Avatar-Net: Multi-scale Zero-shot Style Transfer by Feature Decoration

Lu Sheng[1], Ziyi Lin[2], Jing Shao[2], Xiaogang Wang[1]
[1]CUHK-SenseTime Joint Lab, The Chinese University of Hong Kong [2]SenseTime Research
{lsheng, xgwang}@ee.cuhk.edu.hk, {linziyi, shaojing}@sensetime.com

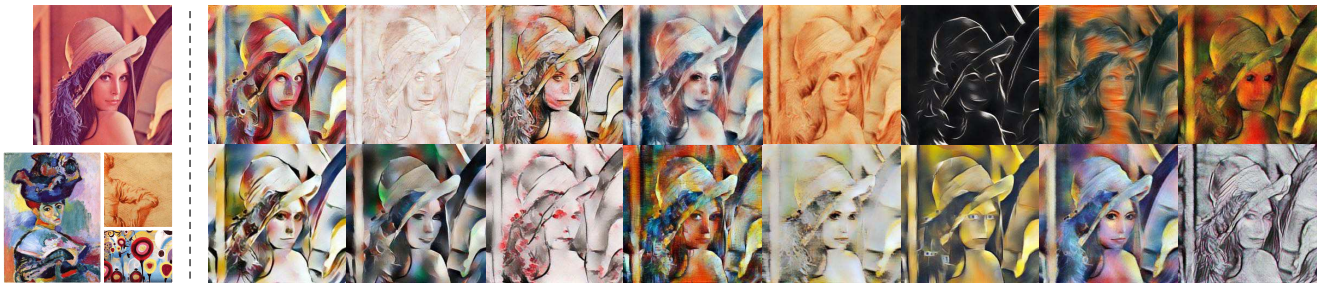


Figure 1. Exemplar stylized results by the proposed Avatar-Net, which faithfully transfers the *Lena* image by arbitrary style.

Abstract

Zero-shot artistic style transfer is an important image synthesis problem aiming at transferring arbitrary style into content images. However, the trade-off between the generalization and efficiency in existing methods impedes a high quality zero-shot style transfer in real-time. In this paper, we resolve this dilemma and propose an efficient yet effective Avatar-Net that enables visually plausible multi-scale transfer for arbitrary style. The key ingredient of our method is a style decorator that makes up the content features by semantically aligned style features from an arbitrary style image, which does not only holistically match their feature distributions but also preserve detailed style patterns in the decorated features. By embedding this module into an image reconstruction network that fuses multi-scale style abstractions, the Avatar-Net renders multi-scale stylization for any style image in one feed-forward pass. We demonstrate the state-of-the-art effectiveness and efficiency of the proposed method in generating high-quality stylized images, with a series of successive applications include multiple style integration, video stylization and etc.

1. Introduction

Unlike taking days or months to create a particular artistic style by a diligent artist, modern computational methods have enabled fast and reliable style creation for natural images. Especially inspired by the remarkable representative power of convolutional neural networks (CNNs), the seminal work by Gatys *et al*. [8, 9] discovered that multi-level feature statistics extracted from a trained CNN notably represent the characteristics of visual styles, which boosts the development of style transfer approaches, either by iterative optimizations [9, 16, 30, 20] or feed-forward networks [5, 13, 18, 20, 31, 28, 29, 25, 3]. However, a dilemma in terms of generalization and quality versus efficiency hampers the availability of style transfer for arbitrary style. Since the optimization-based approaches render visually plausible results for various styles at the sacrifice of executing efficiency[9, 16, 30], while the feed-forward networks are either restricted by a finite set of predefined styles [5, 13, 18, 20, 31, 28], or simplify the zero-shot transfer with compromised visual quality [12, 25, 29].

Valuable efforts have been devoted to solving this dilemma. A common practice is employing external style signals to supervise the content modification [12, 25, 31, 29, 3] on a feed-forward network. But these methods require the networks trained by the perceptual loss [13], which has been known instable and produces compromised stylized patterns [10, 30]. In contrast, another category [19, 4] manipulates the content features under the guidance of the style features in a shared high-level feature space. By decoding the manipulated features back into the image space with a style-agnostic image decoder, the reconstructed images will be stylized with seamless integration of the style patterns. However, current techniques may either over-distort the content and add unconstrained patterns [19], or fail to retrieve complete style patterns when large domain gap ex-

ists between the content and style features [4].

Facing the aforementioned challenges, we propose a zero-shot style transfer model *Avatar-Net* that follows a similar style-agnostic paradigm [19, 4] in the aforementioned latter category. Specifically, we introduce a novel patch-based *style decorator* module that decorates the content features with the characteristics of the style patterns, while keeps the content semantically perceptible. The style decorator does not only match the holistic style distribution, but also explicitly retrieves detailed style patterns without distortion. In the meanwhile, other than auto-encoders [4, 19, 3] for semantic feature extraction and style-agnostic image decoder, we introduce an innovative hourglass network with multiple skip connections for multi-scale holistic style adaptation. It is thus straightforward to perform multi-scale zero-shot style transfer by 1) extracting the content and style features via its encoding module, 2) decorating the content features by the patch-based style decorator, and 3) progressively decoding the stylized features with multi-scale holistic style adaptation. Note that the proposed hourglass network is trained solely to reconstruct natural images. As shown in Fig. 1, the proposed Avatar-Net can synthesize visually plausible stylized *Lena* images for arbitrary style. Comprehensive evaluations have been conducted to compare with the prior style transfer methods [12, 4, 19, 13], our method achieves the state-of-the-art performance in terms of both visual quality and efficiency.

Our contributions in this paper are threefold: (1) A novel patch-based feature manipulation module named as style decorator, transfers the content features to semantically nearest style features and simultaneously minimizes the discrepancy between their holistic feature distributions. (2) A new hourglass network equipped with multi-scale style adaptation enables visually plausible multi-scale transfer for arbitrary style in one feed-forward pass. (3) Theoretical analysis proves that the style decorator module owns superior transferring ability, and the experimental results demonstrate the effectiveness of the proposed method with superior visual quality and economical processing cost.

2. Related Work

Style transfer is a kind of non-realistic rendering techniques [15] that is closely related to texture synthesis [6, 7]. It usually exploits local statistics for efficient cross-view dense correspondences and texture quilting [11, 6, 26, 21]. Although these methods produce appealing stylized images [26, 21], dense correspondences are limited to a pair of images with similar contents, thus inapplicable to zero-shot style transfer.

