

Iterative Feature Transformation for Fast and Versatile Universal Style Transfer

Anonymous ECCV submission

Paper ID 3170

Abstract. The general framework for fast universal style transfer consists of an autoencoder and a feature transformation at the bottleneck. We propose a new transformation that iteratively stylizes features with analytical gradient descent. Experiments show this transformation is advantageous in part because it is fast. With control knobs to balance content preservation and style effect transference, we also show this method can switch between artistic and photo-realistic style transfers and reduce distortion and artifacts. Finally, we show it can be used for applications requiring spatial control and multiple-style transfer.

Keywords: Style transfer

1 Introduction

Style transfer is a task that renders a content image with the style from another image. Modern methods typically rely on neural networks to extract high level features for content and style representations. While achieving remarkable results, they can suffer from a *slow stylizing process* [5] or lack the ability to support *universal style transfer* [9,11,22,23,25,4,14], i.e., the ability to deal with arbitrary style images. To address these limitations, the recent trend has been to design frameworks that consist of an autoencoder that embed a feature transformation to support the style transfer [2,8,15,13,21], as shown in figure 1(A).

At present, the feature transformations used for style transfer mostly focus on transferring style. For some methods [2,8,21], this leads to a distorted content in the resulting stylized images. This limits their applicability to scenarios when a user is performing artistic style transfer. In contrast, some methods are better suited to preserve the content and so produce photo-realistic transfer results [15,13]. Still, these methods can result in unnecessary artifacts and distortions in detailed content as initially examined in prior work [12,19,17] and expanded upon in our experiments.

In this paper, we propose an iterative feature transformation. We conduct experiments to show it realizes fast universal style transfer that can preserve content. More generally, this transformation provides control knobs to change the amount of style effect transference, enabling it to switch between artistic and photo-realistic style transfers. We also demonstrate that our method is versatile in that it can be applied for spatial control and (non-linear) multi-style transfer.

045 2 Related Works

046 A summary of how our work differs from prior work is shown in table 1. We elaborate
 047 in this section on the details conveyed in this table, and how our approach
 048 is uniquely able to meet the many interests of the style transfer community.

051 **Table 1:** Shown is a summary of the differences between our proposed approach and
 052 existing style transfer methods. The first four items are general characteristics of a
 053 method, while the last four exemplify versatility. Our method realizes every listed
 054 item. (**Ava:** Avatar-net. **SS:** Style swap. **Ada:** AdaIN. **Learn-free:** Style-learning-free.
 055 **Balance:** Content-style balance. **Art:** For artistic style transfer only but no photo-
 056 realistic transfer. **NL/L:** Non-linear/Linear fashion.) * [13] does not mention if LST
 057 realizes multi-style transfer, but we believe it can as indicated in the table.

	NST [5]	WCT [15]	Ava [21]	SS [2]	Ada [8]	LST [13]	Ours
Universal	✓	✓	Art	Art	Art	✓	✓
Learn-free	✓	✓	✓				✓
Fast		✓	✓	✓	✓	✓	✓
Balance	✓						✓
Artistic	✓	✓	✓	✓	✓	✓	✓
Photo-realistic	✓	✓				✓	✓
Spatial control	✓	✓	✓		✓	✓	✓
Multi transfer	NL	L	L		L	L*	NL/L

068 *Neural style transfer.* Gatys et al. [5] proposed a method of Neural style transfer
 069 (NST) which has opened a new era of using a deep neural network to extract
 070 content and style representations of images, based on which we can synthesize
 071 an image whose content and style are from two distinct images. Variants of this
 072 algorithm include adding a Markov random field as a regularizer to reduce arti-
 073 facts [10] and introducing extra histogram losses to help improve quality and
 074 convergence [20]. Neural style transfer also allows fine-grained control such as
 075 spatial/semantic control [1,6]. The drawback of Gatys’s framework [5] and vari-
 076 ants is that solving for stylized images is time-consuming due to many iterations
 077 of feed-forward and back-propagation to adjust pixel values. In the experiments,
 078 we will demonstrate our approach leads to considerable speed ups and so ad-
 079 dresses the limitation that NST is very slow.

081 *Feed-forward style networks.* Numerous methods address the issue that NST is
 082 slow by training feed-forward networks [9,11,22,23] on a pre-defined set of style
 083 images to reduce the content and style losses used in [5]. In testing time, content
 084 images can then be stylized with learned styles fast or even in real time for images
 085 that are not high resolution in one forward pass. However, these algorithms did
 086 so at the cost of limiting how many styles they can transfer to a few styles up to
 087 1,000 styles [25,4,14]. In other words, this adaptation leads to a limitation that
 088 such methods do not support universal style transfer. Our work, in contrast,
 089 supports universal style transfer.

Universal style transfer. Chen and Schmidt [2] introduce a framework of an autoencoder with a “Style swap” feature transformation at the bottleneck, with the transformation the key behind it being able to learn to generalize to unseen styles and virtually realize universal style transfer after trained on a large corpus of 80k paintings. Other works [7], with the most recent being AdaIN [8] and Linear Style Transfer (LST) [13], follow the same idea of training on a large dataset of style images but use different transform mechanisms to achieve universality. In contrast, style-learning-free methods, including whitening and coloring transform (WCT) [15] and Avatar-net [21], realize universal artistic style transfer without the need to learn from style images. In our experiments, we compare our method to five modern universal style transfer methods to reveal its advantage in being able to support a larger range of tasks, also often more effectively than existing more constrained methods. It does so without pre-training.

