Neural Style Transfer: A Review

Yongcheng Jing† Yezhou Yang‡ Zunlei Feng† Jingwen Ye† Yizhou Yu† Mingli Song†*

† Microsoft Visual Perception Laboratory, College of Computer Science and Technology, Zhejiang University

‡ School of Computing, Informatics, and Decision Systems Engineering, Arizona State University

{ycjing,zunleifeng,yejingwen,brooksong}@zju.edu.cn yz.yang@asu.edu yizhouy@acm.org

Abstract

The seminal work of Gatys et al. demonstrated the power of Convolutional Neural Networks (CNN) in creating artistic imagery by separating and recombining image content and style. This process of using CNN to render a content image in different styles is referred to as Neural Style Transfer (NST). Since then, NST has become a trending topic both in academic literature and industrial applications. It is receiving increasing attention and a variety of approaches are proposed to either improve or extend the original NST algorithm. This review aims to provide an overview of the current progress towards NST, as well as discussing its various applications and open problems for future research. A list of papers discussed in this review, corresponding codes, pre-trained models and more comparison results are publicly available at: https://github.com/ycjing/Neural-Style-Transfer-Papers.

1. Introduction

Painting is a popular form of art. For thousands of years, people have been attracted by the art of painting with the advent of many fantastic artworks, *e.g.*, Van Gogh's "The Starry Night". In the past, re-drawing an image in a particular style requires a well-trained artist and lots of time.

Since the mid-1990s, the art theories behind the fantastic artworks have been attracting the attention of not only the artists but many computer science researchers. There are plenty of studies exploring how to automatically turn images into synthetic artworks. Among these studies, the advances in *Non-photorealistic Rendering* (NPR) [1, 2, 3] are inspiring, and nowadays, it is a firmly established field in the community of computer graphics. However, most of these NPR stylisation algorithms are designed for particular artistic styles [3, 4] and cannot be easily extended to other styles. While in the community of computer vision, style transfer is usually studied as a generalised problem of texture synthesis, which is to extract and transfer the texture from the source to target. However, only low-level features are considered during this process and the results are usually not that impressive.

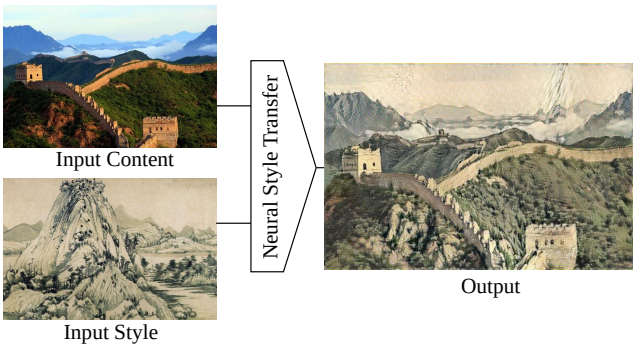


Figure 1. Example of Neural Style Transfer algorithm to transfer the style of a Chinese painting onto a given photograph. The style image is named "Dwelling in the Fuchun Mountains" by Gongwang Huang.

Recently, inspired by the power of *Convolutional Neural Network (CNN)*, Gatys *et al.* [5] first studied how to use CNN to reproduce famous painting styles on natural images. They proposed to model the *content* of a photo as the feature responses from a pre-trained CNN, and further model the *style* of an artwork as the summary feature statistics. Their experimental results demonstrated that the *content* and *style* in a photo were separable, which indicates the probability of changing a photo's *style* while preserving desired *semantic content*. Based on this finding, Gatys

*The corresponding author

et al. [5] first proposed to exploit CNN feature activations to separate and recombine the *content* of a given photo and the *style* of famous artworks. The key idea behind their algorithm is to iteratively optimise an image with the objective of matching desired CNN feature distribution, which involves both the photo's *content* information and artwork's *style* information. Their proposed algorithm successfully produces fantastic stylised images with the appearance of a given artwork. Figure 1 shows an example of transferring the style of a Chinese painting "Dwelling in the Fuchun Mountains" onto a photo of The Great Wall. Since the algorithm of Gatys *et al*. does not have any explicit restrictions on the type of style images, it breaks the constraints of previous approaches. The work of Gatys *et al*. opened up a new field called *Neural Style Transfer (NST)*, which is the process of using *Convolutional Neural Network* to render a content image in different styles.

The seminal work of Gatys *et al*. has attracted wide attention from both academia and industry. In academia, lots of follow-up studies were conducted to either improve or extend this algorithm and before long, these techniques were applied to many successful industrial applications (*e.g*., *Prisma* [6], *Ostagram* [7], *Deep Forger* [8]). However, there is no comprehensive survey summarising and discussing recent advances as well as challenges within this new field of Neural Style Transfer.

In this paper, we aim to provide an overview of current advances (up to March 2018) in Neural Style Transfer (NST). Our contributions are threefold. First, we investigate, classify and summarise recent advances in the field of NST. Second, we present several evaluation methods and experimentally compare different NST algorithms. Third, we summarise current challenges in this field and propose the corresponding possible solutions.

The organisation of this paper is as follows. We start our discussion with a brief review of pre-neural artistic style transfer methods in Section 2. Then Section 3 explores the derivations and foundations of NST. Based on the discussions in Section 3, we categorise and explain existing NST algorithms in Section 4. Some improvement strategies for these methods and their extensions will be given in Section 5. Section 6 presents several methodologies for evaluating NST algorithms and aims to build a standardised benchmark for follow-up studies. Then we demonstrate the commercial applications of NST in Section 7, including both current successful usages and its potential applications. In Section 8, we summarise current challenges in the field of NST, as well as propose the corresponding possible solutions. Finally, Section 9 concludes the paper, and Section 10 delineates several promising directions for future research.

2. Style Transfer in Pre-neural Era

Artistic style transfer is a long-standing research topic. Due to its wide variety of applications, it has been an important research area for more than two decades. Before the appearance of Neural Style Transfer (NST), the related researches in computer graphics have expanded into an area called Non-photorealistic Rendering (NPR). In the field of computer vision, style transfer is often considered as a generalised problem of texture synthesis. In this section, we briefly review some of these pre-neural artistic style transfer algorithms. For a more comprehensive overview, we recommend [3, 9, 10].

Style Transfer via Stroke-based Rendering. Stroke-based rendering (SBR) refers to a process of placing virtual strokes (*e.g*., brush strokes, tiles, stipples) upon a digital canvas to render a photograph with a particular style [11]. The process of SBR is generally starting from a source photo, incrementally compositing strokes to match the photo, and finally producing a non-photorealistic imagery, which looks like the photo but with an artistic style. During this process, an objective function is designed to guide the greedy or iterative placement of strokes. Despite the effectiveness of a large body of SBR algorithms, they are usually designed for a particular style (*e.g*., oil paintings, watercolours, sketches), which is not that flexible.

Style Transfer via Image Analogy. Image analogy aims to learn a mapping between a pair of source images and target stylised images in a supervised manner [12]. The training set of image analogy comprises pairs of unstylised source images and the corresponding stylised images with a particular style. Image analogy algorithm then learns the analogous transformation from the example training pairs, and creates analogous stylised result when given a test input photograph. Image analogy can also be extended in various ways, *e.g*., to learn stroke placements for portrait painting rendering [13]. In general, image analogy is effective for a variety of artistic styles. However, pairs of training data are usually unavailable in practice.

Style Transfer via Image Filtering. Creating an artistic image is actually a process that aims for image simplification and abstraction. Therefore, it is natural to consider adopting and combining some related image processing filters to render a given photo. For example, in [14], Winnemöller *et al*. for the first time exploit bilateral [15] and difference of Gaussians filters [16] to automatically produce cartoon-like effect. In general, image filtering based rendering algorithms are straightforward to implement and efficient in practice. At an expense, they are very limited in style diversity.

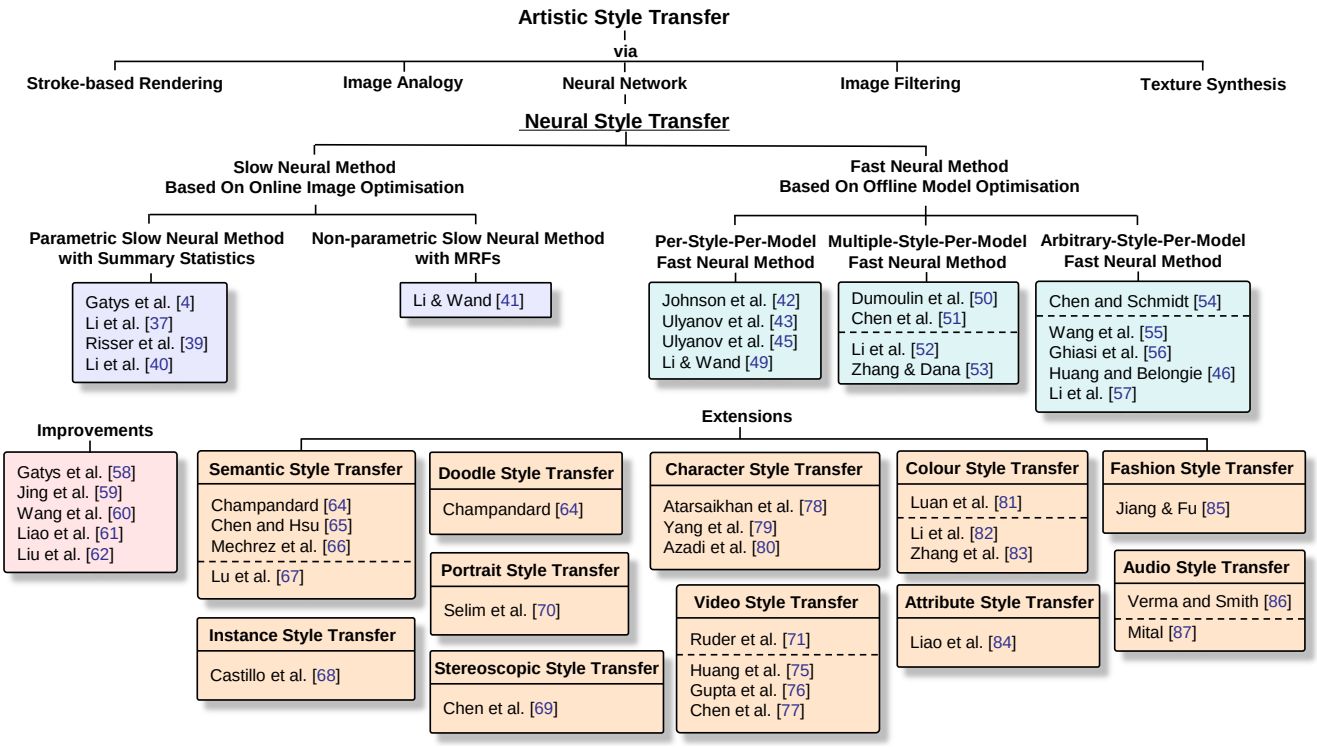


Figure 2. A taxonomy of artistic style transfer techniques.