Optimization-based Stylization. Gatys *et al*. [9, 8] at the first time formulated the style as multi-level feature correlations (*i.e*., Gram matrix) from a trained neural network for image classification, and defined the style transfer as an iterative optimization problem that balances the content similarity and style affinity in the feature level (or termed as perceptual loss [13]). A number of variants have been developed thereafter to adapt this framework to different scenarios and requirements, including photorealistic rendering [23], semantically composite transfer [2], temporal coherence [24] and so on. Changing the global style statistics into Markovian feature assembling, a similar framework is also proposed for semantic image synthesis [16]. Despite visual appealing performances for arbitrary style, the results are not stable [24, 30] and the perceptual loss usually requires careful parameter tunning for different styles [10]. Moreover, the inefficiency of the optimization-based approaches restrains real-time applications.

Feed-Forward Approximation. Recent researches [13, 28, 17, 31, 3, 25, 29, 18, 5] try to tackle the complexity issue by approximating the iterative back-propagating procedure to feed-forward neural networks, either trained by the perceptual loss or Markovian generative adversarial loss [17]. However, [13, 28, 17] have to train a independent network for every style. To strengthen the representation power for multiple styles, StyleBank [3] learns filtering kernels for styles and Li *et al*. [18] transfered styles by binary selection units, as well as Dumoulin *et al*. proposed conditional instance normalization [5] controlled by channel-wise statistics learned for each style. But the manually designed style abstractions are often short for the representation of unseen styles and the combinational optimization over multiple styles often compromises the rendering quality [20, 10].

Zero-shot Style Transfer. To achieve zero-shot style transfer in a feed-forward network, Huang *et al*. [12] adjusted channel-wise statistics of the content features by adaptive instance normalization (AdaIN), and trained a feature decoder by a combinational scale-adapted content and style losses. Chen *et al*. [4] swapped the content feature patches with the closest style features (*i.e*. Style-Swap) at the intermediate layer of a trained auto-encoder, while Li *et al*. [19] transferred multi-level style patterns by recursively applying whitening and coloring transformation (WCT) to a set of trained auto-encoders with different levels. However, AdaIN over-simplifies the transferring procedure and Style-Swap cannot parse sufficient style features when content and style images are semantically different. WCT is the state-of-the-art method, but the holistic transformation may results in distorted style patterns in the transferred features, and generate unwanted patterns in the output image.

The proposed zero-shot style transfer is related to [19, 4, 12]. But the style decorator improves AdaIN and WCT as it reserves the detailed style patterns rather than the parameterized feature statistics. It also outperforms Style-Swap as it effectively parses the complete set of style features regardless of their domain gap. Moreover, the proposed network performs multi-scale style adaptation in one

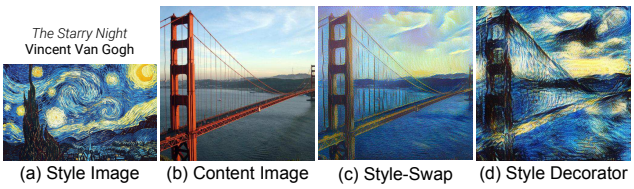


Figure 2. Style patterns are overlaid onto the content patches according to their semantic similarity rather than their texture agreement. (a) and (b) are the style and content images. (c)-(d) are the results by Style-Swap [4] and the proposed style decorator. Style-Swap pays too much attention on the textural similarity, thus receives fewer high-level style patterns.

feed-forward pass, which surpasses AdaIN and WCT since AdaIN requires a style-oriented image decoder and WCT needs a set of recursive feed-forward passes to enable multi-scale style transfer.

3. Style Transfer via Feature Manipulation

Let a trained feed-forward network extract the features in the bottleneck layer for an image $\mathbf{x}$ as $\mathbf{z} = E_{\boldsymbol{\theta}_{\text{enc}}}(\mathbf{x}) \in \mathbb{R}^{H\times W\times C}$, with height H, width W and channel C. The extracted features are decoded via $\tilde{\mathbf{x}} = D_{\boldsymbol{\theta}_{\text{dec}}}(\mathbf{z})$ to the reconstructed images. The encoder and decoder modules are parameterized by $\boldsymbol{\theta}_{\text{enc}}$ and $\boldsymbol{\theta}_{\text{dec}}$, respectively.

Denote $\mathbf{z}_c$ and $\mathbf{z}_s$ as the features for the content image $\mathbf{x}_c$ and style image $\mathbf{x}_s$. By transferring $\mathbf{z}_c$ into the domain of $\mathbf{z}_s$, it is desired that the transferred features $\mathbf{z}_{cs} = \mathcal{F}(\mathbf{z}_c; \mathbf{z}_s)$ infer the spatial distribution of the content features and the textural characteristics of the style features. The key challenge is to design a feature transfer module that both holistically adapts the domain of $\mathbf{z}_{cs}$ to that of $\mathbf{z}_s$ and semantically links elements in $\mathbf{z}_{cs}$ to paired elements in $\mathbf{z}_s$ and $\mathbf{z}_c$.

3.1. Revisiting Normalized Cross-correlation

A type of modules generates $\mathbf{z}_{cs}$ by non-parametrically swapping the patches in $\mathbf{z}_c$ by their nearest neighbors in $\mathbf{z}_s$. It encourages concrete style patterns in $\mathbf{z}_{cs}$ and explicitly binds $\mathbf{z}_c$ and $\mathbf{z}_{cs}$ by their spatial distributions.

Normalized Cross-correlation (NCC). It is one effective metric for patch-wise nearest neighbor searching [16, 4], by scoring the cosine distance between a content patch $\phi_i(\mathbf{z}_c)$ and a style patch $\phi_j(\mathbf{z}_s)$ and returning the nearest patch

$$\phi_i(\mathbf{z}_{cs}) = \underset{j\in\{1,\ldots,N_s\}}{\arg\max} \frac{\langle \phi_i(\mathbf{z}_c), \phi_j(\mathbf{z}_s)\rangle}{\|\phi_i(\mathbf{z}_c)\|\|\phi_j(\mathbf{z}_s)\|}, i \in \{1,\ldots,N_c\}.$$

However, each $\phi_i(\mathbf{z}_c)$ is required to compare with any patch in $\mathbf{z}_s$. Thanks to the GPU accelerated convolution operations, NCC can be converted into several steps of efficient convolutions, such as the way applied in Style-Swap [4, 16].

