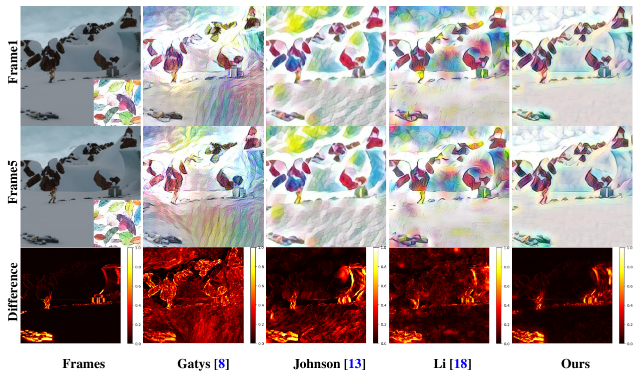

Figure 8. Video style transfer results. The first row and second row show the first frame and fifth frame with corresponding transferred frames by evaluated methods, and the last row shows the heat map of the difference between these two frames.

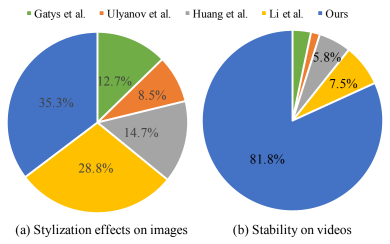

Figure 9. User study of stylization effects on images and stability on videos.

fer feasible. Our model performs more efficiently than the optimization-based method [8] (by three orders of magnitude) as well as WCT [18] (by two orders of magnitude) and comparably to the fast feed-forward schemes [30, 12].

5.4. Video Style Transfer

Video style transfer is conducted between a content video and a style image in a frame-wise manner by using a shallower auto-encoder up to the $relu3\_1$ layer. This process can be more efficient than image based style transfer, since the transformation output from the style branch can be computed only once during initialization (see Fig. 2), and directly applied to the remaining frames. Since our approach can preserve the affinity of the content images, which is naturally consistent and stable, the stylized videos are also visually stable without the need of any auxiliary techniques such as optical flow warping [11]. Fig. 8 shows the direct frame-based style transfer results by the proposed model compared with existing methods [8, 13, 18]. To visualize stability of synthesized video clip, we show heat maps of differences between two frames (i.e., row 3). The differences by the proposed algorithm are closest to that of the original frames, which suggest that our algorithm is able to preserve content affinity during style transfer. To further evaluate the stability of our algorithm in video style transfer, we conduct a user study with 5 video clips synthesized frame-wise by our algorithm and 4 state-of-the-art methods [8, 30, 12, 18]. For each group of videos, we ask each subject to select the most stable video clip. We collect 106 votes from 27 subjects and present the preference of each method in Fig. 9(b). Overall, the proposed algorithm performs well against the other methods, which indicates our approach is able to preserve affinity during style transfer.

5.5. Photo-realistic Style Transfer

As discussed in Section 3, to achieve photo-realistic style transfer, we combine our linear style transfer network with

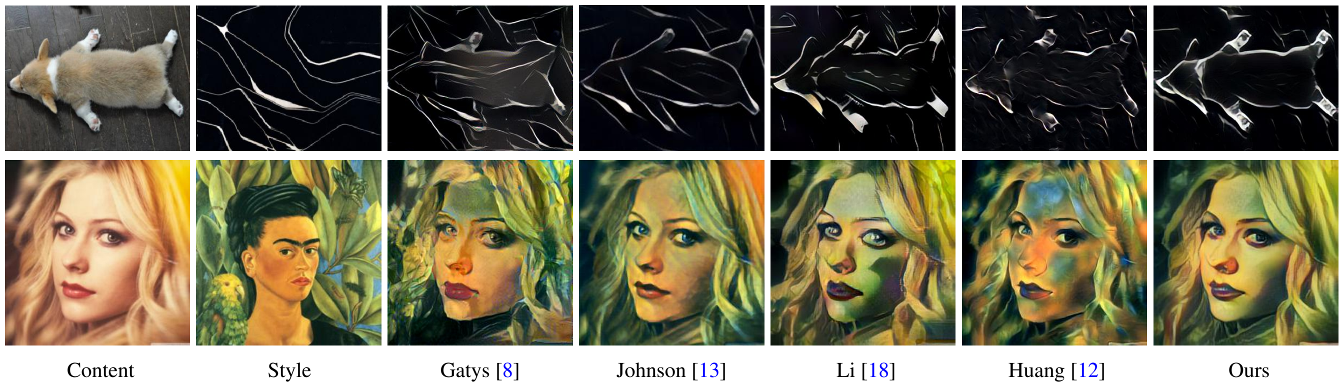

Figure 10. Affinity preserving comparison between our algorithm with [8, 13, 18, 12]. Our result has a more photo-realistic feel with the content images. It faithfully preserves the contour of the dog (row 1) and the shadow of the face (row 2) in the stylized images.