Style Transfer via Texture Synthesis. Textures are repeated visual patterns in an image. Texture synthesis is a process which grows similar textures in the source texture image. It has long been an active research topic in computer vision [17, 18]. Given the distribution $p(I_s)$ of a texture instance I_s, the process of texture synthesis can be considered to draw a sample I from a certain distribution:

$$I \sim p(I|I_s). \tag{1}$$

Texture synthesis is very related to style transfer, since one can consider style as a kind of texture. In that sense, style transfer is actually a process of texture transfer, which constrains the given semantic content of the image I_c while synthesizing textures:

$$I \sim p(I|I_c, I_s). \tag{2}$$

This view has already been proposed in [17], which is one of the earliest works on texture synthesis. After that, there are lots of works [19, 20] following this route, which are generally built upon patch matching and quilting techniques. Even recently, Frigo *et al*. [21] propose an effective style transfer algorithm fully based on traditional texture synthesis method. Their idea is to first divide the input image adaptively into suitable patches, search the optimal mapping from the candidate regions in the style image, and then apply bilinear blending and colour transfer to obtain the final stylised result. Another recent work in [22] proposes a series of steps to stylise images, which include similarity matching of content and style patches, patch aggregation, content fusion based on segmentation algorithm, *etc*. Built upon traditional texture synthesis techniques, style transfer can be performed in an unsupervised manner. However, these texture synthesis based algorithms only exploit low-level image features, which limits their performance.

3. Derivations of Neural Style Transfer

For a better understanding of the NST development, we start by introducing its derivations. To automatically transfer an artistic style, the first and most important issue is how to model and extract style from an image. Since style is also a form of texture, a straightforward way is to relate *Visual Style Modelling* back to previously well-studied *Visual Texture Modelling* methods. After obtaining the style representation, the next issue is how to reconstruct an image with desired style information while preserving its semantic content. There comes the *Image Reconstruction* techniques.

3.1. Visual Texture Modelling

Visual texture modelling [23] is previously studied as the heart of texture synthesis. Throughout the history, there are two distinct approaches to model visual textures, which are

Parametric Texture Modelling with Summary Statistics and *Non-parametric Texture Modelling with MRFs*.

1) Parametric Texture Modelling with Summary Statistics. One path towards texture modelling is to capture image statistics from a sample texture and exploit summary statistical property to model the texture. The idea is first proposed by Julesz [24], which models textures as pixel-based N-th order statistics. Later work in [25] exploits filter responses to analyze textures, instead of direct pixel-based measurements. After that, Portilla and Simoncelli [26] further introduce a texture model based on multi-scale orientated filter responses and use gradient descent to improve synthesised results. A more recent parametric texture modelling approach proposed by Gatys *et al*. [27] is the first to measure summary statistics in the domain of Convolutional Neural Network (CNN). They design a Gram-based representation to model textures, which is the correlations between filter responses in different layers of a pre-trained classification network (VGG network) [28]. More specifically, the Gram-based representation encodes the second order statistics of the set of CNN filter responses. Next, we will explain this representation in detail for the usage of the following sections.

Assume that the feature map of a sample texture image I_s at layer l of VGG network is $\mathcal{F}^l(I_s) \in \mathbb{R}^{C\times H\times W}$, where C is the number of channels, and H and W represent the height and width of the feature map $\mathcal{F}(I_s)$. Then the Gram-based representation can be obtained by computing Gram matrix $\mathcal{G}(\mathcal{F}^l(I_s)') \in \mathbb{R}^{C\times C}$ over the feature map $\mathcal{F}^l(I_s)' \in \mathbb{R}^{C\times(HW)}$ (a reshaped version of $\mathcal{F}^l(I_s)$):

$$\mathcal{G}(\mathcal{F}^l(I_s)') = [\mathcal{F}^l(I_s)'][\mathcal{F}^l(I_s)']^T. \tag{3}$$

This Gram-based texture representation from CNN is effective at modelling wide varieties of both natural and non-natural textures. However, Gram-based representation is designed to capture global statistics and tosses spatial arrangements, which leads to unsatisfying results for modelling regular textures with long-range symmetric structures. To address this problem, Berger and Memisevic [29] propose to horizontally and vertically translate feature map by δ pixels to correlate the feature at position (i, j) with those at positions $(i + \delta, j)$ and $(i, j + \delta)$. In this way, the representation incorporates spatial arrangement information and is therefore more effective at modelling textures with symmetric properties.

2) Non-parametric Texture Modelling with MRFs. Another notable texture modelling methodology is to use non-parametric resampling. A variety of non-parametric methods are based on Markov Random Fields (MRFs) model, which assumes that in a texture image, each pixel is entirely characterised by its spatial neighbourhood. Under this assumption, Efros and Leung [17] propose to synthesise each pixel one by one by searching similar neighbourhoods in the source texture image and assigning the corresponding pixel. Their work is one of the earliest non-parametric algorithms with MRFs. Following their work, Wei and Levoy [18] further speed up the neighbourhood matching process by always using a fixed neighbourhood.

3.2. Image Reconstruction

In general, an essential step for many vision tasks is to extract an abstract representation from the input image. Image reconstruction is actually a reverse process, which is to reconstruct the whole input image from the extracted image representation. It is previously studied to analyse particular image representation and discover what information is contained in the abstract representation. Here our major focus is on CNN representation based image reconstruction algorithms, which can be categorised into *Slow Image Reconstruction based on Online Image Optimisation* and *Fast Image Reconstruction based on Offline Model Optimisation*.

1) Slow Image Reconstruction based on Online Image Optimisation. The first algorithm to reverse CNN representations is proposed by Mahendran and Vedaldi [30, 31]. Given a CNN representation to be reversed, their algorithm iteratively optimises an image (generally starting from random noise) until it has similar desired CNN representation. The iterative optimisation process is based on gradient descent in image space. Therefore, the process is time-consuming especially when the desired reconstructed image is large.

2) Fast Image Reconstruction based on Offline Model Optimisation. To address the efficiency issue of [30, 31], Dosovitskiy and Brox [32] propose to train a feed-forward network in advance and put the computational burden at training stage. At testing stage, the reverse process can be simply done with a network forward pass. Their algorithm significantly speeds up the image reconstruction process. In their later work [33], they further combine Generative Adversarial Network (GAN) [34] to improve the results.

4. A Taxonomy of Neural Style Transfer Algorithms

Neural Style Transfer (NST) is a subset of the large body of aforementioned artistic style transfer, as shown in Figure 2. It actually denotes the group *Style Transfer via Neural Network*. One can also say that NST is a combination of *Style Transfer via Texture Synthesis* and *Convolutional Neural Network*. In this section, we provide a categorisation of NST algorithms. Current NST methods fit into one

of two categories, *Slow Neural Method Based On Online Image Optimisation* and *Fast Neural Method Based On Offline Model Optimisation*. The first category transfers the style by iteratively optimising an image, *i.e.*, algorithms belong to this category are built upon *Slow Image Reconstruction* techniques. The second category optimises a generative model offline and produces the stylised image with a single forward pass, which actually exploits the idea of *Fast Image Reconstruction* techniques.

4.1. Slow Neural Method Based On Online Image Optimisation

DeepDream [35] is the first attempt to produce artistic images by reversing CNN representations with *Slow Image Reconstruction* techniques. By further combining *Visual Texture Modelling* techniques to model style, *Slow Neural Methods Based On Online Image Optimisation* are subsequently proposed, which build the early foundations for the field of NST. Their basic idea is to first model and extract style and content information from the corresponding style and content images, recombine them as the target representation, and then iteratively reconstruct a stylised result that matches the target representation. In general, different *Slow Neural Methods* share the same *Slow Image Reconstruction* technique, but differ in the way they model the visual style, which is built on the aforementioned two categories of *Slow Visual Texture Modelling* techniques.

4.1.1 Parametric Slow Neural Method with Summary Statistics

The first subset of Slow Neural Methods is based on *Parametric Texture Modelling with Summary Statistics*. The style is characterised as a set of spatial summary statistics.

We start by introducing the first NST algorithm proposed by Gatys *et al*. [5, 4]. By reconstructing representations from intermediate layers in VGG network, Gatys *et al*. observe that deep convolutional neural network is capable of extracting semantic image content from an arbitrary photograph and some appearance information from the well-known artwork. According to this observation, they build the content component of the newly stylised image by penalising the difference of high-level representations derived from content and stylised images, and further build the style component by matching Gram-based summary statistics of style and stylised images, which is derived from their proposed texture modelling technique [27] (Section 3.1). The details of their algorithm are as follows.

Given a content image I_c and a style image I_s, the algorithm in [4] tries to seek a stylised image I that minimises the following objective:

$$\begin{aligned} I^* &= \arg\min_I \mathcal{L}_{total}(I_c, I_s, I) \\ &= \arg\min_I \ \alpha\mathcal{L}_c(I_c, I) + \beta\mathcal{L}_s(I_s, I), \end{aligned} \tag{4}$$

where $\mathcal{L}_c$ compares the content representation of a given content image to that of the (yet unknown) stylised image, and $\mathcal{L}_s$ compares the Gram-based style representation derived from a style image to that of the (yet unknown) stylised image. α and β are used to balance the content component and style component in the stylised result.

The content loss $\mathcal{L}_c$ is defined by the squared Euclidean distance between the feature representations $\mathcal{F}^l$ of the content image I_c in layer l and that of the (yet unknown) stylised image I:

$$\mathcal{L}_c = \sum\nolimits_{l\in\{l_c\}} \|\mathcal{F}^l(I_c) - \mathcal{F}^l(I)\|^2, \tag{5}$$

where $\{l_c\}$ denotes the set of VGG layers for computing the content loss. For the style loss $\mathcal{L}_s$, [4] exploits Gram-based visual texture modelling technique to model the style, which has already been explained in Section 3.1. Therefore, the style loss is defined by the squared Euclidean distance between the Gram-based style representations of I_s and I:

$$\mathcal{L}_s = \sum\nolimits_{l\in\{l_s\}} \|\mathcal{G}(\mathcal{F}^l(I_s)') - \mathcal{G}(\mathcal{F}^l(I)')\|^2, \tag{6}$$

where $\mathcal{G}$ is the aforementioned Gram matrix to encode the second order statistics of the set of filter responses. $\{l_s\}$ represents the set of VGG layers for calculating the style loss. The choice of $\{l_c\}$ and $\{l_c\}$ empirically follows the principle that the usage of lower layer tends to retain low-level features (*e.g.*, colours), while the usage of higher layer generally preserves more high-level semantic content information. Therefore, $\mathcal{L}_s$ is usually computed with lower layers and $\mathcal{L}_c$ is computed with higher layers. Given the pre-trained VGG-19 [28] as the loss network, Gatys et al.'s choice in [4] is $\{l_c\} = \{relu1\_1, relu2\_1, relu3\_1, relu4\_1, relu5\_1\}$ and $\{l_s\} = \{relu4\_2\}$. Also, VGG loss network is not the only option. Similar performance can be achieved by selecting other pre-trained classification networks, *e.g.*, ResNet [36].

In Equation (4), both $\mathcal{L}_c$ and $\mathcal{L}_s$ are differentiable. Thus, with random noise as the initial I, Equation (4) can be minimised by using gradient descent in image space with back-propagation. In addition, a total variation denoising term is usually added to encourage the smoothness in the stylised result in practice.