Discussions. Although concrete style patterns are preserved in $\mathbf{z}_{cs}$, it is highly biased towards the content features in $\mathbf{z}_c$, since it suggests the matched style patches strictly follow the local variations as the content patches. Therefore, the spiral patterns in sky in Vincent Van Gogh's *starry night* cannot be propagated to a sky patch in the content image, as shown in Fig. 2. And only a limited portion of style patterns are parsed to assemble $\mathbf{z}_{cs}$ when the domain of $\mathbf{z}_c$ and that of $\mathbf{z}_s$ are far apart, as depicted by the swapping procedure shown in Fig. 3(c). Thus the stylized images preserve more content patterns with only a small portion of semantically aligned style patterns. In fact, two patches can be matched as long as they are paired in some forms of semantic agreement rather than this over-restrictive patch similarity. It means that a desired patch matching can be *invariant* to their domain gap and the patches are paired by comparing their local statistics in a *common* space that the textural characteristics have been dispelled.

3.2. Style Decorator

To resolve drawbacks raised from the existing feature transfer modules, we propose a novel style decorator module that robustly propagates the thorough style patterns onto the content features so that the distributions of $\mathbf{z}_{cs}$ and $\mathbf{z}_s$ are maximally aligned and the detailed style patterns are semantically perceptible in $\mathbf{z}_{cs}$. It is achieved by a relaxed cross correlation between patches in a normalized coordinate that sufficiently whitens the textures from the original feature spaces for both features. The swapped normalized features $\bar{\mathbf{z}}_{cs}$ explicitly correspond to the original style features, thus from which we are able to reconstruct the stylized features $\mathbf{z}_{cs}$ that both statistically follow the style of $\mathbf{z}_s$ and the spatial layout of the content feature $\mathbf{z}_c$. The style decorator is achieved by a three-step procedure:

Step-I: Projection. We project $\mathbf{z}_c$ and $\mathbf{z}_s$ onto the same space, written as $\bar{\mathbf{z}}_c$ and $\bar{\mathbf{z}}_s$, with sufficient compactness and similar scale in magnitude. In detail, the projection operation for the content (style) feature is

$$\bar{\mathbf{z}}_c = \mathbf{W}_c \otimes (\mathbf{z}_c - \boldsymbol{\mu}(\mathbf{z}_c)), \bar{\mathbf{z}}_s = \mathbf{W}_s \otimes (\mathbf{z}_s - \boldsymbol{\mu}(\mathbf{z}_s)), \quad (1)$$

where $\otimes$ denotes convolutional operator, and the whitening kernels $\mathbf{W}_c$ and $\mathbf{W}_s$ are derived by some forms of whitening matrices onto the covariance matrices $\mathcal{C}(\mathbf{z}_c)$ and $\mathcal{C}(\mathbf{z}_s)$. And $\boldsymbol{\mu}(\mathbf{z}_c)$ and $\boldsymbol{\mu}(\mathbf{z}_s)$ are the mean features. $\bar{\mathbf{z}}_c$ and $\bar{\mathbf{z}}_s$ keep the characteristics from their original data, but both of them statistically follow standard normal distribution.

Step-II: Matching and Reassembling. In this normalized domain, we want to align any element in $\bar{\mathbf{z}}_c$ with the nearest element in $\bar{\mathbf{z}}_s$, so as to reconstruct $\bar{\mathbf{z}}_c$ by reassembling the corresponded elements in $\bar{\mathbf{z}}_s$. In this case, the reassembled normalized style features $\bar{\mathbf{z}}_{cs}$ contains concrete normalized style patterns but their spatial layouts come from the normalized content features. Therefore, the matching and reassembling operations seamlessly incorporate the desired content and style characteristics in the resultant $\bar{\mathbf{z}}_{cs}$.

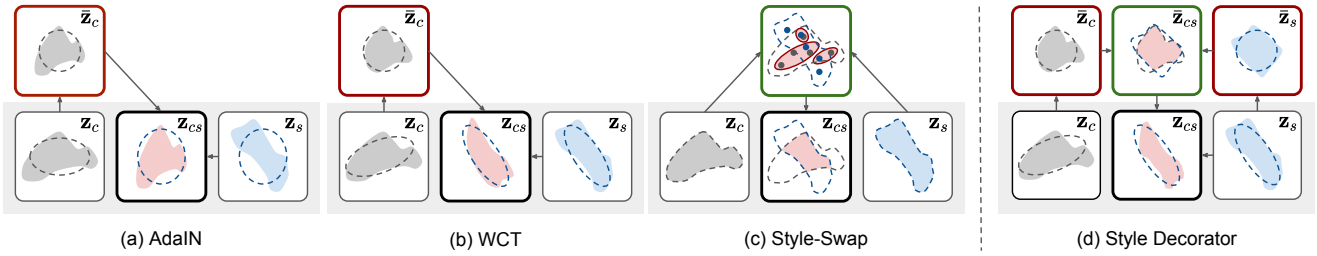


Figure 3. Comparison of distribution transformation by different feature transfer modules. (a) AdaIN cannot completely dispel the textures from $\mathbf{z}_c$ and propagate enough style textures to $\mathbf{z}_{cs}$. (b) WCT has optimal matching between $\mathcal{G}(\mathbf{z}_{cs})$ and $\mathcal{G}(\mathbf{z}_s)$ but it introduces rare patterns observed in the transformed features. (c) Style-Swap matches patches from $\mathbf{z}_c$ and $\mathbf{z}_s$, and thus only captures the overlapped features. (d) Style decorator matches the normalized features $\bar{\mathbf{z}}_c$ and $\bar{\mathbf{z}}_s$, the generated $\mathbf{z}_{cs}$ parses all possible style patterns to the content features. The red boxes show the normalization and the green boxes visualize the normalized cross-correlation.

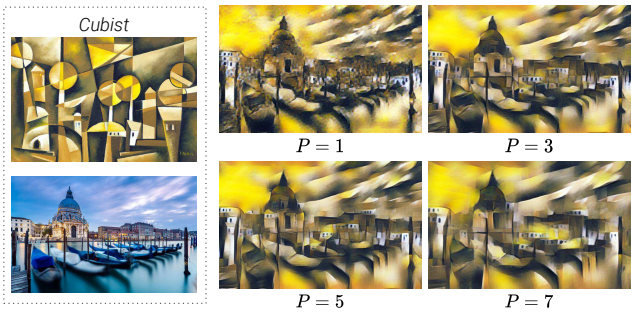


Figure 4. Stylization controlled by different patch size. With an increasing patch size, the stylized image tends to be more blocky like the geometric patterns in the style image *cubist*.