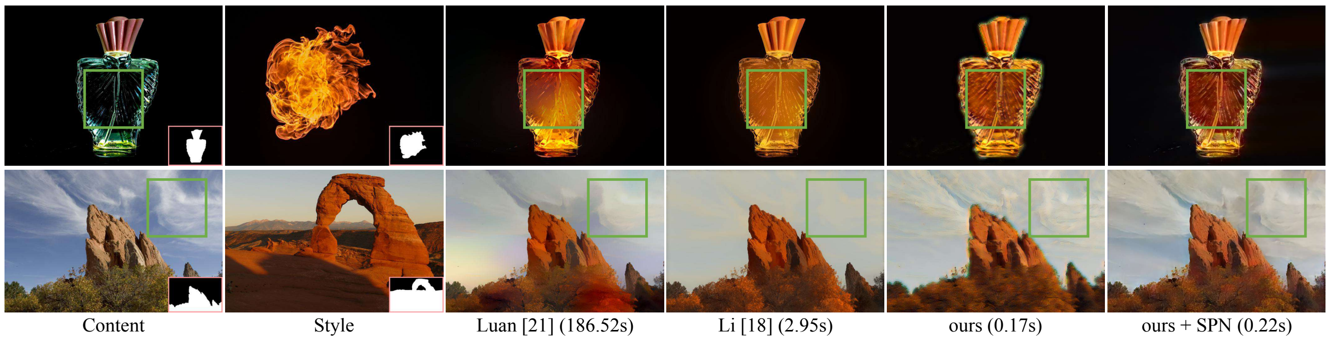

Figure 11. Photo-realistic style transfer results. The spatial mask is displayed at the right bottom corner of each content and style image. "+SPN" in the last column means results filtered by the SPN module after stylization. Inference time are tested and averaged over 60 512×256 images.

a SPN [22] to minimize distortions caused by the auto-encoder. The SPN is similar as Liu et al. [22], which contains a CNN with 8 convolution layers that outputs all local weights that formulate a pixel-affinity matrix, and a linear propagation layer that outputs the filtered images. At the training stage, we fix the style transfer encoder-decoder and feed a reconstruction image produced by the auto-encoder into the propagation module as the input. The propagation module outputs a refined reconstruction image under the guidance of the corresponding whitened image. We then compute the Euclidean loss between the refined and original images. Note that no style transfer process is involved during the training stage.

At the inference stage, we first transfer the corresponding regions with respect to a pre-defined mask for every content/style pairs. This is carried out by separating the masked regions only within the transformation module (see Fig. 2) and combining the transformed features of different regions, according to the mask. Since the inputs to the transformation module are covariance matrices, our style transfer network can take arbitrary masks. We apply pre-trained propagation module directly onto the transferred image while using its whitened content image as the input to the guidance network. We show the qualitative comparison with recent works [23, 19] in Fig. 11 along with their inference time tested on 512×256 images. As shown in the squared regions, thanks to our data-driven transformation and propagation module, the proposed approach preserves photo-realistic details better (e.g., the texture of the sky or bottle) in content images when transferring color from style images. Furthermore, our end-to-end approach is two orders of magnitude faster than [23] and one order of magnitude faster than [19].

6. Conclusions

In this work, we propose a framework for analyzing universal style transfer methods and present an effective as well as efficient algorithm by learning linear transformations. In addition, we propose a linear propagation module to enable a feed-forward network for photo-realistic style transfer. Our algorithm is computationally efficient, flexible for numerous tasks, and effective for stylizing images and videos. Experimental results demonstrate that the proposed algorithm performs favorably against the state-of-the-art methods on image and video style transfer.

Acknowledgement

This work is supported in part by the NSF CAREER Grant #1149783, and gifts from Adobe, Verisk, and NEC.