Gram-based style representation is not the only choice to statistically encode style information. There are also some other effective statistical style representations, which are derived from Gram-based representation. Li *et al*. [37]

derive some different style representations by considering style transfer in the domain of transfer learning, or more specifically, *domain adaption* [38]. Given that training and testing data are drawn from different distributions, the goal of domain adaption is to adapt a model trained on labelled training data from a source domain to predict labels of unlabelled testing data from a target domain. One way for domain adaption is to match a sample in the source domain to that in the target domain by minimising their distribution discrepancy, in which *Maximum Mean Discrepancy (MMD)* is a popular choice to measure the discrepancy between two distributions. Li *et al*. prove that matching Gram-based style representations between a pair of style and stylised images is intrinsically minimising MMD with a quadratic polynomial kernel. Therefore, it is expected that other kernel functions for MMD can be equally applied in NST, *e.g*., the linear kernel, polynomial kernel and Gaussian kernel. Another related representation is BN statistic representation, which is to use mean and variance of the feature maps in VGG layers to model style:

$$\mathcal{L}_s = \sum\nolimits_{l\in\{l_s\}} \frac{1}{C^l}\sum_{c=1}^{C^l} \|\mu(\mathcal{F}_c^l(I_s)) - \mu(\mathcal{F}_c^l(I))\|^2 + \|\sigma(\mathcal{F}_c^l(I_s)) - \sigma(\mathcal{F}_c^l(I))\|^2, \quad (7)$$

where $\mathcal{F}_c^l \in \mathbb{R}^{H\times W}$ is the c-th feature map channel at layer l of VGG network, and C^l is the number of channels.

However, the Gram-based algorithm has the limitation of instabilities during optimisations. Also, it requires manually tuning the parameters, which is very tedious. Risser *et al*. [39] find that feature activations with quite different means and variances can still have the same Gram matrix, which is the main reason of instabilities. Inspired by this observation, Risser *et al*. introduce an extra histogram loss, which guides the optimisation to match the entire histogram of feature activations. They also present a preliminary solution to automatic parameter tuning, which is to explicitly prevent gradients with extreme values through extreme gradient normalisation.

All these aforementioned neural methods only compare content and stylised images in the CNN feature space to make the stylised image semantically similar to the content image. But since CNN features inevitably lose some low-level information contained in the image, there are usually some unappealing distorted structures and irregular artefacts in the stylised results. To preserve the coherence of fine structures during stylization, Li *et al*. [40] propose to incorporate additional constraints upon low-level features in pixel space. They introduce an additional Laplacian loss, which is defined as the squared Euclidean distance between the Laplacian filter responses of a content image and stylised result. Laplacian filter computes the second order derivatives of the pixels in an image and is widely used for edge detection.

4.1.2 Non-parametric Slow Neural Method with MRFs

Non-parametric Slow Neural Method is built on the basis of *Non-parametric Texture Modelling with MRFs*. This category considers NST at a local level, *i.e*., operating on patches to match the style.

Li and Wand [41] are the first to propose an MRF-based NST algorithm. They find that the parametric NST method with summary statistics only captures the per-pixel feature correlations and does not constrain the spatial layout, which leads to a less visual plausibility result for photo-realistic styles. Their solution is to model the style in a non-parametric way and introduce a new style loss function which includes a patch-based MRF prior:

$$\mathcal{L}_s = \sum\nolimits_{l\in\{l_s\}} \sum_{i=1}^{m} \|\Psi_i(\mathcal{F}^l(I)) - \Psi_{NN(i)}(\mathcal{F}^l(I_s))\|^2, \quad (8)$$

where $\Psi(\mathcal{F}^l(I))$ is the set of all local patches from the feature map $\mathcal{F}^l(I)$. Ψ_i denotes the i^{th} local patch and $\Psi_{NN(i)}$ is the most similar style patch with the i-th local patch in the stylised image I. The best matching $\Psi_{NN(i)}$ is obtained by calculating normalised cross-correlation over all the style patches in the style image I_s. m is the total number of local patches. Since their algorithm matches a style in the patch-level, the fine structure and arrangement can be preserved much better. Given a photograph as the content, their algorithm achieves remarkable results, especially for photo-realistic styles.

4.2. Fast Neural Method Based On Offline Model Optimisation

Although the *Slow Neural Method Based On Online Image Optimisation* is able to yield impressive stylised images, there are still some limitations. The most concerned limitation is the efficiency issue. The second category *Fast Neural Method* addresses the speed and computational cost issue by exploiting *Fast Image Reconstruction based on Offline Model Optimisation* to reconstruct the stylised result, *i.e*., a feed-forward network g is optimised over a large set of images I_c for one or more style images I_s:

$$\theta^* = \arg\min_{\theta} \mathcal{L}_{total}(I_c, I_s, g_{\theta^*}(I_c)),\ I^* = g_{\theta^*}(I_c). \quad (9)$$

Depending on the number of artistic styles a single g can produce, *Fast Neural Methods* are further divided into *Per-Style-Per-Model* Fast Neural Method (PSPM), *Multiple-Style-Per-Model* Fast Neural Method (MSPM), and *Arbitrary-Style-Per-Model* Fast Neural Method (ASPM).

4.2.1 Per-Style-Per-Model Fast Neural Method

1) Parametric PSPM with Summary Statistics. The first two *Fast Neural Methods* are proposed by Johnson *et al*. [42] and Ulyanov *et al*. [43] respectively. These two methods share a similar idea, which is to pre-train a feed-forward style-specific network and produce a stylised result with a single forward pass at testing stage. They only differ in the network architecture, for which Johnson *et al*.'s design roughly follows the network proposed by Radford *et al*. [44] but with residual blocks as well as fractionally strided convolutions, and Ulyanov *et al*. use a multi-scale architecture as the generator network. The objective function is similar to the algorithm of Gatys *et al*. [4], which indicates that they are also *Parametric Methods with Summary Statistics*.

Shortly after [42, 43], Ulyanov *et al*. [45] further find that simply applying normalisation to every single image rather than a batch of images (precisely *Batch Normalization (BN)*) leads to a significant improvement in stylisation quality. This single image normalisation is called *Instance Normalisation* (IN), which is actually equivalent to batch normalisation when the batch size is set to 1. The style transfer network with IN is shown to converge faster than BN and also achieves visually better results. One interpretation is that IN is actually a form of style normalisation and can directly normalise the style of each content image to the desired style [46]. Therefore, the objective is easier to learn as the rest of the network only needs to take care of the content loss. *Instance Normalisation* (IN) also intrinsically shares a similar idea with *Adaptive Batch Normalisation* (AdaBN) [47, 48], which uses domain-specific mean and variance statistics in BN layers for domain adaption.

2) Non-parametric PSPM with MRFs. Another work by Li and Wand [49] is inspired by the MRF-based NST [41] algorithm in Section 4.1.2. They address the efficiency issue by training a Markovian feed-forward network using adversarial training. Similar to [41], their algorithm is a *Patch-based Non-parametric Method with MRFs*. Their method is shown to outperform the algorithms of Johnson *et al*. and Ulyanov *et al*. in the preservation of coherent textures in complex images, thanks to their patch-based design.

4.2.2 Multiple-Style-Per-Model Fast Neural Method

Although the above PSPM approaches can produce stylised images two orders of magnitude faster than previous slow NST methods, separate generative networks have to be trained for each particular style image, which is quite time-consuming and inflexible. But many paintings (*e.g*., impressionist paintings) actually share similar paint strokes and only differ in their colour palettes. Intuitively, it is redundant to train a separate network for each of them. MSPM is therefore proposed, which improves the flexibility of PSPM by further incorporating multiple styles into one single model. There are generally two paths towards handling this problem: 1) tying only a small number of parameters in a network to each style ([50, 51]) and 2) still exploiting only a single network like PSPM but combining both style and content as inputs ([52, 53]).

1) Tying only a small number of parameters to each style. An early work by Dumoulin *et al*. [50] is built on the basis of the proposed IN layer in PSPM algorithm [45] (Section 4.2.1). They surprisingly find that using the same convolutional parameters but only scaling and shifting parameters in IN layers is sufficient to model different styles. Therefore, they propose an algorithm to train a conditional multi-style transfer network based on conditional instance normalisation (CIN), which is defined as:

$$\text{CIN}(\mathcal{F}(I_c), s) = \gamma^s \left(\frac{\mathcal{F}(I_c) - \mu(\mathcal{F}(I_c))}{\sigma(\mathcal{F}(I_c))} \right) + \beta^s, \quad (10)$$

where $\mathcal{F}$ is the input feature activation and s is the index of the desired style from a set of style images. As shown in Equation (10), the conditioning for each style I_s is done by scaling and shifting parameters γ^s and β^s after normalising feature activation $\mathcal{F}(I_c)$, *i.e*., each style I_s can be achieved by tuning parameters of an affine transformation. The interpretation is similar to that for [45] in Section 4.2.1, *i.e*., the normalisation of feature statistics with different affine parameters can normalise input content image to different styles. Furthermore, the algorithm of Dumoulin *et al*. can also be extended to combine multiple styles in a single stylised result by combining affine parameters of different styles.

Another algorithm which follows the first path of MSPM is proposed by Chen *et al*. [51]. Their idea is to explicitly decouple style and content, *i.e*., using separate network components to learn the corresponding content and style information. More specifically, they use mid-level convolutional filters (called "StyleBank" layer) to individually learn different styles. Each style is tied to a set of parameters in "StyleBank" layer. The rest components in the network are used to learn semantic content information, which is shared by different styles. Their algorithm also supports flexible incremental training, which is to fix the content components in the network and only train "StyleBank" layer for the newly coming style.

2) Combining both style and content as inputs. One disadvantage of the first category is that the model size generally becomes larger with the increase of the number of learned styles. The second path of MSPM addresses this limitation by fully exploring the capability of one single network and combining both content and style into the network

for style identification. Different MSPM algorithms differ in the way to incorporate style into the network.

Li *et al*. [52] propose to first sample a set of noise maps from a uniform distribution and establish a one-to-one mapping f between each style and noise map. For clarity, we divide the style transfer network into an encoder (Enc) and decoder (Dec) pair. For each style, the corresponding noise map $f(I_s)$ is concatenated ($\oplus$) with the encoded feature activations $Enc(I_c)$ and then feeded into the decoder to get the stylised result: $I = Dec(\ f(I_s)\ \oplus\ Enc(I_c)\ )$.

In [53], Zhang and Dana first forward each style image in the style set through the pre-trained VGG network and obtain multi-scale feature activations $\mathcal{F}(I_s)$ in different VGG layers. Then multi-scale $\mathcal{F}(I_s)$ are combined with multi-scale encoded features $Enc(I_c)$ from different layers in the encoder through their proposed inspiration layers. The inspiration layers are designed to reshape $\mathcal{F}(I_s)$ to match the desired dimension, and also have a learnable weight matrix to tune feature maps to help minimise the objective function.

4.2.3 Arbitrary-Style-Per-Model Fast Neural Method

Arbitrary-Style-Per-Model Fast Neural Method (ASPM) aims at one-model-for-all, *i.e*., one single trainable model to transfer arbitrary artistic styles. There are also two types of ASPM, one built upon *Non-parametric Texture Modelling with MRFs* and the other one built upon *Parametric Texture Modelling with Summary Statistics*.