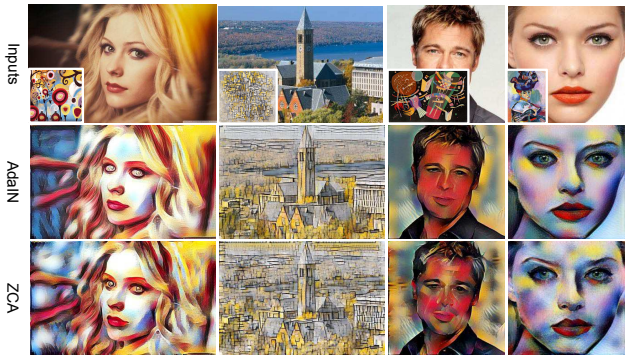


Figure 5. Both ZCA and AdaIN are effective for the projection and reconstruction steps. AdaIN will preserve a bit more amount of content patterns but it is much more efficient than ZCA.

We apply the patch matching and reassembling via the help of normalized cross-correlation, which can be efficiently transformed into several convolutional operations as

$$\bar{\mathbf{z}}_{cs} = \mathbf{\Phi}(\bar{\mathbf{z}}_s)^\top \otimes \mathcal{B}(\bar{\mathbf{\Phi}}(\bar{\mathbf{z}}_s) \otimes \bar{\mathbf{z}}_c), \tag{2}$$

where $\mathbf{\Phi}(\bar{\mathbf{z}}_s) \in \mathbb{R}^{P \times P \times C \times (H \times W)}$ is the style kernel consists of patches in $\bar{\mathbf{z}}_s$, and P is the patch size. In addition, $\bar{\mathbf{\Phi}}(\bar{\mathbf{z}}_s)$ is the normalized style kernel that is divided by the channel-wise ℓ_2 norm of each patch. The patch matching procedure is at first convolving the normalized content feature $\bar{\mathbf{z}}_c$ with $\bar{\mathbf{\Phi}}(\bar{\mathbf{z}}_s)$ and then binarizing the resultant scores (*i.e.*, $\mathcal{B}(\cdot)$) such that the maximum value along the channel is 1 and the rest are 0. Then the reassembling procedure applies a deconvolutional operation by $\mathbf{\Phi}(\bar{\mathbf{z}}_s)^\top$ to regenerate the style patterns from the binary scores.

This patch matching between $\bar{\mathbf{z}}_c$ and $\bar{\mathbf{z}}_s$ parses effective and complete correspondences since $\bar{\mathbf{z}}_c$ and $\bar{\mathbf{z}}_s$ have the maximal overlap with each other, thus each element in $\bar{\mathbf{z}}_c$ can find a suitable correspondence in $\bar{\mathbf{z}}_s$ whilst possibly every element in $\bar{\mathbf{z}}_s$ can be well retrieved, as demonstrated in Fig. 3(d). The matching and reassembling step actually adds two style-controlled convolutional layers to the feed-forward network and thus its implementation is usually efficient in modern deep learning platforms.

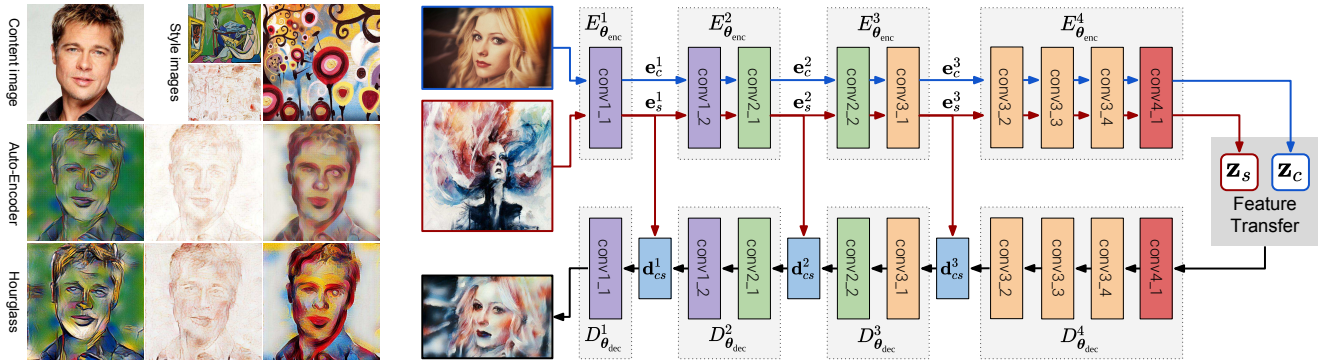
Step-III: Reconstruction. After reorganizing the normalized style patches into $\bar{\mathbf{z}}_{cs}$, we will reconstruct it into the domain of the style feature $\mathbf{z}_s$. In detail, we apply the coloring transformation with respect to $\mathbf{z}_s$ into the reassembled feature, the reconstruction of the stylized feature is

$$\mathbf{z}_{cs} = \mathbf{C}_s \otimes \bar{\mathbf{z}}_{cs} + \boldsymbol{\mu}(\mathbf{z}_s), \tag{3}$$

where the coloring kernel $\mathbf{C}_s$ is also derived from the covariance matrix $\mathcal{C}(\mathbf{z}_s)$.

Discussions. Similar as WCT and AdaIN, the *projection* and *reconstruction* steps are designed to encourage that the second-order statistics of the stylized feature $\mathbf{z}_{cs}$ to be matched to those of the style feature $\mathbf{z}_s$[1], *i.e.*, their Gram matrices $\mathcal{G}(\mathbf{z}_s) \simeq \mathcal{G}(\mathbf{z}_{cs})$. Actually, we may employ Adaptive Instance Normalization (AdaIN) [12], or Zero-phase Component Analysis (ZCA) [14] used in WCT [19] as the whitening and coloring kernels. Whitened features $\bar{\mathbf{z}}$ by ZCA minimize the ℓ_2 distance with respect to $\mathbf{z}$, thus they

[1]They will not be exactly the same, since the population of $\bar{\mathbf{z}}_{cs}$ has been changed to follow a similar spatial distribution of $\bar{\mathbf{z}}_c$.



(a) Auto-Encoder v.s. Hourglass (b) Network Architecture

Figure 6. (a) Stylization comparison by auto-encoder and style-augmented hourglass networks. Style decorator is applied as the feature transfer module. The auto-encoder and the hourglass networks share the same main branch. (b) The network architecture of the proposed style-augmented hourglass network. Detailed implementation is depicted in Sec. 5.1.

marginally preserve the original feature orientations and element-wise distributions. AdaIN is much faster than ZCA but it cannot optimally dispel feature correlations as what ZCA does, thus the transferred features contain a slightly more content patterns, as shown in Fig. 3(a) and (b). But for the sake of efficiency, the AdaIN induced style decorator is good enough for a real-time zero-shot style transfer, as shown in Fig. 5.