Photo-realistic transfer. Compared to artistic transfer, photo-realistic transfer is more constrained in that the results need to look convincingly realistic to an end user. NST and its variant [18] pioneered the use of neural networks for photo-realistic transfer. Among the aforementioned autoencoder-based methods for universal style transfer, the transformations in WCT and LST are applicable [16,13], while other transformations [2,8,21] are only suitable for artistic transfer. Our experiments demonstrates an advantage of our transformation for photo-realistic transfer in balancing content preservation and style transfer, with it considerably reducing artifacts and distortion compared to WCT and LST.

3 Method

In this section, we show how we derive our new transformation for style transfer. Throughout the derivation, we will explain why this transformation is GPU-friendly, how it balances content information and style effect, and how it can be employed for the applications of spatial control and multi-style transfer.

3.1 General Framework - Background

The general framework for fast style transfer consists of an autoencoder (i.e., an encoder-decoder pair) and a feature transformation at the bottleneck, as shown in figure 1(A). An encoder first extracts features from content and style images, features are transformed by the transformation method, and a transformed feature is mapped to an image by the decoder. Transferring style features from multiple layers can be done by a cascade of autoencoders as in WCT [15]. A standard approach pioneered by Gatys et al. with NST[5] is to use as the *content information* of the image a feature map of the image that comes from a higher layer of VGG network and use as the *style information* of an image the Gram matrices of feature maps from different layers in VGG network. Formally, the set-up consists of *relu4_1* layer for content feature extraction and *relu1_1*, *relu2_1*, *relu3_1*, and *relu4_1* layers for style feature extraction. We denote the reshaped

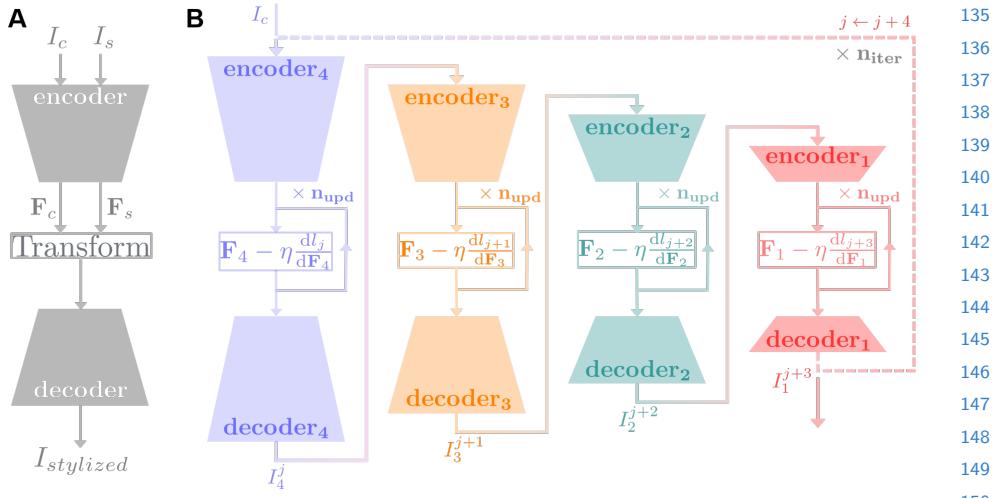


Fig. 1: (A) General framework for universal style transfer: an encoder extracts features \mathbf{F}_c and \mathbf{F}_s from content and style images I_c and I_s respectively. \mathbf{F}_c is transformed with reference to \mathbf{F}_s and then decoded to a stylized image $I_{stylized}$. (B) Schematic diagram of our style transfer algorithm by analytical gradient descent. Style image I_s is suppressed for clarity and style information resides in the gradients $\frac{dI}{d\mathbf{F}}$'s. Content image I_c is first encoded as a feature \mathbf{F}_4 . \mathbf{F}_4 is gradually stylized by our new iterative transformation with gradient descent and then decoded to a coarsely stylized image I_4^j , which is treated as a new content image to the next autoencoder for finer stylization. The same procedure applies iteratively until the last autoencoder to derive the final result I_1^{j+3} . If more style effect is wanted, a user can repeat from the first autoencoder by setting I_1^{j+3} as the content.

feature map of the image I from the layer $reluN_1$ as $\mathbf{F}_N(I) \in \mathbb{R}^{C_N \times H_N(I)W_N(I)}$, where $H_N(I)$, $W_N(I)$, and C_N are height, width, and channel length of the layer output. Suppose I_c and I_s are the content and style images, respectively, and $\mathbf{F}_N(I_c) \triangleq \mathbf{F}_{N,c}$ and $\mathbf{F}_N(I_s) \triangleq \mathbf{F}_{N,s}$. Then the stylized image I of the same size of I_c can be derived by solving the following optimization problem:

$$\min_I \underbrace{\|\mathbf{F}_4(I) - \mathbf{F}_{4,c}\|_F^2}_{\text{content loss}} + \underbrace{\sum_{N=1}^4 \lambda_N \left\| \frac{1}{n_N} \mathbf{F}_N(I) \mathbf{F}_N(I)^T - \frac{1}{m_N} \mathbf{F}_{N,s} \mathbf{F}_{N,s}^T \right\|_F^2}_{\text{style loss}}, \quad (1)$$

where λ_N 's are weights between content and style losses, n_N and m_N equal to $H_N(I)W_N(I)$ and $H_N(I_s)W_N(I_s)$, respectively.

3.2 New iterative transformation with analytical gradient descent

Our novelty lies in approximating the solution to the NST [5] objective by embedding a new transformation that iteratively updates features in the cascade of

180 four autoencoders using analytical gradient descent. An overview of our method
 181 is illustrated in figure 1(B).