1) Non-parametric ASPM with MRFs. The first ASPM algorithm is proposed by Chen and Schmidt [54]. They first extract a set of activation patches from content and style feature activations computed in pre-trained VGG network. Then they match each content patch to the most similar style patch and swap them (called "Style Swap" in [54]). The stylised result can be produced by reconstructing the resulting activation map after "Style Swap", with either *Slow Image Reconstruction based on Online Image Optimisation* or *Fast Image Reconstruction based on Offline Model Optimisation*.

2) Parametric ASPM with Summary Statistics. Considering [50] in Section 4.2.2, the simplest approach for arbitrary style transfer is to train a separate parameter prediction network P to predict γ^s and β^s in Equation (10) with a number of training styles. The idea is first proposed by Wang *et al*. [55] and then in [56] by Ghiasi *et al*. Given a test style image I_s, CIN layers in the style transfer network take affine parameters γ^s and β^s from $P(I_s)$, and normalise the input content image to the desired style with a forward pass.

Another similar approach based on [50] is proposed by Huang and Belongie [46]. Instead of training a parameter prediction network, Huang and Belongie propose to modify conditional instance normalisation (CIN) in Equation (10) to adaptive instance normalisation (AdaIN):

$$\text{AdaIN}(\mathcal{F}(I_c), \mathcal{F}(I_s)) = \sigma(\mathcal{F}(I_s)) \left(\frac{\mathcal{F}(I_c) - \mu(\mathcal{F}(I_c))}{\sigma(\mathcal{F}(I_c))} \right) + \mu(\mathcal{F}(I_s)). \quad (11)$$

AdaIN transfers the channel-wise mean and variance feature statistics between content and style feature activations, which also shares a similar idea with [54]. Different from [50], the encoder in the style transfer network of [46] is fixed and comprises the first few layers in pre-trained VGG network. Therefore, $\mathcal{F}$ in [46] is actually the feature activation from a pre-trained VGG network. The decoder part needs to be trained with a large set of style and content images to decode resulting feature activations after AdaIN to the stylised result: $I = Dec(\ \text{AdaIN}(\mathcal{F}(I_c), \mathcal{F}(I_s))\ )$.

A more recent work by Li *et al*. [57] attempts to exploit a series of feature transformations to transfer arbitrary artistic style in a style learning free manner. Similar to [46], Li *et al*. use the first few layers of pre-trained VGG as the encoder and train the corresponding decoder. But they replace the AdaIN layer [46] in between the encoder and decoder with a pair of whitening and colouring transformations (WCT): $I = Dec(\ \text{WCT}(\mathcal{F}(I_c), \mathcal{F}(I_s))\ )$. Their algorithm is built on the observation that the whitening transformation can remove the style related information and preserve the structure of content. Therefore, receiving content activations $\mathcal{F}(I_c)$ from the encoder, whitening transformation can filter the original style out of the input content image and return a filtered representation with only content information. Then, by applying colouring transformation, the style patterns contained in $\mathcal{F}(I_s)$ are incorporated into the filtered content representation, and the stylised result I can be obtained by decoding the transformed features. They also extend this single-level stylisation to multi-level stylisation to further improve visual quality.

5. Improvements and Extensions

Since the boom of Neural Style Transfer (NST), there are also some researches devoted to improving current NST algorithms by controlling perceptual factors (*e.g*., stroke size control, spatial style control, and colour control). Also, all of aforementioned NST methods are designed for general still images. They may not be appropriate for other types of images (*e.g*., doodles, head portraits, and video frames). Thus, a variety of follow-up studies aim to extend general NST algorithms to these particular types of images and even extend them beyond artistic image style (*e.g*., audio style).

(a) Content (b) Style (c) Small Stroke Size (d) Large Stroke Size

Figure 3. Control the brush stroke size in "Slow" Neural Style Transfer. (c) is the output with smaller brush size and (d) with larger brush size. The style image is "The Starry Night" by Vincent van Gogh.

Controlling Perceptual Factors in Neural Style Transfer. Gatys *et al*. themselves [58] propose several slight modifications to improve their previous algorithm [4]. They demonstrate a spatial style control strategy, which is to define a guidance map for the feature activations, where the desired region (getting the style) is assigned 1 and otherwise, 0. While for the colour control, the origin NST algorithm produces stylised images with the colour distribution of the style image. However, sometimes people prefer a colour-preserving style transfer, *i.e*., preserving the colour of the content image during style transfer. The corresponding solution is to first transform the style image's colours to match the content image's colours before style transfer, or alternatively perform style transfer only in the luminance channel.

For stroke size control, the problem is much more complex. We show sample results of stroke size control in Figure 3. The discussions of stroke size control strategy need to be split into several cases [59]:

1) Slow Neural Style Transfer with non-high-resolution images: Since current style statistics (*e.g*., Gram-based and BN-based statistics) are scale-sensitive [59], to achieve different stroke sizes, the solution is simply resizing a given style image to different scales.

2) Fast Style Transfer with non-high-resolution images: One possible solution is to resize the input image to different scales before the forward pass, which inevitably hurts stylisation quality. Another possible solution is to train multiple models with different scales of a style image, which is space and time consuming. Also, the possible solution fails to preserve *stroke consistency* among results with different stroke sizes, *i.e*., the results vary in stroke orientations, stroke configurations, *etc*. However, users generally desire to only change the stroke size but not others. To address this problem, Jing *et al*. [59] propose a stroke controllable PSPM algorithm. The core component of their algorithm is a *StrokePyramid* module, which learns different stroke sizes with adaptive receptive fields. Without trading off quality and speed, their algorithm is the first to exploit one single model to achieve flexible continuous stroke size control while preserving *stroke consistency*, and further achieve spatial stroke size control to produce new artistic effects. Although one can also use ASPM algorithm to control stroke size, ASPM trades off quality and speed. As a result, ASPM is not effective at producing fine strokes and details compared with [59].

3) Slow Neural Style Transfer with high-resolution images: For high-resolution images (*e.g*., 3000×3000 pixels in [58]), a large stroke size cannot be achieved by simply resizing style image to a large scale. Since only the region in the content image with a receptive field size of VGG can be affected by a neuron in the loss network, there is almost no visual difference between a large and larger brush strokes in a small image region with receptive field size. Gatys *et al*. [58] tackle this problem by proposing a coarse-to-fine Slow Style Transfer procedure with several steps of downsampling, stylising, upsampling and final stylising.

4) Fast Style Transfer with high-resolution images: Similar to 3), stroke size in stylised result does not vary with style image scale for high-resolution images. The solution is also similar to Gatys *et al*.'s algorithm in [58], which is a coarse-to-fine stylisation procedure [60]. The idea is to exploit a multimodel, which comprises multiple subnetworks. Each subnetwork receives the upsampled stylised result of the previous subnetwork as the input, and stylises it again with finer strokes.

Another limitation of current NST algorithms is that they do not consider the depth information contained in the image. To address this limitation, depth preserving NST algorithms [61, 62] are proposed. Their approach is to add a depth loss function based on [42] to measure the depth difference between the content image and the (yet unknown) stylised image. The image depth is acquired by applying a single-image depth estimation algorithm (*e.g*., Zoran et al.'s work in [63]).

Semantic Style Transfer. Given a pair of style and content images which are similar in content, the goal of semantic style transfer is to build a semantic correspondence between the style and content, which maps each style region to a corresponding semantically similar content region. Then

the style in each style region is transferred to the semantically similar content region.

1) Slow Semantic Style Transfer. Since the patch matching scheme naturally meets the requirements of the region-based correspondence, Champandard [64] proposes to build a semantic style transfer algorithm based on the aforementioned patch-based algorithm [41] (Section 4.1.2). Actually, the result produced by [41] is close to the target of semantic style transfer but without incorporating an accurate segmentation mask, which sometimes leads to a wrong semantic match. Therefore, Champandard augments an additional semantic channel upon [41], which is a downsampled semantic segmentation map. The segmentation map can be either manually annotated or from a semantic segmentation algorithm. Despite the remarkable results produced by [64], MRF-based design is not the only choice. Instead of combining MRF prior, Chen and Hsu [65] provide an alternative way for semantic style transfer, which is to exploit masking out process to constrain the spatial correspondence and also a higher order style feature statistic to further improve the result. More recently, Mechrez *et al*. [66] propose an alternative contextual loss to realise semantic style transfer in a segmentation-free manner.

2) Fast Semantic Style Transfer. As before, the efficiency issue is always a big issue. Both [64] and [65] are based on Slow NST algorithms and therefore leave much room for improvement. Lu *et al*. [67] speed up the process by optimising the objective function in feature space, instead of in pixel space. More specifically, they propose to do feature reconstruction, instead of image reconstruction as previous algorithms do. This optimisation strategy reduces the computation burden, since the loss does not need to propagate through a deep network. The resulting reconstructed feature is decoded into the final result with a trained decoder. Since the speed of [67] does not reach real-time, there is still big room for further research.

Instance Style Transfer. Instance style transfer is built on instance segmentation and aims to stylise only a single user-specified object within an image. The challenge mainly lies in the transition between a stylised object and non-stylised background. Castillo *et al*. [68] tackle this problem by adding an extra MRF-based loss to smooth and anti-alias boundary pixels.

Doodle Style Transfer. An interesting extension can be found in [64], which is to exploit NST to transform rough sketches into fine artworks. The method is simply discarding content loss term and using doodles as segmentation map to do semantic style transfer.

Stereoscopic Style Transfer. Driven by the demand of AR/VR, Chen *et al*. [69] propose a stereoscopic NST algorithm for stereoscopic images. They propose a disparity loss to penalise the bidirectional disparity. Their algorithm is shown to produce more consistent strokes for different views.

Portrait Style Transfer. Current style transfer algorithms are usually not appropriate for head portrait images. As they do not impose spatial constraints, directly applying these existing algorithms to head portraits will deform facial structures, which is unacceptable for the human visual system. Selim *et al*. [70] address this problem and extend [4] to head portrait painting transfer. They propose to use the notion of gain maps to constrain spatial configurations, which can preserve the facial structures while transferring the texture of the style image.

Video Style Transfer. NST algorithms for video sequences are substantially proposed shortly after Gatys et al.'s first NST algorithm for still images [4]. Different from still image style transfer, the design of video style transfer algorithm needs to consider the smooth transition between adjacent video frames. Like before, we divide related algorithms into Slow and Fast Video Style Transfer.

1) Slow Video Style Transfer based on Online Image Optimisation. The first video style transfer algorithm is proposed by Ruder *et al*. [71, 72]. They introduce a temporal consistency loss based on optical flow to penalise the deviations along point trajectories. The optical flow is calculated by using novel optical flow estimation algorithms [73, 74]. As a result, their algorithm eliminates temporal artefacts and produces smooth stylised videos. However, they build their algorithm upon [4] and need several minutes to process a single frame.