References

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

[2] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016. 10, 12

[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 5

[4] V. Dumoulin, J. Shlens, and M. Kudlur. A learned representation for artistic style. *CoRR, abs/1610.07629*, 2016. 2

[5] A. Dundar, M.-Y. Liu, T.-C. Wang, J. Zedlewski, and J. Kautz. Domain stylization: A strong, simple baseline for synthetic to real image domain adaptation. *arXiv preprint arXiv:1807.09384*, 2018. 10

[6] S. Y. Falong Shen and G. Zeng. Neural style transfer via meta networks. In *CVPR*, 2018. 2

[7] O. Frigo, N. Sabater, J. Delon, and P. Hellier. Split and match: Example-based adaptive patch sampling for unsupervised style transfer. In *CVPR*, 2016. 2

[8] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *CVPR*, 2016. 1, 2, 4, 6, 7, 8, 10, 13

[9] L. A. Gatys, A. S. Ecker, M. Bethge, A. Hertzmann, and E. Shechtman. Controlling perceptual factors in neural style transfer. In *CVPR*, 2017. 2, 6

[10] G. Ghiasi, H. Lee, M. Kudlur, V. Dumoulin, and J. Shlens. Exploring the structure of a real-time, arbitrary neural artistic stylization network. In *BMVC*, 2017. 2, 10, 13

[11] A. Gupta, J. Johnson, A. Alahi, and L. Fei-Fei. Characterizing and improving stability in neural style transfer. In *ICCV*, 2017. 3, 4, 7

[12] X. Huang and S. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, 2017. 1, 2, 3, 4, 6, 7, 8, 10, 13

[13] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016. 1, 2, 4, 6, 7, 8

[14] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICML*, 2015. 5

[15] C. Li and M. Wand. Combining markov random fields and convolutional neural networks for image synthesis. In *CVPR*, 2016. 2

[16] C. Li and M. Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *ECCV*, 2016. 2

[17] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang. Diversified texture synthesis with feed-forward networks. In *CVPR*, 2017. 2

[18] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang. Universal style transfer via feature transforms. In *NIPS*, 2017. 1, 2, 3, 4, 6, 7, 8, 10, 13

[19] Y. Li, M.-Y. Liu, X. Li, M.-H. Yang, and J. Kautz. A closed-form solution to photorealistic image stylization. *ECCV*, 2018. 1, 2, 8, 10, 11

[20] Y. Li, N. Wang, J. Liu, and X. Hou. Demystifying neural style transfer. In *IJCAI*, 2017. 10

[21] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollr. Microsoft coco: Common objects in context. In *ECCV*, 2014. 5

[22] S. Liu, S. D. Mello, J. Gu, G. Zhong, M.-H. Yang, and J. Kautz. Learning affinity via spatial propagation networks. In *NIPS*, 2017. 1, 4, 8

[23] F. Luan, S. Paris, E. Shechtman, and K. Bala. Deep photo style transfer. In *CVPR*, 2017. 2, 8, 10, 11

[24] K. Nichol. Painter by numbers, wikiart. `https://www.kaggle.com/c/painter-by-numbers`, 2016. 5

[25] 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. In *NIPS Workshop*, 2017. 5

[26] S. R. Richter, V. Vineet, S. Roth, and V. Koltun. Playing for data: Ground truth from computer games. In *ECCV*, 2016. 10

[27] L. Sheng, Z. Lin, J. Shao, and X. Wang. Avatar-net: Multi-scale zero-shot style transfer by feature decoration. In *CVPR*, 2018. 2, 7

[28] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 5

[29] B. Sun, J. Feng, and K. Saenko. Return of frustratingly easy domain adaptation. In *AAAI*, 2016. 2, 10

[30] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. S. Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *ICML*, 2016. 1, 2, 6, 7, 10, 13

[31] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In *CVPR*, 2017. 2, 6

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

[33] P. Wilmot, E. Risser, and C. Barnes. Stable and controllable neural texture synthesis and style transfer using histogram losses. *arXiv preprint arXiv:1701.08893*, 2017. 2

[34] H. Zhang and K. Dana. Multi-style generative network for real-time transfer. *arXiv preprint arXiv:1703.06953*, 2017. 2

[35] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *CVPR*, 2017. 10, 12