182 We introduce our approach to address the limitation that Equation 1 cannot
 183 be analytically solved and optimization requires gradient descent with back-
 184 propagation, which is time consuming. We borrow the idea of alternating mini-
 185 mization used in convex optimization to speed up and approximately solve this
 186 equation. In alternating minimization of a function $f(x, y)$, we first fix y to an
 187 initial value y_0 and optimize over x to x_1 , then we optimize over y to y_1 with
 188 x fixed to x_1 , and repeat, alternating until convergence. By analogy, we first
 189 optimize for \mathbf{F}_4 in equation 1:

$$190 \min_{\mathbf{F}_4} \|\mathbf{F}_4 - \mathbf{F}_{4,c}\|_F^2 + \lambda_4 \left\| \frac{1}{n_4} \mathbf{F}_4 \mathbf{F}_4^T - \frac{1}{m_4} \mathbf{F}_{4,s} \mathbf{F}_{4,s}^T \right\|_F^2. \quad (2)$$

193 Suppose the value of \mathbf{F}_4 found by solving equation 2 is $\mathbf{F}_4^{(1)}$, which is the feature
 194 map of image $I_4^{(1)}$ from layer *relu4_1*. Note that this image can be derived by
 195 exploiting an autoencoder as in [15]. Since feature maps from different layers
 196 are coupled through the image $I_4^{(1)}$, we also have feature maps $\mathbf{F}_N^{(1)} = \mathbf{F}_N(I_4^{(1)})$,
 197 $N = 3, 2, 1$. Due to the coupling, fixing \mathbf{F}_4 , \mathbf{F}_2 , and \mathbf{F}_1 to $\mathbf{F}_4^{(1)}$, $\mathbf{F}_2^{(1)}$, and $\mathbf{F}_1^{(1)}$,
 198 respectively, can be done by fixing \mathbf{F}_3 to $\mathbf{F}_3^{(1)}$. Therefore, in the next step where
 199 we optimize for \mathbf{F}_3 with other feature maps fixed, we solve the following problem:
 200

$$202 \min_{\mathbf{F}_3} \|\mathbf{F}_3 - \mathbf{F}_3^{(1)}\|_F^2 + \lambda_3 \left\| \frac{1}{n_3} \mathbf{F}_3 \mathbf{F}_3^T - \frac{1}{m_3} \mathbf{F}_{3,s} \mathbf{F}_{3,s}^T \right\|_F^2, \quad (3)$$

203 where the first term captures the concept of the hard fixation $\mathbf{F}_3 = \mathbf{F}_3^{(1)}$ that
 204 is relaxed to a soft proximity. Let $\mathbf{F}_3^{(2)} \triangleq \mathbf{F}_3(I_3^{(2)})$ be the result of optimization
 205 and $\mathbf{F}_2^{(2)} = \mathbf{F}_2(I_3^{(2)})$. Similarly, we can solve for \mathbf{F}_2 in equation 1 with extra soft
 206 proximity $\|\mathbf{F}_2 - \mathbf{F}_2^{(2)}\|_F^2$. In general, with $\mathbf{F}_4^{(0)} \triangleq \mathbf{F}_{4,c}$ the objective of each step
 207 is as follows:

$$211 \min_{\mathbf{F}_N} l_j(\mathbf{F}_N) = \min_{\mathbf{F}_N} \underbrace{\|\mathbf{F}_N - \mathbf{F}_N^{(j)}\|_F^2}_{\text{soft proximity loss}} + \lambda_N \underbrace{\left\| \frac{1}{n_N} \mathbf{F}_N \mathbf{F}_N^T - \frac{1}{m_N} \mathbf{F}_{N,s} \mathbf{F}_{N,s}^T \right\|_F^2}_{\text{style loss}}. \quad (4)$$

215 We alternate between the minimization of \mathbf{F}_N 's until convergence.

216 To incorporate soft proximity, we use gradient descent to solve the optimiza-
 217 tion problem in equation 4.¹ Unlike solving equation 1 that uses back-
 218 propagation to compute the gradients, the gradient $\frac{d l_j(\mathbf{F}_N)}{d \mathbf{F}_N}$ has an analytical
 219 form, and the optimization process can be done fast using GPU acceleration.

221 ¹ Of note, previous analysis [3] shows that the feature \mathbf{F}_{wct} derived by applying WCT
 222 to $\mathbf{F}_{N,c}$ and $\mathbf{F}_{N,s}$ makes the value of the style loss in equation 4 go to zero, and
 223 hence could serve as an approximate solution. However, WCT does not consider the
 224 balance between soft proximity loss and style loss.

Specifically, the gradient can be written in the following form (detailed derivation is given in Supplementary Material):

$$\frac{dl_j}{d\mathbf{F}_N} = 2(\mathbf{F}_N - \mathbf{F}_N^{(j)}) + \frac{4\lambda_N}{n_N} \left(\frac{1}{n_N} \mathbf{F}_N \mathbf{F}_N^T - \frac{1}{m_N} \mathbf{F}_{N,s} \mathbf{F}_{N,s}^T \right) \mathbf{F}_N, \quad (5)$$

and the gradient descent updates \mathbf{F}_N by repeating the following update rule n_{upd} times:

$$\mathbf{F}_N \leftarrow \mathbf{F}_N - \eta \frac{dl_j}{d\mathbf{F}_N}, \quad (6)$$

where η is the learning rate. **Equations 5 and 6 together form the iterative feature transformation method.** We can observe that the computation of it is a mix of matrix multiplication and matrix addition, and thus is GPU-friendly.