On the other hand, the *matching and reassembling* step pairs the elements in $\mathbf{z}_c$ and $\mathbf{z}_s$ by matching their normalized counterparts, thus it effectively reduces the bias and enriches the parsing diversity, which is much more effective than the way by Style-Swap [4]. In addition, the patch size P affects the scale of the presented styles since a larger patch size leads to a more global style patterns, for example the blocky geometric patterns of style *cubist* in Fig. 4.

4. Multi-scale Zero-shot Style Transfer

The proposed Avatar-Net employs a hourglass network with multi-scale style adaptation modules that progressively fuse the styles from the encoded features into the corresponded decoded features, thus it enables multi-scale style transfer in one feed-forward pass, as shown in Fig. 6(a).

As shown in Fig. 6(b), the main branch of the Avatar-Net is analogous to conventional encoder-decoder architectures that stack an encoder $E_{\boldsymbol{\theta}_{\text{enc}}}(\cdot)$ and an decoder $D_{\boldsymbol{\theta}_{\text{dec}}}(\cdot)$. In detail, the encoder is a concatenation of several encoding blocks $E^l_{\boldsymbol{\theta}_{\text{enc}}}(\cdot)$ that progressively extracts the intermediate features $\mathbf{e}^l = E^l_{\boldsymbol{\theta}_{\text{enc}}}(\mathbf{e}^{l-1}), l \in \{1, \dots, L\}$ and produces the bottleneck feature $\mathbf{z}$ after the final block $\mathbf{z} = E^{L+1}_{\boldsymbol{\theta}_{\text{enc}}}(\mathbf{e}^L)$. Inversely, the decoder progressively generates intermediate features $\mathbf{d}^l = D^{l+1}_{\boldsymbol{\theta}_{\text{dec}}}(\mathbf{d}^{l+1})$ starting from $\mathbf{z}$, in which $\mathbf{d}^l$ is further updated by fusing with the corresponded encoded features $\mathbf{e}^l$ via the style adaptive feature fusion module. The output image is decoded by $\tilde{\mathbf{x}} = D^1_{\boldsymbol{\theta}_{\text{dec}}}(\mathbf{d}^1)$.

Given the set of encoded style features and the decoded stylized feature pairs $(\mathbf{e}^l_s, \mathbf{d}^l_{cs}), l \in \{1, \dots, L\}$ extracted from the main branch of the Hourglass network, the Style Fusion module is similarly as AdaIN [12]:

$$\mathcal{F}_{\text{SF}}(\mathbf{d}^l_{cs}; \mathbf{e}^l_s) = \boldsymbol{\sigma}(\mathbf{e}^l_s) \circ \left(\frac{\mathbf{d}^l_{cs} - \boldsymbol{\mu}(\mathbf{d}^l_{cs})}{\boldsymbol{\sigma}(\mathbf{d}^l_{cs})} \right) + \boldsymbol{\mu}(\mathbf{e}^l_s), \quad (4)$$

where $\circ$ denotes channel-wise multiplication and $\boldsymbol{\sigma}(\cdot)$ is the channel-wise standard deviation. One may argue that this module is suboptimal to ZCA as it does not optimally match their second-order statistics, as visualized in Fig. 3(a). But as the proposed network suggest similar pattern distribution of $\mathbf{d}^l_{cs}$ as that of the encoded features $\mathbf{e}^l_s$, the training of the decoder will let the proposed module move towards ZCA. Moreover, its economical computational complexity also speeds up the stylization.

The style transfer requires a special procedure to fit the proposed network architecture. At first, Avatar-Net takes a content image $\mathbf{x}_c$ and an arbitrary style image $\mathbf{x}_s$ as inputs, and extracts $\mathbf{z}_c$ and $\mathbf{z}_s$ in the bottleneck layer through the encoder module. In the meanwhile, the style image $\mathbf{x}_s$ also bypasses the multi-scale encoded features $\{\mathbf{e}^l_s\}^L_{l=1}$. Secondly, the content feature $\mathbf{z}_c$ is then transferred based on the style features $\mathbf{z}_s$ through the proposed style decorator module. In the end, the stylized image $\tilde{\mathbf{x}}_{cs}$ is inverted by the decoded module $D_{\boldsymbol{\theta}_{\text{dec}}}(\mathbf{z}_{cs})$ with multiple style fusion modules that progressively modify the the decoded features $\mathbf{d}^l_{cs}$ under the guidance of multi-scale style patterns $\mathbf{e}^l_s$, from $l = L$ to 1, as shown in Fig. 6(b).

5. Experimental Results and Discussions

5.1. Network Architecture And Training

Avatar-Net copies the architecture of a pretrained VGG-19 (up to `conv4_1`) [27] to the encoder $E_{\boldsymbol{\theta}_{\text{enc}}}(\cdot)$. The de-