2) Fast Video Style Transfer based on Offline Model Optimisation. Several follow-up studies are devoted to stylising a given video in real-time. Huang *et al*. [75] propose to augment Ruder et al.'s temporal consistency loss [71] upon current PSPM algorithm. Given two consecutive frames, the temporal consistency loss is directly computed using two corresponding outputs of style transfer network to encourage pixel-wise consistency, and a corresponding two-frame synergic training strategy is introduced for the computation of temporal consistency loss. Another concurrent work which shares a similar idea with [75] but with an additional exploration of style instability problem can be found in [76]. Different from [75, 76], Chen *et al*. [77] propose a flow subnetwork to produce feature flow and incorporate optical flow information in feature space. Their algorithm is built on a pre-trained style transfer network (an encoder-decoder pair) and wraps feature activations from the pre-trained stylisation encoder using the obtained feature flow.

Character Style Transfer. Given a style image containing multiple characters, the goal of *Character Style Transfer* is to apply the idea of NST to generate new fonts and text effects. In [78], Atarsaikhan *et al*. directly apply the algorithm in [4] to font style transfer and achieve remarkable results. While Yang *et al*. [79] propose to first characterise style elements and exploit extracted characteristics to guide the generation of text effects. A more recent work [80] designs a conditional GAN model for glyph shape prediction, and also an ornamentation network for colour and texture prediction. By training these two networks jointly, font style transfer can be realised in an end-to-end manner.

Colour Style Transfer. Colour style transfer aims to transfer the style of colour distributions. The general idea is to build upon current semantic style transfer but to eliminate distortions and preserve the original structure of the content image.

1) Slow Colour Style Transfer. The earliest colour style transfer approach is proposed by Luan *et al*. [81]. They propose to add a photorealism regularization upon [64] to penalise image distortions. But since Luan et al.'s algorithm is built on an online image optimisation based *Slow Semantic Style Transfer* method [64], their algorithm is computationally expensive.

2) Fast Colour Style Transfer. Li *et al*. [82] address the efficiency issue of [81] by handling this problem with two steps, the stylisation step and smoothing step. The stylisation step is to apply the NST algorithm in [57] but replace upsampling layers with unpooling layers to produce the stylised result with fewer distortions. Then the smoothing step further eliminates structural artefacts. These two aforementioned algorithms [81, 82] are mainly designed for natural images. Another work in [83] proposes to exploit GAN to transfer the colour from human-designed anime images to sketches. Their algorithm demonstrates a promising application of Colour Style Transfer, which is the automatic image colourisation.

Attribute Style Transfer. Image attributes are generally referred to image colours, textures, *etc*. Previously, image attribute transfer is accomplished through image analogy [12] in a supervised manner (Section 2). Derived from the idea of patch-based NST [41], Liao *et al*. [84] propose a deep image analogy to study image analogy in the domain of CNN features. Their algorithm is based on patch matching technique and realises a weakly supervised image analogy, *i.e*., their algorithm only needs a single pair of source and target images instead of a large training set.

Fashion Style Transfer. Fashion style transfer receives fashion style image as the target and generates clothing images with desired fashion styles. The challenge of Fashion Style Transfer lies in the preservation of similar design with the basic input clothing while blending desired style patterns. This idea is first proposed by Jiang and Fu [85]. They tackle this problem by proposing a pair of fashion style generator and discriminator.

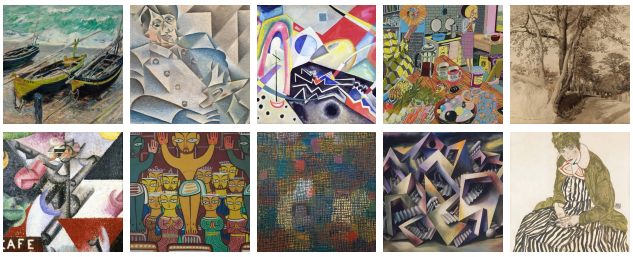


Figure 4. Diversified style images used in our experiment.

Audio Style Transfer. In addition to transferring image styles, [86, 87] extend the domain of image style to audio style, and synthesise new sounds by transferring the desired style from a target audio. The study of audio style transfer also follows the route of image style transfer, *i.e*., *Slow Audio Style Transfer* and then *Fast Audio Style Transfer*. Inspired by image optimisation based image style transfer, Verma and Smith [86] propose a *Slow Audio Style Transfer* algorithm based on online audio optimisation. They start from a noise signal and optimise it iteratively using back-propagation. [87] improves the efficiency by transferring an audio in a feed-forward manner and can produce the result in real-time.

6. Evaluation Methodology

The evaluations of NST algorithms remain an open and important problem in this field. In general, there are two major types of evaluation methodologies that can be employed in the field of NST, *i.e*., qualitative evaluation and quantitative evaluation. Qualitative evaluation relies on the aesthetic judgements of observers. The evaluation results are related to lots of factors (*e.g*., age and occupation of participants). While quantitative evaluation focuses on the precise evaluation metrics, which include time complexity, loss variation, *etc*.

In this section, we experimentally compare different NST algorithms both qualitatively and quantitatively. We hope our study can build a *standardised benchmark* for this area.

6.1. Experimental Setup

Evaluation datasets. Totally, there are ten style images and forty content images. For style images, we select artworks of diversified styles, as shown in Figure 4. For example, there are impressionism artwork, cubism artwork,

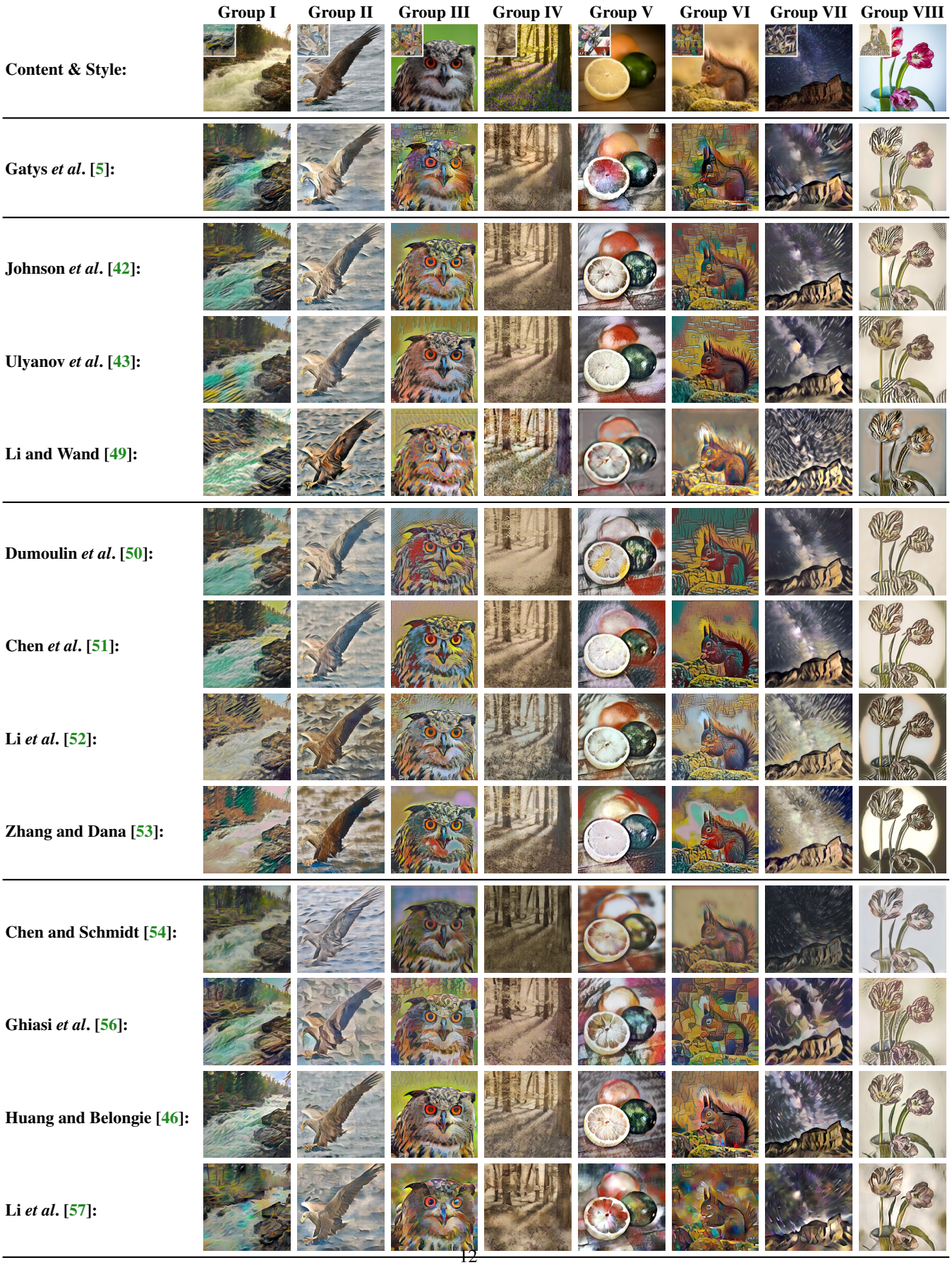


Figure 5. Some example results for qualitative evaluation. Content images are credited to flickr users b togol (I and VII), Kevin Robson (II), Abdulkader Imam (III), Elliot Hook (IV), Yoann JEZEQUEL (V), Elliot Hook (VI), and Eva (VIII). Style images are from Google Arts & Culture.

abstract artwork, contemporary artwork, futurism artwork, surrealist artwork, and expressionism artwork. Regarding the mediums, some of these artworks are painted on canvas, while others are painted on cardboard or wool, cotton, polyester, *etc*. For content images, we also try to select a wide variety of photos, which include animal photography, still life photography, landscape photography and portrait photography. All the images are never seen during training.

Principles. To maximise the fairness of the comparisons, we also obey the following principles during our experiment:

1) In order to cover every detail in each algorithm, we try to use the provided implementation from their published literatures. For [5], since there is no official implementation provided by the authors, we use a popular open source code [88] which is also admitted by the authors. Except for [50, 29] which are based on TensorFlow, all the other codes are based on Torch 7, which maximises the fairness especially for speed comparison.

2) Since the visual effect is influenced by the content and style weight, it is difficult to compare results with different degrees of stylisation. Simply giving the same content and style weight is not an optimal solution due to the different ways to calculate losses in each algorithm (*e.g*., different choices of content and style layers, different loss functions). Therefore, in our experiment, we try our best to balance the content and style weight among different algorithms.

3) We try to use the default parameters (*e.g*., choice of layers, learning rate, *etc*.) suggested by the authors except for the aforementioned content and style weight. Although the results for some algorithms may be further improved by more careful hyperparameter tuning, we select the authors' default parameters since we hold the point that the *sensitivity for hyperparameters* is also an important implicit criterion for comparison. For example, we cannot say an algorithm is effective if it needs heavy work to tune its parameters for each style.

There are also some other implementation details to be noted. For [42] and [43], we use the instance normalisation strategy proposed in [45], which is not covered in the published papers. Also, we do not consider the diversity loss term (proposed in [45, 52]) for all algorithms, *i.e*., one pair of content and style images corresponds to one stylised result in our experiment. For Chen and Schmidt's algorithm [54], we use the feed-forward reconstruction to reconstruct the stylised results.

6.2. Qualitative Evaluation

NST is an art creation process. It is difficult to define the aesthetic criterion for an artwork. Therefore, for the same stylised result, different people may have different or even opposite views. Here, we choose to present stylised results of different algorithms and leave the judgement to readers. Example stylised results are shown in Figure 5. More results can be found in the supplementary material[1]. In Figure 5, we build several blocks to separate results of different categories of NST algorithms.