The whole style transfer algorithm by analytical gradient descent is summarized in figure 1. The cascade of four autoencoders is the same structure as used in WCT style transfer algorithm [15]. Here $encoder_N$ denotes the part of the VGG network from the input layer to $reluN_1$ layer. The structure of $decoder_N$ is symmetrical to $encoder_N$ and implements an inverse function of $encoder_N$. At the bottleneck of each autoencoder, the feature map updates n_{upd} times, whose value can be decided by trial and error. If more style effect is wanted, we can iterate over the autoencoder cascade for multiple rounds, say n_{iter} times². These two parameters n_{upd} and n_{iter} together with the learning rate η and the weights λ_N 's are the four knobs that control how much style effect to be transferred.

3.3 Applications that demonstrate the versatility of our method

Spatial control. In spatial control of style transfer [6], a content image I_c and a style image I_s are segmented into regions. The r -th region of the content image is stylized with the corresponding r -th region in the style image (exemplified in figure 4). Suppose the set $\mathcal{S}_{N,c}^r$ of indices $\{r_{N,c}^1, r_{N,c}^2, \dots\}$ indicates the columns of feature $\mathbf{F}_{N,c}$ that correspond to the pixels in the r -th region of I_c , and similarly we have $\mathcal{S}_{N,s}^r$ for $\mathbf{F}_{N,s}$ and I_s . Let $\mathbf{F}_N^r \in \mathbb{R}^{C_N \times |\mathcal{S}_{N,c}^r|}$ be $\mathbf{F}_N[:, \mathcal{S}_{N,c}^r]$ and $\mathbf{F}_{N,s}^r \in \mathbb{R}^{C_N \times |\mathcal{S}_{N,s}^r|}$ be $\mathbf{F}_{N,s}[:, \mathcal{S}_{N,s}^r]$, where $|\mathcal{S}|$ is the size of \mathcal{S} . To realize spatial control with our style transfer method, we modify equation 5 as follows:

$$\frac{dl_j}{d\mathbf{F}_N^r} = 2(\mathbf{F}_N^r - (\mathbf{F}_N^{(j)})^r) + \frac{4\lambda_N}{n_N} \left(\frac{1}{|\mathcal{S}_{N,c}^r|} \mathbf{F}_N^r (\mathbf{F}_N^r)^T - \frac{1}{|\mathcal{S}_{N,s}^r|} \mathbf{F}_{N,s}^r (\mathbf{F}_{N,s}^r)^T \right) \mathbf{F}_N^r, \forall r, \quad (7)$$

and $\frac{dl_j}{d\mathbf{F}_N}$ is the collection of $\frac{dl_j}{d\mathbf{F}_N^r}$'s.

Multiple-style transfer. Our algorithm can be easily extended from single-style transfer to multiple-style transfer (exemplified in figure 5). In particular, if now

² According to our experience, $n_{iter} = 3$ is enough for convergence and does not differ too much from $n_{iter} = 2$.

we have q style images $I_{s,1}, I_{s,2}, \dots, I_{s,q}$ with feature maps $\mathbf{F}_{N,s}^1, \mathbf{F}_{N,s}^2, \dots, \mathbf{F}_{N,s}^q$, the optimization problem in equation 4 should be modified as

$$\min_{\mathbf{F}_N} \|\mathbf{F}_N - \mathbf{F}_N^{(j)}\|_F^2 + \sum_{k=1}^q \lambda_N^k \left\| \frac{1}{n_N} \mathbf{F}_N \mathbf{F}_N^T - \frac{1}{m_N^k} \mathbf{F}_{N,s}^k (\mathbf{F}_{N,s}^k)^T \right\|_F^2. \quad (8)$$

Unlike equation 4 where λ_N controls only the balance between content and style losses, λ_N^k 's in equation 8 are also the weights between different style effects. We provide the derivation of the gradient equation to solve this objective in the Supplementary Materials

4 Experiments

In the experiments, we first validate that our transformation is GPU-friendly.³ To do so, we show our approach is the fastest among modern universal transfer methods that do not require pre-training. Our next experiments demonstrate that the knobs controlling the balance between content preservation and style effect transfer benefit photo-realistic style transfer by reducing artifacts and distortion. Finally, we demonstrate the versatility of our method by showing it can be used for two additional applications.

4.1 Single-style transfer

We first demonstrate how our method compares to existing style transfer methods in terms of speed. We also show qualitative results.

Baselines. We evaluate two groups of methods that support universal transfer style transfer. Their properties are summarized in table 1.

One subset of existing methods requires pre-training on style images: Style swap [2], AdaIN [8], and LST [13]. Style swap is the first to introduce the framework of an autoencoder with an embedded feature transform to realize universal style transfer. To achieve faster speed, AdaIN replaces the computationally expensive style swap layer with a lightweight transformation that matches variances between style images and stylized images. Furthermore, LST introduces a transformation that aims to match covariances.

Our method is more similar to the second group which are methods that can perform style transfer *without pre-training*. These methods include: NST [5], WCT [15], and Avatar-net [21]. NST is the first deep learning method that realizes style transfer. WCT is the first style-learning-free method for fast style transfer. For WCT, we use the version of a cascade of four autoencoders due to its better performance [16]. Avatar-net introduces a style decorator that combines the transforms of WCT and Style swap.

³ The training details of the autoencoders are provided in the Supplementary Materials.

Implementation details for our method. We use the following hyperparameter setting for our algorithm for artistic style transfer: learning rate $\eta = 0.01$, weights $\{\lambda_1, \lambda_2, \lambda_3, \lambda_4\} = \{10^4, 10^4, 10^3, 10^2\}$, number of update $n_{upd} = 20$, and number of iteration $n_{iter} = 2$.