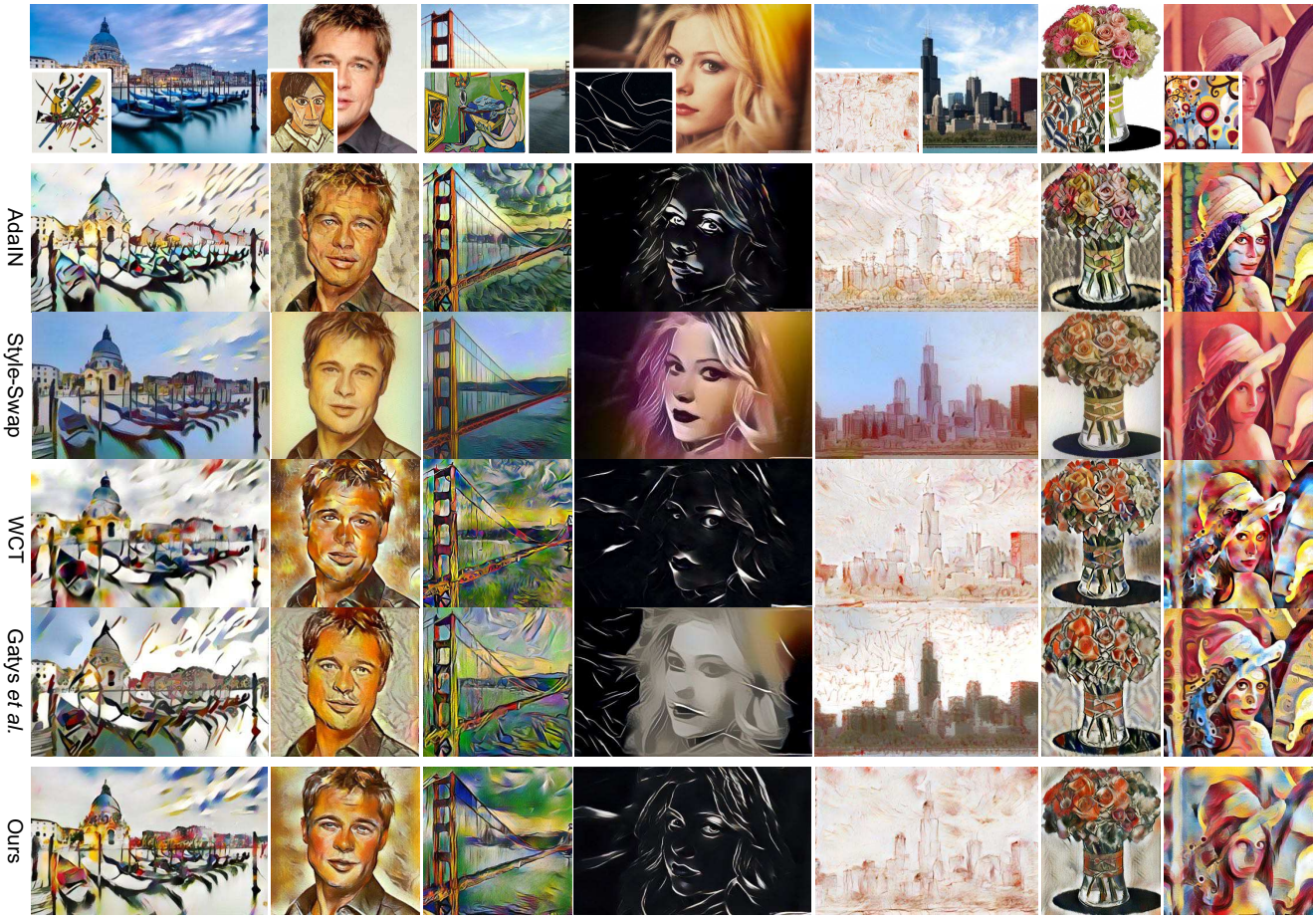


Figure 7. Comparison of some exemplar stylization results. The top row shows the content and style pairs and the rest rows present the stylized results by AdaIN [12], Style-Swap [4], WCT [19], Gatys *et al*. [9], and the proposed method.

coder $D_{\boldsymbol{\theta}_{\text{dec}}}(\cdot)$ is randomly initialized and mirrors the encoder with all pooling layers replaced by nearest upsampling and padding layers by reflectance padding. The shortcut connections link `conv1_1`, `conv2_1` and `conv3_1` to their corresponded decoded layers. Normalization is not applied in each `conv` layer so as to increase the reconstruction performance and stability, as suggested by [12].

The proposed model is to reconstruct perceptually similar images as the input images [13]:

$$\ell_{\text{total}} = \|\mathbf{x} - \tilde{\mathbf{x}}\|_2^2 + \lambda_1 \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \|\boldsymbol{\Psi}^i_{\text{VGG}}(\mathbf{x}) - \boldsymbol{\Psi}^i_{\text{VGG}}(\tilde{\mathbf{x}})\|_2^2 + \lambda_2 \ell_{\text{TV}}(\mathbf{x}), \quad (5)$$

where $\boldsymbol{\Psi}^i_{\text{VGG}}(\mathbf{x})$ extract the feature of $\mathbf{x}$ at layer i in a fixed VGG-19 network. The set $\mathcal{I}$ contains `conv1_1`, `conv2_1`, `conv3_1` and `conv4_1` layers. Total variation loss $\ell_{\text{TV}}(\mathbf{x})$ is also added to enforce piece-wise smoothness. The weighting parameters are simply set as $\lambda_1 = 0.1$ and $\lambda_2 = 1$ for balancing the gradients from each term.

We train the network on the MSCOCO [22] dataset with roughly $80,000$ training samples. Adam optimizer is applied with a fixed learning rate of 0.001 and a batch size of 16. During the training phase, the training images are randomly resized and cropped to 256×256 patches. Note that our method is suitable for images with arbitrary size.

5.2. Comparison with Prior Arts

To validate the effectiveness of the proposed method, we compare it with two types of zero-shot stylization methods based on (1) iterative optimization [9] and (2) feed-forward network approximation [12, 19, 4][2].

Qualitative Evaluations. The optimization-based approach [9] is able to transfer arbitrary style but the generated results are not stable because the weights trading off the content and style losses are sensitive and the optimization is vulnerable to be stuck onto a bad local minimum, as

[2]The results by the reference methods are obtained by their public codes with default configurations.

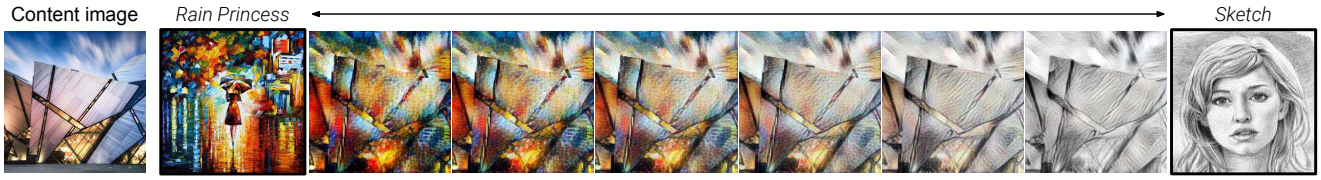


Figure 8. Style Interpolation between *Rain Princess* and *Sketch*.

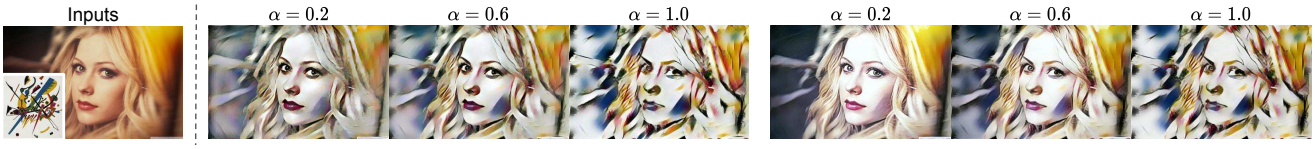


Figure 9. Trading off between the content and style images. The left half shows the trading off between the normalized features $\bar{\mathbf{z}}_{cs}$ and $\bar{\mathbf{z}}_c$. The right half illustrates the balancing between the content and stylized features $\mathbf{z}_{cs}$ and $\mathbf{z}_c$.