1) Results of Slow Neural Style Transfer. Following the content & style images, the first block contains the results of Gatys et al.'s Slow NST algorithm based on online image optimisation [4]. The style transfer process is computationally expensive, but in contrast, the results are appealing in visual quality. Therefore, the algorithm of Gatys *et al*. is usually regarded as the gold-standard method in the community of NST.

2) Results of PSPM Fast Style Transfer. The second block shows the results of Per-Style-Per-Model Fast NST algorithms (Section 4.2). Each model only fits one style. It can be noticed that the stylised results of Ulyanov *et al*. [43] and Johnson *et al*. [42] are somewhat similar. This is not surprising since they share a similar idea and only differ in their detailed network architectures. For the results of Li and Wand [49], the results are sightly less impressive. Since [49] is based on Generative Adversarial Network (GAN), to some extent, the training process is not that stable. But we believe that GAN-based style transfer is a very promising direction, and there are already some other GAN-based works [80, 83, 89, 90] (Section 5) in the field of NST.

3) Results of MSPM Fast Style Transfer. The third block demonstrates the results of Multiple-Style-Per-Model Fast NST algorithms. Multiple styles are incorporated into a single model. The idea of both Dumoulin et al.'s algorithm [50] and Chen et al.'s algorithm [51] is to tie a small number of parameters to each style. Also, both of them build their algorithm upon the architecture of [42]. Therefore, it is not reasonable that their results are visually similar. Although the results of [50, 51] are appealing, their model size will become larger with the increase of the number of learned styles. In contrast, Zhang and Dana's algorithm [53] and Li et al.'s algorithm [52] use a single network with the same trainable network weights for multiple styles. The model size issue is tackled, but there seems to be some interferences among different styles (Group II and VII), which slightly influences the stylisation quality.

4) Results of ASPM Fast Style Transfer. The forth block presents the last category of Fast Style Transfer, namely Arbitrary-Style-Per-Model Fast NST algorithms. Their idea is one-model-for-all. Globally, the results of ASPM are slightly less impressive than other types of algorithms. This

[1] http://yongchengjing.com/pdf/review_supp.pdf

Table 1. Average speed comparison of Neural Style Transfer algorithms for images of size 256×256 pixels, 512×512 pixels and 1024×1024 pixels (on an NVIDIA Quadro M6000)

Methods	**Time(s)**			**Styles/Model**
	256 × 256	**512 × 512**	**1024 × 1024**	
Gatys *et al*. [5]	14.32	51.19	200.3	∞
Johnson *et al*. [42]	0.014	0.045	0.166	1
Ulyanov *et al*. [43]	0.022	0.047	0.145	1
Li and Wand [49]	0.015	0.055	0.229	1
Zhang and Dana [53]	0.019 (**0.039**)	0.059 (**0.133**)	0.230 (**0.533**)	$k(k \in Z^+)$
Li *et al*. [52]	0.017	0.064	0.254	$k(k \in Z^+)$
Chen and Schmidt [54]	0.123 (**0.130**)	1.495 (**1.520**)	–	∞
Huang and Belongie [46]	0.026 (**0.037**)	0.095 (**0.137**)	0.382 (**0.552**)	∞
Li *et al*. [57]	0.620	1.139	2.947	∞

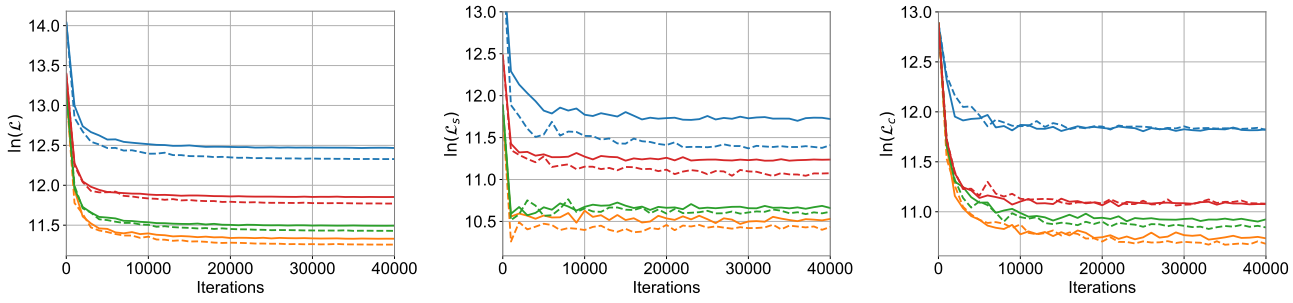
Note: The fifth column shows the number of styles that a single model can produce. Time both excludes (out of parenthesis) and includes (in parenthesis) the style encoding process is shown, since [53], [54] and [46] support storing encoded style statistics in advance to further speed up the stylisation process for the same style but different content images. Time of [54] for producing 1024×1024 images is not shown due to the memory limitation. The speed of [50, 56] are similar to [42] since they share similar architecture. We do not redundantly list them in this table.

is acceptable in that a three-way trade-off between speed, flexibility and quality is common in research. Chen and Schmidt's patch-based algorithm [54] seems to not combine enough style elements into the content image. Their algorithm is based on similar patch swap. When lots of content patches are swapped with style patches that do not contain enough style elements, the target style will not be reflected well. Ghiasi et al.'s algorithm [56] is data-driven and their stylisation quality is very dependent on the varieties of training styles. For the algorithm of Huang and Belongie [46], they propose to match global summary feature statistics and successfully improve the visual quality compared with [54]. However, their algorithm seems not good at handling complex style patterns (Group III and VI), and their stylisation quality is still related to the varieties of training styles. The algorithm of Li *et al*. [57] replaces the training process with a series of transformations. But [57] is not effective at producing sharp details and fine strokes.

6.3. Quantitative Evaluation

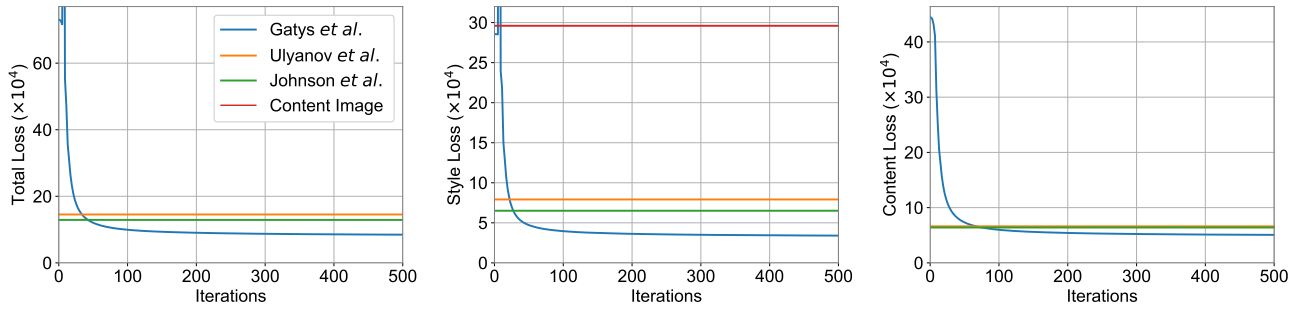
Regarding the quantitative evaluation, we mainly focus on five evaluation metrics, which are: generating time for a single content image of different sizes; training time for a single model; average loss for content images to measure how well the loss function is minimised; loss variation during training to measure how fast the model converges; style scalability to measure how large the learned style set can be.

1) Stylisation speed. The issue of efficiency is the focus of Fast NST algorithms. In this subsection, we compare different algorithms quantitatively in terms of the stylisation speed. Table 1 demonstrates the average time to stylise one image with three resolutions using different algorithms. In our experiment, the style images have the same size as the content images. The fifth column in Table 1 represents the number of styles one model of each algorithm can produce. $k(k \in Z^+)$ denotes that a single model can produce multiple styles, which corresponds to MSPM algorithms. ∞ means a single model works for any style, which corresponds to ASPM algorithms. The numbers reported in Table 1 are obtained by averaging the generating time of 100 images. Note that we do not include the speed of [50, 56] in Table 1 as their algorithm is to scale and shift parameters based on the algorithm of Johnson *et al*. [42]. The time required to stylise one image using [50, 29] is very close to [42] under the same setting. For Chen et al.'s algorithm in [51], since their algorithm is protected by patent and they do not make public the detailed architecture design, here we just attach the speed information provided by the authors for reference: On a Pascal Titan X GPU, 256×256: 0.007s; 512×512: 0.024s; 1024×1024: 0.089s. For Chen and Schmidt's algorithm [54], the time for producing 1024×1024 image is not reported due to the limit of video memory. Swapping patches for two 1024×1024 images needs more than 24 GB video memory and thus, the stylisation process is not practical. We can observe that except for [54, 57], all the other Fast NST algorithms are capable of stylising even high-resolution content images in real-time. ASPM algorithms are generally slower than PSPM and MSPM, which demonstrates the aforementioned three-way trade-off again.



(a) Total Loss Curve (b) Style Loss Curve (c) Content Loss Curve

Figure 6. Training curves of total loss, style loss and content loss of different algorithms. Solid curves represent the loss variation of the algorithm of Ulyanov *et al*. [43], while the dashed curves represent the algorithm of Johnson *et al*. [42]. Different colours correspond to different random selected styles from our style set.



(a) Total Loss (b) Style Loss (c) Content Loss

Figure 7. Average total loss, style loss and content loss of different algorithms. The reported numbers are averaged over our set of style and content images.

2) Training time. Another concern is the training time for one single model. The training time of different algorithms is hard to compare as sometimes the model trained with just a few iterations is capable of producing enough visually appealing results. So we just outline our training time of different algorithms (under the same setting) as a reference for follow-up studies. On a NVIDIA Quadro M6000, the training time for a single model is about 3.5 hours for the algorithm of Johnson *et al*. [42], 3 hours for the algorithm of Ulyanov *et al*. [43], 2 hours for the algorithm of Li and Wand [49], 4 hours for Zhang and Dana [53], and 8 hours for Li *et al*. [52]. Chen and Schmidt's algorithm [54] and Huang and Belongie's algorithm [46] take much longer (*e.g*., a couple of days), which is acceptable since a pre-trained model can work for any style. The training time of [56] depends on how large the training style set is. For MSPM algorithms, the training time can be further reduced through incremental learning over a pre-trained model. For example, the algorithm of Chen *et al*. only needs 8 minutes to incrementally learn a new style, as reported in [51].

3) Loss comparison. One way to evaluate some Fast NST algorithms which share the same loss function is to compare their loss variation during training, *i.e*., the training curve comparison. It helps researchers to justify the choice of architecture design by measuring how fast the model converges and how well the same loss function can be minimised. Here we compare training curves of two popular Fast NST algorithms [42, 43] in Figure 6, since most of the follow-up works are based on their architecture designs. We remove the total variation term and keep the same objective for both two algorithms. Other settings (*e.g*., loss network, chosen layers) are also kept the same. For the style images, we randomly select four styles from our style set and represent them in different colours in Figure 6. It can be observed that the two algorithms are similar in terms of the convergence speed. Also, both algorithms minimise the content loss well during training, and they mainly differ in the speed of learning the style objective. The algorithm in [42] minimises the style loss better.