Experimental Design. We benchmarked speed performance for each method on an Nvidia GTX 1080 Ti with 11 GB memory. To enrich our analysis, we analyzed speed across four different resolutions for the content image: 256×256 , 512×512 , 768×768 , and 1024×1024 . This is interesting in part because it can capture the common image resolutions for the two use cases we study: photo-realistic and artistic transfer. While photo-realistic transfer typically is conducted with higher resolution images, artistic transfer supports the range of resolutions from low to high. This is also interesting because it captures the limitations of some methods in handling the full range of image resolutions. The size of style images are fixed to 512×512 . We conducted experiments with 20 style images each applied to 5 content images. Elapsed time in seconds is estimated by averaging the time from 100 experimental results.

Table 2: Speed performance of universal style transfer methods. Four different sizes of content images are considered. For WCT and our algorithms, we report two values for $n_{iter} = 1$ and $n_{iter} = 2$ (in parentheses). The former value transfers less of the style, which is better-suited for photo-realistic transfer. The latter value transfers more style, which is better-suited for artistic transfer. Results show our method is consistently faster than all baselines which similarly do not require pre-training. **OOM:** Out of memory. **Unit:** Second.

Content image size	No pre-training				Pre-training		
	NST	WCT	Avatar	Ours	Style swap	AdaIN	LST
256×256	15.28	1.15 (2.31)	0.96	0.15 (0.28)	0.55	0.036	0.04
512×512	33.29	1.24 (2.47)	1.08	0.33 (0.65)	1.89	0.046	0.05
768×768	64.50	1.38 (2.69)	1.28	0.64 (1.29)	4.29*	0.068	0.10
1024×1024	108.94	1.63 (3.28)	1.54	1.07 (2.15)	OOM	0.105	0.15

* average over 3 results due to out-of-memory error midway

Speed Performance Results. Results are shown in table 2.

As shown, our method is consistently faster than all baselines which similarly do not require pre-training. For example, our method is approximately 100 times faster than NST and approximately 3-10 times faster than WCT and Avatar-net. We attribute the slowest speed observed from NST to it using gradient descent with back-propagation to solve the equation 1. While our method consistently outperforms all these methods, we observe that the gains can fall with higher resolution images. Specifically, at the largest resolution of 1024x1024, we observe our method is roughly 1.5 times faster than WCT and Avatar-net. We attribute

this to the repeated computation of $\mathbf{F}_N \mathbf{F}_N^T$ in equation 5 with large size of \mathbf{F}_N . While our advantage is still shown over existing methods, further speed-ups could be achieved by increasing the learning rate η and accordingly decreasing the number of update n_{upd} . Another interesting observation is that Avatar-net works slightly faster than WCT, despite the fact that both avoid any pre-training. This is because Avatar-net only exploits one autoencoder composed of $encoder_4$ and $decoder_4$. The drawback of Avatar-net though is that the style transfer majorly results from *relu4_1* style feature, while the features from *relu1_1*, *relu2_1*, and *relu3_1* layers are marginally transferred.

Compared to the methods which require pre-training, our method is both slightly better and slightly worse. For example, we observe that Style swap is both slower and cannot handle large content images; e.g., at least roughly three times slower. In contrast, we observe speed gains for AdaIN and LST. We attribute this advantage to their training on a large dataset of style images and so embedding knowledge of styles in the values of the learned model parameters. However, this pre-training brings a downside in that users have less flexible control on the transfer effect which can lead to distorted content. We quantify this disadvantage below in Section 4.2.

Qualitative Style-Transfer Results. Examples of style transfer results are shown in figures 2 and 3 for artistic style transfer and photo-realistic style transfer respectively. Of note, only a subset of the methods support photo-realistic style transfer: NST, WCT, LST, and ours. As shown, like prior methods, our approach can produce visually-pleasing style transfer results.

Qualitatively, the strength of our approach is more evident for the photo-realistic results, which is the setting where the content must be convincing that it is plausibly real. We observe that our method can preserve content well while NST, WCT, and LST suffer from some artifacts and distorted content. In the first example of the canal, NST introduces extra light strokes, WCT produces unpleasant artifacts in the clouds and their reflection on the water, and LST does not hold content well (as shown in the zoomed-in picture). In the second example of the monastery, we can clearly see in the result of our method the boundaries on the brick wall and the pattern and the reflection on the ground. These are ruined to some extent in the results from other methods. Furthermore, LST shows no reflection of light on the river in the third example. In the next section, we quantify this disadvantage.

4.2 Photo-realistic style transfer - quantitative analysis

We aim to show that our method can preserve content better/produce fewer artifacts than existing methods while still transferring style effects.

Baselines. We compare our method with NST, WCT and LST that have been considered for photo-realistic style transfer. For the other related methods, Avatar-net, Style swap and AdaIN are prone to distort content much and hence only



Fig. 2: Comparison of artistic style transfer for universal style transfer algorithms. **Green panel:** methods that do not require pre-training: NST [5], WCT [15], Avatar net [21], and ours. **Orange panel:** methods that require pre-training: Style swap [2], AdaIN [8], and LST [13]. Like prior methods, our approach can produce visually-pleasing style transfer results.