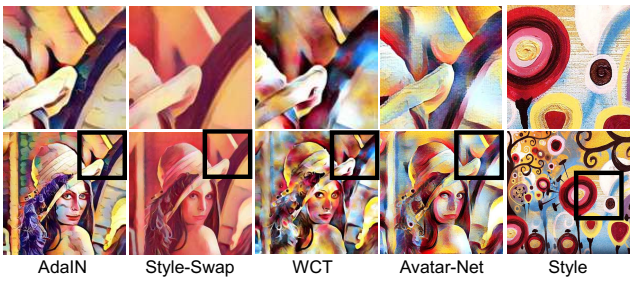


Figure 10. Result close-ups. Regions marked by bounding boxes are zoomed in for a better visualization.

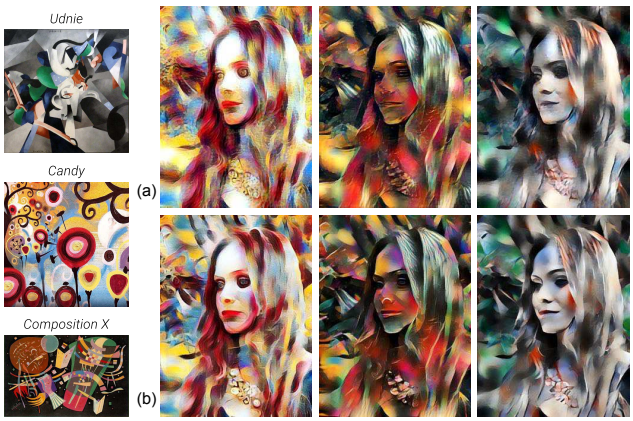


Figure 11. Memory reduction. (a) The economical style decorator by sampling the style kernels with a stride of 4, in which the patch width is 5. (b) The complete style decorator. The number of style patches in (a) are $\frac{1}{16}$ of that in (b) but the performances are similar except a slight lost of detailed style patterns.

shown in 4th and 5th columns in Fig. 7. Both AdaIN [12] and WCT [19] holistically adjust the content features to match the second-order statistics of the style features, but AdaIN mimics the channel-wise means and variances and thus provides a suboptimal solution and the learned decoder usually add similar repeated texture patterns for all stylized images. Although WCT optimally matches the second-order statistics, it cannot always parse the original style patterns and may generate unseen and distorted patterns, for example the background clutters in the 2nd column and unwanted spiral patterns in the 3rd column, as well as missed circular patterns in the last column for the style *candy*. Style-Swap is too rigorous so that the stylized features strictly preserve the content features and only receive low-level style patterns. This artifact almost occurs in every examples in Fig. 7.

Method	Execution Time	
	256×256 (sec)	512×512 (sec)
Gatys *et al.* [9]	12.18	43.25
AdaIN [12]	0.053	0.11
WCT [19]	0.62	0.93
Style-Swap [4]	0.064	0.23
Ours-ZCA	0.26	0.47
Ours-ZCA-Sampling	0.24	0.32
Ours-AdaIN	0.071	0.28

Table 1. Execution time comparison.

The proposed method seamlessly reassembles the style patterns according to the semantic spatial distribution of the content image, which works for different kinds of styles, from global hue to local strokes and sketches, as shown in Fig. 7. Even though style patterns replace the content information to regenerate the stylized images, necessary content components like human faces, building and skylines are still visually perceived as the combination of the style patterns.

We also provide result close-ups in Fig. 10, the visual result by Avatar-Net receives concrete multi-scale style patterns (e.g., color distribution, brush strokes and circular patterns in candy image). WCT distorts the brush strokes and circular patterns. AdaIN cannot even keep the color distribution, while style-swap fails in this example.

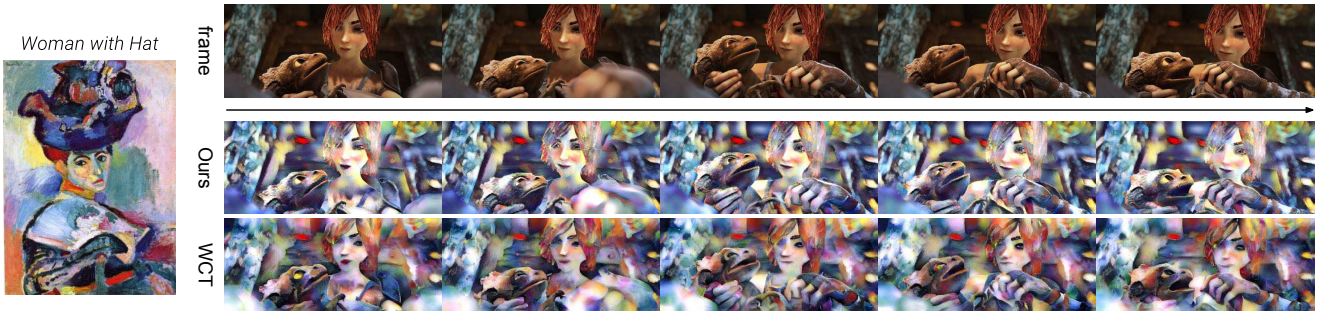


Figure 12. Video stylization. A sequence in MPI Sintel dataset [1] is stylized by Henri Matisse's *Woman With Hat*. **Style Decorator**: The stylized patterns in one object keep coherent among adjacent frames. **WCT**: The results have shape distortions around the face region and contain flickering artifacts among adjacent frames. Please refer to the supplementary materials for a video demonstration.

Efficiency. In Tab. 1, we compare the running time with the reference methods. We implemented these methods on the Tensorflow platform for a fair comparison. The method by Gatys *et al*. [9] is slow since it requires hundreds of forward and backward passes to converge. WCT [19], AdaIN [12] and Style-Swap [4] are all based on feed-forward networks, in which WCT is relatively slower as it requires several feed-forward passes and the operation of SVD has to be executed in CPU. Our implemented Style-Swap is efficient in GPU and thus much faster than the reported speed [4].

The proposed method is more efficient than WCT, because the hourglass architecture enables multi-scale processing in one feed-forward pass. But the style decorator equipped by ZCA also needs a CPU-based SVD, its speed is thus not comparable to AdaIN and Style-Swap. We can improve the efficiency by replacing it by AdaIN, in which case the execution time is tremendously reduced to the same level of Style-Swap and AdaIN. Note that the proposed method can be economical in memory, by randomly (or grid) sampling the style patches in the style decorator procedure. Due to the local coherence in the style features, we find the resultant stylized images have similar performances as the complete version (in Fig. 11).