Another related criterion is to compare the final loss values of different algorithms over a set of test images. This metric demonstrates how well the same loss function can be minimised by using different algorithms. For a fair comparison, the loss function and other settings are also required to be kept the same. We show the results of one Slow

NST algorithm [4] and two Fast NST algorithms [42, 43] in Figure 7. The result is consistent with the aforementioned trade-off between speed and quality. Although Fast NST algorithms are capable of stylising images in real-time, they are not good as Slow NST algorithm in terms of minimising the same loss function.

4) Style scalability. Scalability is a very important criterion for MSPM algorithms. However, it is very hard to measure since the maximum capabilities of a single model is highly related to the set of particular styles. If most styles have somewhat similar patterns, a single model can produce thousands of styles or even more, since these similar styles share somewhat similar distribution of style feature statistics. In contrast, if the style patterns vary a lot among different style images, the capability of a single model will be much smaller. But it is hard to measure how much these styles differ from each other in style patterns. Therefore, to provide the reader a reference, here we just summarise the authors' attempt for style scalability: the number is 32 for [50], 1000 for both [51] and [52], and 100 for [53].

7. Applications

Due to the amazing stylised results, the research of NST has led to many successful industrial applications and begun to deliver commercial benefits. In this section, we summarise these applications and present some potential usages.

7.1. Social Communication

One reason why NST catches eyes in both academia and industry is its popularity in some social networking sites, *e.g.*, Facebook and Twitter. A recently emerged mobile application named *Prisma* [6] is one of the first industrial applications that provide the NST algorithm as a service. Due to its high stylisation quality, *Prisma* achieved great success and is becoming popular around the world. Before long, some other applications providing the same service appeared one after another and began to deliver commercial benefits, *e.g.*, a web application *Ostagram* [7] requires users to pay for a faster stylisation speed. Under the help of these industrial applications [8, 91, 92], people can create their own fantastic art paintings and share their artwork with others on Twitter and Facebook, which is a new form of social communication. There are also some related application papers: [93] introduces an iOS app *Pictory* which combines style transfer techniques with image filtering; [94] further presents the technical implementation details of *Pictory*; [95] demonstrates the design of anther GPU-based mobile app *ProsumerFX*.

The application of NST in social communication reinforces the connections between people and also has positive effects on both academia and industry. For academia, when people share their own masterpiece, their comments can help the researchers to further improve the algorithm. Moreover, the application of NST in social communication also drives the advances of other new techniques. For instance, inspired by the real-time requirements of NST for videos, Facebook AI Research (FAIR) first developed a new mobile-embedded deep learning system *Caffe2Go* and then *Caffe2* (now merged with PyTorch), which can run deep neural networks on mobile phones [96]. For industry, the application brings commercial benefits and promotes the economic development.

7.2. User-assisted Creation Tools

Another use of NST is to make it act as user-assisted creation tools. Although there are no popular applications that applied the NST technique in creation tools, we believe that it will be a promising potential usage in the future.

As a creation tool for painters and designers, NST can make it more convenient for a painter to create an artwork of a particular style, especially when creating computer-made artworks. Moreover, with NST algorithms, it is trivial to produce stylised fashion elements for fashion designers and stylised CAD drawings for architects in a variety of styles, which will be costly when creating them by hand.

7.3. Production Tools for Entertainment Applications

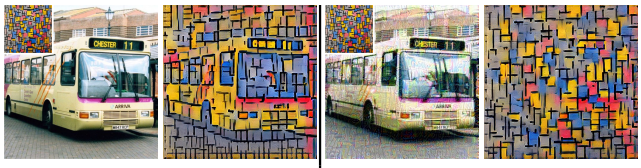
Some entertainment applications such as movies, animations and games are probably the most application forms of NST. For example, creating an animation usually requires 8 to 24 painted frames per second. The production costs will be largely reduced if NST can be applied to automatically stylise a live-action video into an animation style. Similarly, NST can significantly save time and costs when applied to the creation of some movies and computer games.

There are already some application papers aiming at introducing how to apply NST to the production of movies, *e.g.*, Joshi *et al*. explore the use of NST in redrawing some scenes in a movie named *Come Swim* [97], which indicates the promising potential applications of NST in this field.

8. Challenges and Possible Solutions

The advances in the field of NST is amazing and some algorithms have already found use in industrial applications. Although current algorithms achieve remarkable results, there are still several challenges and open issues. In this section, we summarise key challenges within this field of NST and discuss their corresponding possible solutions.

1) Three-way trade-off. The most concerned challenge is probably the three-way trade-off between speed, flexibility and quality in NST. Although current ASPM algorithms



(a) (b) (c) (d)

Figure 8. Adversarial example for Neural Style Transfer: (a) is the original content & style image pair and (b) is the stylised result of (a) with [42]; (c) is the generated adversarial example and (d) is the stylised result of (c) with the same model as (b).

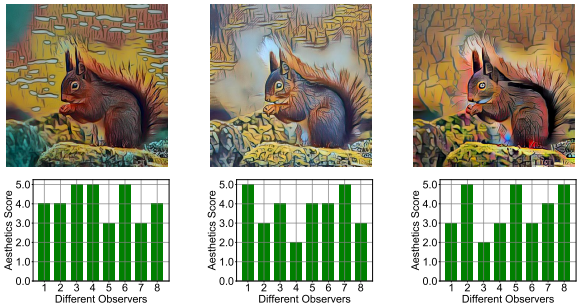


Figure 9. Example of aesthetic preference scores for the outputs of different algorithms given the same style and content.

successfully transfer arbitrary styles, they are not that satisfying in perceptual quality and speed. The quality of data-driven ASPM quite relies on the diversity of training styles. However, one can hardly cover every style due to the great diversity of artworks. Image transformation based ASPM transfer arbitrary styles in a learning-free manner, but it is behind others in speed.

One of the keys for this problem may be a better understanding of the optimisation procedure in NST. The choice of optimiser (*e.g.*, Adam and L-BFGS) in NST greatly influences the visual quality. We believe that a deep understanding towards optimisation procedure will help understand how to find the local minima that leads to a high quality. Also, a well-studied automatic layer chosen strategy would also help improve the quality.

2) Interpretable Neural Style Transfer. Another important issue is the interpretability of NST algorithms. Like many other CNN-based vision tasks, NST is a black box, which makes it quite uncontrollable. Interpreting CNN feature statistics based NST can benefit the separation of different style attributes and address the problem of a finer control during stylisation. For example, current NST algorithms cannot guarantee the detailed orientations and continuities of curves in stylised results. However, brush stroke orientation is an important element in paintings, which can impress the viewer and convey the painter's ideas. Regarding the solution to this problem, fortunately, there are already researches devoted to interpreting CNN [98] which would shed light on the interpretable NST.

3) Adversarial Neural Style Transfer. Several studies have shown that deep classification network is easily fooled by *adversarial examples* [99, 100], which are generated by applying perturbations to input images (*e.g.*, Figure 8(c)). The emergence of adversarial examples reveals the difference between deep neural network and human vision system. The perturbed result by changing an originally correctly classified image is still recognisable to humans, but leads to a misclassified label for deep neural network. Previous studies on adversarial examples mainly focus on deep classification network. However, in Figure 8, we demonstrate that adversarial examples also exist in deep generative network. In Figure 8(d), one can hardly recognise the semantic content, which is originally contained in Figure 8(c). The corresponding countermeasure to this adversarial NST would benefit from previous research on deep classification network. A recent survey on adversarial examples can be found in [101].

4) Evaluation methodology. We believe that the lack of a gold standard aesthetic criterion is a major cause that prevents NST from becoming a mainstream research direction like object detection and recognition. Li *et al*. [57] propose to design a user study to address the aesthetic evaluation problem. It is not practical since the results vary a lot with different observers. We conduct an experiment for user studies and show our results in Figure 9. Given the same stylised result, different observers have quite different ratings. We believe that the problem of a standard aesthetic criterion for NST is a generalised problem of *Photographic Image Aesthetic Assessment*, and one could get inspirations from related researches in this area. Here, we recommend [102] for an overview of *Photographic Image Aesthetic Assessment*.

9. Discussions and Conclusions

Over the past several years, NST has continued to become an inspiring research area, motivated by both scientific challenges and industrial demands. A considerable amount of researches have been conducted in the field of NST. Key advances in this field are summarised in Figure 2. NST is quite a fast-paced area, and we are looking forwarding to more exciting works devoted to advancing the development of this field.

During the period of preparing this review, we are also delighted to find that related researches on NST also bring new inspirations for other areas and accelerate the development of a wider vision community:

1) For the area of *Image Reconstruction*, derived from NST,

Ulyanov *et al*. [103] propose a novel deep image prior, which replaces the manually-designed total variation regulariser in [30] with a randomly initialised deep neural network. Given a task-dependent loss function $\mathcal{L}$, an image I_o and a fixed uniform noise z as inputs, their algorithm can be formulated as:

$$\theta^* = \arg\min_{\theta} \mathcal{L}(g_{\theta^*}(z), I_o),\ I^* = g_{\theta^*}(z). \quad (12)$$

One can easily notice that Equation (12) is very similar to Equation (9). Actually, the process in [103] is equivalent with the training process of Fast NST when there is only one available image in the training set, but replacing I_c with z and $\mathcal{L}_{total}$ with $\mathcal{L}$. In other words, g in [103] is actually trained to overfit one single sample.

2) Inspired by NST, Upchurch *et al*. [104] propose a deep feature interpolation technique and provide a new baseline for the area of *Image Transformation* (*e.g*., face aging and smiling). Upon the procedure of Slow NST algorithm [4], they add an extra step which is interpolating in the VGG feature space. In this way, their algorithm successfully changes image contents in a learning-free manner.

3) Another field closely related to NST is *Face Photo-sketch Synthesis*. For example, [105] exploits style transfer to generate shadings and textures for final face sketches. Similarly, for the area of *Face Swapping*, the idea of Fast NST algorithm [43] can be directly applied to build a feed-forward *Fast Face-Swap* algorithm [106].

4) NST also provides a new way for *Domain Adaption*, as is validated in the work of Atapour-Abarghouei and Breckon [107]. They apply style transfer technique to translate images from different domains so as to improve the generalisation capabilities of their *Monocular Depth Estimation* model.

10. Future Work

Promising directions for future research on NST mainly focus on three aspects. The first one is to solve the existing aforementioned challenges in the field of NST. Descriptions of key challenges and the corresponding possible solutions have been discussed in Section 8. The second aspect is to derive more extensions from general NST, as presented in Section 5. These interesting extensions can bring benefit to both academia and industry, and may even expand into a brand-new field in the future. It is also promising to exploit NST techniques to benefit other vision communities, as introduced in Section 9.

Acknowledgement

We would like to thank Hang Zhang, Dongdong Chen and Tian Qi Chen for providing pre-trained models for our study, and thank Xun Huang and Yijun Li for helpful discussions. We would also like to thank the anonymous reviewers for their insightful comments and suggestions.