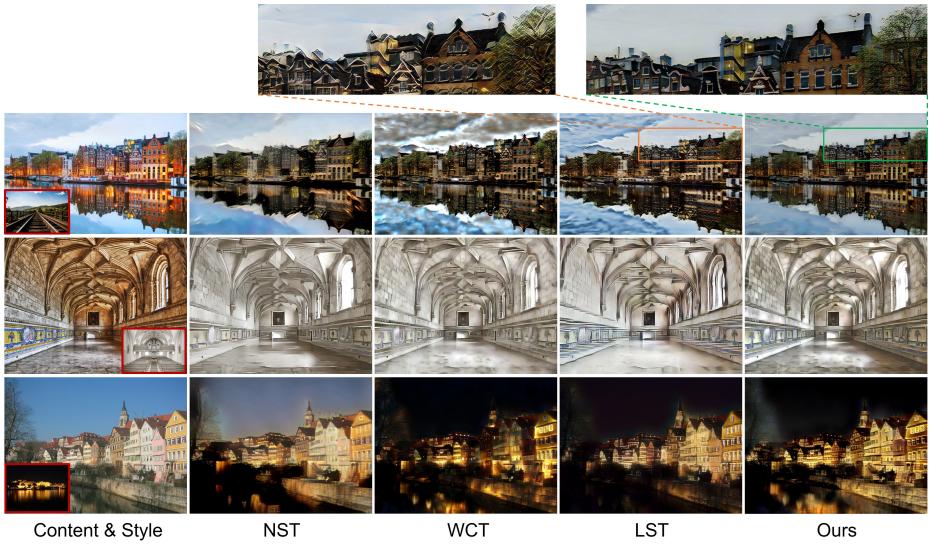


Fig. 3: Photo-realistic style transfer results for our algorithm and NST [5], WCT [15], and LST [13]. Note that no post-processing and no spatial control is applied. Results show that our method can preserve content better and cause fewer artifacts (discussed in the text).

suitable for artistic style transfer. NST was the first to show the possibility of applying deep neural networks for photo-realistic style transfer. LST shows that a network pre-trained on paintings is also applicable to real photos as style images. WCT is more widely studied: PhotoWCT [16] makes stylized images from WCT more realistic by post-processing to reduce artifacts. WCT2 [24] embeds WCT in an autoencoder that better preserves content using wavelet transform. Here we focus on the effect from feature transformations rather than post-processing and fancy autoencoders. Therefore, only vanilla autoencoders are used for WCT, LST, and our method and we do not apply post-processing in the experiment.⁴

Implementation details for our method. In photo-realistic style transfer, we need to avoid transferring too much style effect to prevent artifacts and distorted content. Therefore, we use the same values of η and λ_N 's as in artistic style transfer but reduce the values n_{upd} and n_{iter} to 15 and 1.⁵

Experimental design. We generated 30 stylized images from 30 pairs of a real content image and real style image (shown in the Supplementary Materials). To quantify the amount of distortion, we calculate SSIM (structural similarity) and FSIM (feature similarity) values between content images and stylized images in

⁴ Note that a fancy autoencoder, such as WCT2 [24] using wavelet pooling, can further prevent distortion. However, it is not the focus of this paper.

⁵ For completeness, we also report parallel time analysis for $n_{upd} = 15$ and $n_{iter} = 1$ to that reported in the previous section. Due to space constraints, we show results in the Supplementary Materials. Outcomes are similar.

495 grayscale. SSIM is the product of three comparison measurements: luminance,
 496 contrast, and structure, where the structure measurement directly captures the
 497 level of distortion. We report the values of both SSIM and the structure index.
 498 FSIM is based on two feature maps: phase congruency and gradient magnitude,
 499 both of which evaluate local structure similarity from different angles.

500
 501 **Table 3:** Evaluation of distortion level in photo-realistic style transfer. For SSIM,
 502 both SSIM values and structure indices (in parentheses) are reported. Among three
 503 autoencoder-based algorithms, our feature transformation results in highest scores in
 504 both SSIM and FSIM metrics, preserving content best.

	NST	Autoencoder-based		
		WCT	LST	Ours
SSIM	0.5644 (0.7567)	0.4476 (0.7265)	0.4429 (0.6919)	0.5301 (0.7513)
FSIM	0.7870	0.7643	0.7658	0.8197

510
 511 *Results.* As mentioned previously, our transformation balances between content
 512 preservation and style effect transferal, unlike WCT and LST investing mostly
 513 in the latter and thus distorting content more. This is justified in the results
 514 shown in table 3, where NST sets a reference line for other three fast methods.
 515 As shown, NST gets the highest score in SSIM. Our method outperforms other
 516 autoencoder-based methods and has a score close to that of NST. Besides, our
 517 method gets the highest score in FSIM metric. These imply that our feature
 518 transformation preserves content better than WCT and LST.

519 We also show stylized images in the Supplementary Materials to demonstrate
 520 that style is transferred by our method.

522 4.3 Other applications

524 To highlight the versatility of our method, we also demonstrate that our method
 525 can be applied for style transfer with spatial control and multi-style transfer.

527 *Style Transfer Results Using Spatial Control.* Results for style transfer with
 528 spatial control are shown in figure 4. Content images and style images are
 529 segmented into regions denoted by different colors. A segment in a content image
 530 is stylized with the style from the segment denoted the same color in a style
 531 image. This captures that our work is suitable for the spatial control task.

533 *Multi-Style Transfer.* Among the universal style transfer methods considered,
 534 only NST and our algorithm can mix multiple styles in a non-linear fashion,
 535 while AdaIN, Avatar-net, and WCT blend different styles by linear interpolation
 536 of transformed features. It is not explicitly mentioned in the original paper of
 537 NST [5], but extending NST to multiple-style transfer is straightforward. We
 538 describe these details as well as how to use WCT or multi-style transfer in the
 539 Supplementary Materials.