5.3. Applications

The flexibility of the Avatar-Net is further manifested by several applications as follows:

Trade-off Between Content and Style. The degree of stylization can be adjusted by two variants. (1) Adjust the normalized stylized features $\bar{\mathbf{z}}_{cs} \leftarrow \alpha\bar{\mathbf{z}}_c + (1-\alpha)\bar{\mathbf{z}}_{cs}, \forall\alpha \in [0,1]$, and then recover the stylized features $\mathbf{z}_{cs}$. (2) Directly adjust the stylized features $\mathbf{z}_{cs} \leftarrow \alpha\mathbf{z}_c + (1-\alpha)\mathbf{z}_{cs}$. In the former variant, $\alpha = 0$ lets the model degrade to the WCT, while $\alpha = 1$ means the image is only reconstructed by the style patterns. It is harder to control the latter variant due to the magnitude dissimilarities between the content and style features. We depict these variants in this experiment (in Fig. 9), but we do not adjust the low-level style adaptation in the shortcut links, which can be interpolated by [12] to enable multi-scale transfer.

Style Interpolation. Convexly integrating multiple style images $\{\mathbf{x}_s^k\}_{k=1}^K$ with weights $\{w_k\}_{k=1}^K$ that $\sum_{k=1}^K w_k = 1$, the stylized features are $\mathbf{z}_{cs} = \sum_{k=1}^K w_k \mathbf{z}_{cs}^k$. The style adaptations in the shortcut links are also extended to a convex combination of the stylized features. But due to the magnitude dissimilarities among different style features, the interpolated stylization is also affected by their feature magnitudes, as shown in Fig. 8.

Video Stylization. In addition to image-based stylization, our model can perform video stylization merely based on per-frame style transfer, as visualized in Fig. 12. The style decorator module is stable and coherent over adjacent frames since the alignment between feature patches is usually spatially invariant and robust to small content variations. On the contrary, WCT, as an example, contains severe content distortions and temporal flickering.

6. Concluding Remarks

In this work, we propose a fast and reliable multi-scale zero-shot style transfer method integrating an style decorator for semantic style feature propagation and a hourglass network for multi-scale holistic style adaptation. Experimental results demonstrate its superiority in generating arbitrary stylized images. As a future direction, one may replace the projection and reconstruction steps in the style decorator by learnable modules for increased alignment robustness and executing efficiency.

Acknowledgments. This work is supported in part by SenseTime Group Limited, in part by the General Research Fund through the Research Grants Council of Hong Kong under Grants CUHK14213616, CUHK14206114, CUHK14205615, CUHK419412, CUHK14203015, CUHK14207814, CUHK14202217, and in part by the Hong Kong Innovation and Technology Support Programme Grant ITS/121/15FX.

References

[1] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In A. Fitzgibbon et al. (Eds.), editor, *ECCV*, Part IV, LNCS 7577, pages 611–625. Springer-Verlag, Oct. 2012. 8

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

[3] D. Chen, L. Yuan, J. Liao, N. Yu, and G. Hua. Stylebank: An explicit representation for neural image style transfer. In *CVPR*, 2017. 1, 2

[4] T. Q. Chen and M. Schmidt. Fast patch-based style transfer of arbitrary style. *arXiv preprint arXiv:1612.04337*, 2016. 1, 2, 3, 5, 6, 7, 8

[5] V. Dumoulin, J. Shlens, and M. Kudlur. A learned representation for artistic style. In *ICLR*, 2017. 1, 2

[6] A. A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 341–346. ACM, 2001. 2

[7] M. Elad and P. Milanfar. Style transfer via texture synthesis. *IEEE Trans. Image Process.*, 26(5):2338–2351, 2017. 2

[8] L. A. Gatys, A. S. Ecker, and M. Bethge. Texture synthesis using convolutional neural networks. In *NIPS*, pages 262–270, 2015. 1, 2

[9] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *CVPR*, pages 2414–2423, 2016. 1, 2, 6, 7, 8

[10] A. Gupta, J. Johnson, A. Alahi, and L. Fei-Fei. Characterizing and improving stability in neural style transfer. In *ICCV*, 2017. 1, 2

[11] 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

[12] X. Huang and S. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, 2017. 1, 2, 4, 5, 6, 7, 8

[13] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, pages 694–711. Springer, 2016. 1, 2, 6

[14] A. Kessy, A. Lewin, and K. Strimmer. Optimal whitening and decorrelation. *The American Statistician*, (just-accepted), 2017. 4

[15] J. E. Kyprianidis, J. Collomosse, T. Wang, and T. Isenberg. State of the "art": A taxonomy of artistic stylization techniques for images and video. *IEEE Trans. Vis. Comput. Graphics*, 19(5):866–885, 2013. 2

[16] C. Li and M. Wand. Combining markov random fields and convolutional neural networks for image synthesis. In *CVPR*, pages 2479–2486, 2016. 1, 2, 3

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

[18] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang. Diversified texture synthesis with feed-forward networks. In *CVPR*, 2017. 1, 2

[19] 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, 4, 6, 7, 8

[20] Y. Li, N. Wang, J. Liu, and X. Hou. Demystifying neural style transfer. In *IJCAI*, pages 2230–2236, 2017. 1, 2

[21] J. Liao, Y. Yao, L. Yuan, G. Hua, and S. B. Kang. Visual attribute transfer through deep image analogy. *arXiv preprint arXiv:1705.01088*, 2017. 2

[22] 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 *ECCV*, pages 740–755. Springer, 2014. 6

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

[24] M. Ruder, A. Dosovitskiy, and T. Brox. Artistic style transfer for videos. In *German Conference on Pattern Recognition*, pages 26–36. Springer, 2016. 2

[25] F. Shen, S. Yan, and G. Zeng. Meta networks for neural style transfer. *arXiv preprint arXiv:1709.04111*, 2017. 1, 2

[26] Y. Shih, S. Paris, C. Barnes, W. T. Freeman, and F. Durand. Style transfer for headshot portraits. *TOG*, 2014. 2

[27] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 5

[28] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. S. Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *ICML*, pages 1349–1357, 2016. 1, 2

[29] 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. 1, 2

[30] 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. 1, 2

[31] H. Zhang and K. Dana. Multi-style generative network for real-time transfer. *arXiv preprint arXiv:1703.06953*, 2017. 1, 2