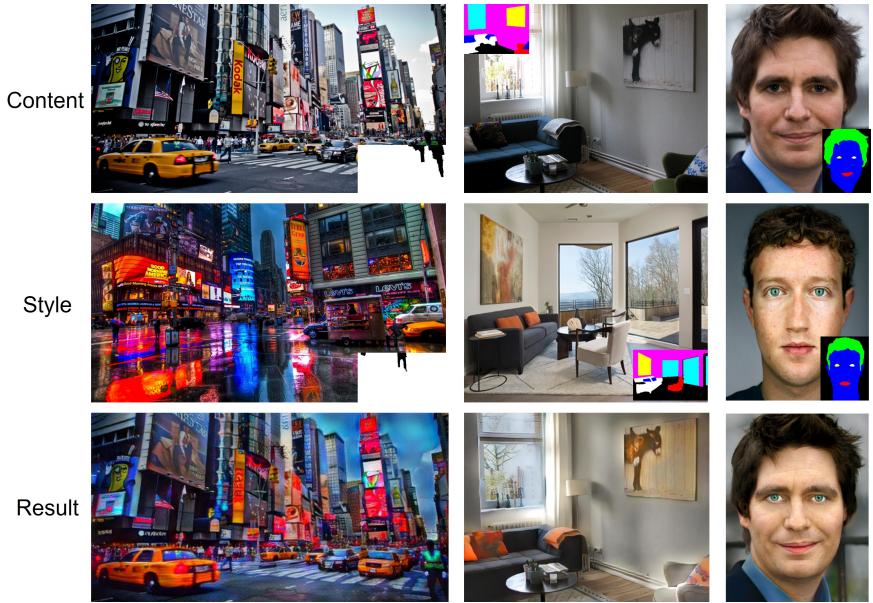


Fig. 4: Style transfer with spatial control with our algorithm. Style from a segment in a style image is transferred to the segment denoted the same color in a content image. Results show that with slight re-formulation our method can support spatial control.

We show three examples of double-style transfer using NST, WCT, and our methods in figure 5⁶. In each example, the weights for two styles are equal. By observing the change of color, we can tell that each double-style transfer result from WCT is somewhat an average of two single-style transfer results: the dark grey sky in the first example is the mean of light grey and black skies, the light green tint in the second example is the mean of dark green and light grey, and the orange leaves in the third example is the mean of red and green leaves. Moreover, in the second example the effect of sketch is covered by that of the painting of shipwreck and is barely told.

In contrast, our method preserves each style in the double-style transfer results. In the first example, the glowing and dark effect is in the sky and the ink stroke effect is on the buildings. The style of the painting of shipwreck in the second example is on the buildings and their reflection on the sea, and the sketch style can be found in the sky and the sea. In the third example, the leaves overhead are the mixture of red and green leaves, and more green leaves can be noted on the wall. NST, on the other hand, also produces non-linear effects, but might result in an unsatisfactory outcome, for instance, the third example.

⁶ Due to limited space and a similar trend of linear transition from one style to the other, we show the results from AdaIN and Avatar-net in Supplementary Material but not in main context.

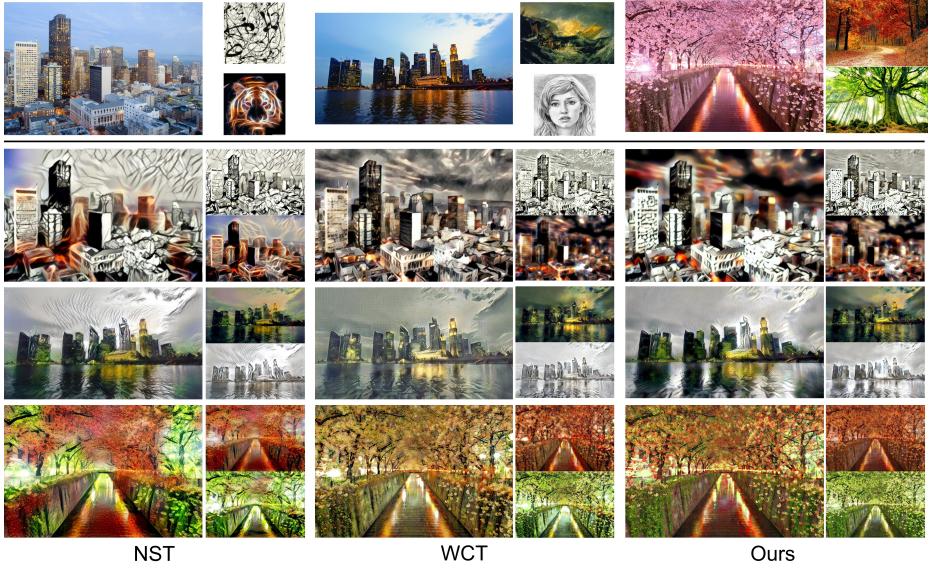


Fig. 5: Results of double-style transfer using NST [5], WCT [15], and our methods. **Upper:** Three sets of a content image and two style images. **Lower:** Smaller stylized images are single-style transfer results from each style image. Larger images are the results of double-style transfer. Our method preserves the integrity of each style better than WCT and has more pleasant results than NST.

5 Conclusion

We introduce an iterative feature transformation for universal style transfer that arises from approximating solution to the objective of NST. Our method provides control knobs that balance content preservation and style effect transferal and help switch between artistic and photo-realistic style transfers. For versatility, we show that it supports semantic control and multiple-style transfer. We show the effectiveness of our method by comparing with NST and the representative transformations for fast universal style transfer.

Future extensions of our work include an alternative feature update rule and a compact network. For example, the vanilla gradient descent here might be replaceable with a steepest gradient descent. If possible, the learning rate can be dynamically determined and the number of update can be reduced for faster speed. On the other hand, theoretically the cascade of the four autoencoders used here should be able to be converted to an equivalent but compact autoencoder. If it works in practice, large input images and faster speed are feasible.

Acknowledgements

We gratefully acknowledge funding from Anonymous.

630 References

- 631 1. Champandard, A.J.: Semantic style transfer and turning two-bit doodles into fine
632 artworks. arXiv preprint arXiv:1603.01768 (2016)
- 633 2. Chen, T.Q., Schmidt, M.: Fast patch-based style transfer of arbitrary style. arXiv
634 preprint arXiv:1612.04337 (2016)
- 635 3. Chiu, T.Y.: Understanding generalized whitening and coloring transform for uni-
636 versal style transfer. In: Proceedings of the IEEE International Conference on Com-
637 puter Vision. pp. 4452–4460 (2019)
- 638 4. Dumoulin, V., Shlens, J., Kudlur, M.: A learned representation for artistic style.
639 arXiv preprint arXiv:1610.07629 (2016)
- 640 5. Gatys, L.A., Ecker, A.S., Bethge, M.: A neural algorithm of artistic style. arXiv
641 preprint arXiv:1508.06576 (2015)
- 642 6. Gatys, L.A., Ecker, A.S., Bethge, M., Hertzmann, A., Shechtman, E.: Controlling
643 perceptual factors in neural style transfer. In: Proceedings of the IEEE Conference
644 on Computer Vision and Pattern Recognition. pp. 3985–3993 (2017)
- 645 7. Ghiasi, G., Lee, H., Kudlur, M., Dumoulin, V., Shlens, J.: Exploring the struc-
646 ture of a real-time, arbitrary neural artistic stylization network. arXiv preprint
647 arXiv:1705.06830 (2017)
- 648 8. Huang, X., Belongie, S.: Arbitrary style transfer in real-time with adaptive instance
649 normalization. In: Proceedings of the IEEE International Conference on Computer
650 Vision. pp. 1501–1510 (2017)
- 651 9. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer
652 and super-resolution. In: European conference on computer vision. pp. 694–711.
653 Springer (2016)
- 654 10. Li, C., Wand, M.: Combining markov random fields and convolutional neural net-
655 works for image synthesis. In: Proceedings of the IEEE Conference on Computer
656 Vision and Pattern Recognition. pp. 2479–2486 (2016)
- 657 11. Li, C., Wand, M.: Precomputed real-time texture synthesis with markovian gener-
658 ative adversarial networks. In: European conference on computer vision. pp. 702–
659 716. Springer (2016)
- 660 12. Li, P., Zhao, L., Xu, D., Lu, D.: Optimal transport of deep feature for image style
661 transfer. In: Proceedings of the 2019 4th International Conference on Multimedia
662 Systems and Signal Processing. pp. 167–171 (2019)
- 663 13. Li, X., Liu, S., Kautz, J., Yang, M.H.: Learning linear transformations for fast im-
664 age and video style transfer. In: Proceedings of the IEEE Conference on Computer
665 Vision and Pattern Recognition. pp. 3809–3817 (2019)
- 666 14. Li, Y., Fang, C., Yang, J., Wang, Z., Lu, X., Yang, M.H.: Diversified texture syn-
667 thesis with feed-forward networks. In: Proceedings of the IEEE Conference on
668 Computer Vision and Pattern Recognition. pp. 3920–3928 (2017)
- 669 15. Li, Y., Fang, C., Yang, J., Wang, Z., Lu, X., Yang, M.H.: Universal style transfer
670 via feature transforms. In: Advances in neural information processing systems. pp.
671 386–396 (2017)
- 672 16. Li, Y., Liu, M.Y., Li, X., Yang, M.H., Kautz, J.: A closed-form solution to photore-
673 alistic image stylization. In: Proceedings of the European Conference on Computer
674 Vision (ECCV). pp. 453–468 (2018)
- 675 17. Lu, M., Zhao, H., Yao, A., Chen, Y., Xu, F., Zhang, L.: A closed-form solution to
676 universal style transfer. In: Proceedings of the IEEE International Conference on
677 Computer Vision. pp. 5952–5961 (2019)

- 675 18. Luan, F., Paris, S., Shechtman, E., Bala, K.: Deep photo style transfer. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4990–4998 (2017) 676
677 19. Mroueh, Y.: Wasserstein style transfer. arXiv preprint arXiv:1905.12828 (2019) 678
678 20. Risser, E., Wilmot, P., Barnes, C.: Stable and controllable neural texture synthesis 679 and style transfer using histogram losses. arXiv preprint arXiv:1701.08893 (2017) 680
680 21. Sheng, L., Lin, Z., Shao, J., Wang, X.: Avatar-net: Multi-scale zero-shot style 681 transfer by feature decoration. In: Proceedings of the IEEE Conference on Computer 682 Vision and Pattern Recognition. pp. 8242–8250 (2018) 683
683 22. Ulyanov, D., Lebedev, V., Vedaldi, A., Lempitsky, V.S.: Texture networks: Feed- 684 forward synthesis of textures and stylized images. In: ICML. vol. 1, p. 4 (2016) 684
685 23. Ulyanov, D., Vedaldi, A., Lempitsky, V.: Improved texture networks: Maximizing 686 quality and diversity in feed-forward stylization and texture synthesis. In: Proceedings 687 of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 6924–6932 (2017) 688
689 24. Yoo, J., Uh, Y., Chun, S., Kang, B., Ha, J.W.: Photorealistic style transfer via 690 wavelet transforms. In: Proceedings of the IEEE International Conference on Computer 691 Vision. pp. 9036–9045 (2019) 692
692 25. Zhang, H., Dana, K.: Multi-style generative network for real-time transfer. arXiv 693 preprint arXiv:1703.06953 (2017)
- 694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719