This work is supported in part by National Key Research and Development Program (2016YFB1200203), National Natural Science Foundation of China (61572428, U1509206), Fundamental Research Funds for the Central Universities (2017FZA5014), Key Research and Development Program of Zhejiang Province (2018C01004), and Alibaba-Zhejiang University Joint Research Institute of Frontier Technologies.

References

[1] B. Gooch and A. Gooch, *Non-photorealistic rendering*. Natick, MA, USA: A. K. Peters, Ltd., 2001. 1

[2] T. Strothotte and S. Schlechtweg, *Non-photorealistic computer graphics: modeling, rendering, and animation*. Morgan Kaufmann, 2002. 1

[3] P. Rosin and J. Collomosse, *Image and video-based artistic stylisation*. Springer Science & Business Media, 2012, vol. 42. 1, 2

[4] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2414–2423. 1, 5, 7, 9, 10, 11, 13, 16, 18

[5] L. A. Gatys, A. S. Ecker, and M. Bethge, "A neural algorithm of artistic style," *ArXiv e-prints*, Aug. 2015. 1, 2, 5, 12, 13, 14

[6] I. Prisma Labs, "Prisma: Turn memories into art using artificial intelligence," 2016. [Online]. Available: http://prisma-ai.com 2, 16

[7] "Ostagram," 2016. [Online]. Available: http://ostagram.ru 2, 16

[8] A. J. Champandard, "Deep forger: Paint photos in the style of famous artists," 2015. [Online]. Available: http://deepforger.com 2, 16

[9] 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 transactions on visualization and computer graphics*, vol. 19, no. 5, pp. 866–885, 2013. 2

[10] A. Semmo, T. Isenberg, and J. Döllner, "Neural style transfer: A paradigm shift for image-based artistic rendering?" in *Proceedings of the Symposium on Non-Photorealistic Animation and Rendering*. ACM, 2017, pp. 5:1–5:13. 2

[11] A. Hertzmann, "Painterly rendering with curved brush strokes of multiple sizes," in *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*. ACM, 1998, pp. 453–460. 2

[12] 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*. ACM, 2001, pp. 327–340. 2, 11

[13] M. Zhao and S.-C. Zhu, "Portrait painting using active templates," in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Non-Photorealistic Animation and Rendering*. ACM, 2011, pp. 117–124. 2

[14] H. Winnemöller, S. C. Olsen, and B. Gooch, "Real-time video abstraction," in *ACM Transactions On Graphics (TOG)*, vol. 25, no. 3. ACM, 2006, pp. 1221–1226. 2

[15] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Proceedings of the IEEE International Conference on Computer Vision*. IEEE, 1998, pp. 839–846. 2

[16] B. Gooch, E. Reinhard, and A. Gooch, "Human facial illustrations: Creation and psychophysical evaluation," *ACM Transactions on Graphics*, vol. 23, no. 1, pp. 27–44, 2004. 2

[17] A. A. Efros and T. K. Leung, "Texture synthesis by non-parametric sampling," in *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2. IEEE, 1999, pp. 1033–1038. 3, 4

[18] L.-Y. Wei and M. Levoy, "Fast texture synthesis using tree-structured vector quantization," in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 2000, pp. 479–488. 3, 4

[19] 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*. ACM, 2001, pp. 341–346. 3

[20] I. Drori, D. Cohen-Or, and H. Yeshurun, "Example-based style synthesis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2. IEEE, 2003, pp. II–143. 3

[21] O. Frigo, N. Sabater, J. Delon, and P. Hellier, "Split and match: Example-based adaptive patch sampling for unsupervised style transfer," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 553–561. 3

[22] M. Elad and P. Milanfar, "Style transfer via texture synthesis," *IEEE Transactions on Image Processing*, vol. 26, no. 5, pp. 2338–2351, 2017. 3

[23] L.-Y. Wei, S. Lefebvre, V. Kwatra, and G. Turk, "State of the art in example-based texture synthesis," in *Eurographics 2009, State of the Art Report, EG-STAR*. Eurographics Association, 2009, pp. 93–117. 3

[24] B. Julesz, "Visual pattern discrimination," *IRE transactions on Information Theory*, vol. 8, no. 2, pp. 84–92, 1962. 4

[25] D. J. Heeger and J. R. Bergen, "Pyramid-based texture analysis/synthesis," in *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. ACM, 1995, pp. 229–238. 4

[26] J. Portilla and E. P. Simoncelli, "A parametric texture model based on joint statistics of complex wavelet coefficients," *International journal of computer vision*, vol. 40, no. 1, pp. 49–70, 2000. 4

[27] L. A. Gatys, A. S. Ecker, and M. Bethge, "Texture synthesis using convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 262–270. 4, 5

[28] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014. 4, 5

[29] G. Berger and R. Memisevic, "Incorporating long-range consistency in cnn-based texture generation," in *International Conference on Learning Representations*, 2017. 4, 13, 14

[30] A. Mahendran and A. Vedaldi, "Understanding deep image representations by inverting them," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 5188–5196. 4, 18

[31] ——, "Visualizing deep convolutional neural networks using natural pre-images," *International Journal of Computer Vision*, vol. 120, no. 3, pp. 233–255, 2016. 4

[32] A. Dosovitskiy and T. Brox, "Inverting visual representations with convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4829–4837. 4

[33] ——, "Generating images with perceptual similarity metrics based on deep networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 658–666. 4

[34] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680. 4

[35] A. Mordvintsev, C. Olah, and M. Tyka, "Inceptionism: Going deeper into neural networks," 2015. [Online]. Available: https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html 5

[36] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778. 5

[37] Y. Li, N. Wang, J. Liu, and X. Hou, "Demystifying neural style transfer," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, 2017, pp. 2230–2236. [Online]. Available: https://doi.org/10.24963/ijcai.2017/310 5

[38] V. M. Patel, R. Gopalan, R. Li, and R. Chellappa, "Visual domain adaptation: A survey of recent advances," *IEEE signal processing magazine*, vol. 32, no. 3, pp. 53–69, 2015. 6

[39] E. Risser, P. Wilmot, and C. Barnes, "Stable and controllable neural texture synthesis and style transfer using histogram losses," *ArXiv e-prints*, Jan. 2017. 6

[40] S. Li, X. Xu, L. Nie, and T.-S. Chua, "Laplacian-steered neural style transfer," in *Proceedings of the 2017 ACM on Multimedia Conference*. ACM, 2017, pp. 1716–1724. 6

[41] C. Li and M. Wand, "Combining markov random fields and convolutional neural networks for image synthesis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2479–2486. 6, 7, 10, 11

[42] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *European Conference on Computer Vision*, 2016, pp. 694–711. 7, 9, 12, 13, 14, 15, 16, 17

[43] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. Lempitsky, "Texture networks: Feed-forward synthesis of textures and stylized images," in *International Conference on Machine Learning*, 2016, pp. 1349–1357. 7, 12, 13, 14, 15, 16, 18

[44] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *ArXiv e-prints*, Nov. 2015. 7

[45] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6924–6932. 7, 13

[46] X. Huang and S. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1501–1510. 7, 8, 12, 14, 15

[47] Y. Li, N. Wang, J. Shi, J. Liu, and X. Hou, "Revisiting batch normalization for practical domain adaptation," in *International Conference on Learning Representations Workshop*, 2017. 7

[48] Y. Li, N. Wang, J. Shi, X. Hou, and J. Liu, "Adaptive batch normalization for practical domain adaptation," *Pattern Recognition*, vol. 80, pp. 109–117, 2018. 7

[49] C. Li and M. Wand, "Precomputed real-time texture synthesis with markovian generative adversarial networks," in *European Conference on Computer Vision*, 2016, pp. 702–716. 7, 12, 13, 14, 15

[50] V. Dumoulin, J. Shlens, and M. Kudlur, "A learned representation for artistic style," in *International Conference on Learning Representations*, 2017. 7, 8, 12, 13, 14, 16

[51] D. Chen, L. Yuan, J. Liao, N. Yu, and G. Hua, "Stylebank: An explicit representation for neural image style transfer," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1897–1906. 7, 12, 13, 14, 15, 16

[52] Y. Li, F. Chen, J. Yang, Z. Wang, X. Lu, and M.-H. Yang, "Diversified texture synthesis with feed-forward networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3920–3928. 7, 8, 12, 13, 14, 15, 16

[53] H. Zhang and K. Dana, "Multi-style generative network for real-time transfer," *arXiv preprint arXiv:1703.06953*, 2017. 7, 8, 12, 13, 14, 15, 16

[54] T. Q. Chen and M. Schmidt, "Fast patch-based style transfer of arbitrary style," in *Proceedings of the NIPS Workshop on Constructive Machine Learning*, 2016. 8, 12, 13, 14, 15

[55] 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. 8

[56] G. Ghiasi, H. Lee, M. Kudlur, V. Dumoulin, and J. Shlens, "Exploring the structure of a real-time, arbitrary neural artistic stylization network," in *Proceedings of the British Machine Vision Conference*, 2017. 8, 12, 14, 15

[57] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang, "Universal style transfer via feature transforms," in *Advances in Neural Information Processing Systems*, 2017, pp. 385–395. 8, 11, 12, 14, 17

[58] L. A. Gatys, A. S. Ecker, M. Bethge, A. Hertzmann, and E. Shechtman, "Controlling perceptual factors in neural style transfer," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3985–3993. 9

[59] Y. Jing, Y. Liu, Y. Yang, Z. Feng, Y. Yu, D. Tao, and M. Song, "Stroke controllable fast style transfer with adaptive receptive fields," *arXiv preprint arXiv:1802.07101*, 2018. 9

[60] X. Wang, G. Oxholm, D. Zhang, and Y.-F. Wang, "Multimodal transfer: A hierarchical deep convolutional neural network for fast artistic style transfer," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5239–5247. 9

[61] R. Liao, Y. Xia, and X. Zhang, "Depth-preserving style transfer," 2016. 9

[62] X.-C. Liu, M.-M. Cheng, Y.-K. Lai, and P. L. Rosin, "Depth-aware neural style transfer," in *Proceedings of the Symposium on Non-Photorealistic Animation and Rendering*, 2017, pp. 4:1–4:10. 9

[63] D. Zoran, P. Isola, D. Krishnan, and W. T. Freeman, "Learning ordinal relationships for mid-level vision," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 388–396. 9

[64] A. J. Champandard, "Semantic style transfer and turning two-bit doodles into fine artworks," *ArXiv e-prints*, Mar. 2016. 10, 11

[65] Y.-L. Chen and C.-T. Hsu, "Towards deep style transfer: A content-aware perspective," in *Proceedings of the British Machine Vision Conference*, 2016. 10

[66] R. Mechrez, I. Talmi, and L. Zelnik-Manor, "The contextual loss for image transformation with non-aligned data," *arXiv preprint arXiv:1803.02077*, 2018. 10

[67] M. Lu, H. Zhao, A. Yao, F. Xu, Y. Chen, and L. Zhang, "Decoder network over lightweight reconstructed feature for fast semantic style transfer," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2469–2477. 10

[68] C. Castillo, S. De, X. Han, B. Singh, A. K. Yadav, and T. Goldstein, "Son of zorn's lemma: Targeted style transfer using instance-aware semantic segmentation," in *IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2017, pp. 1348–1352. 10

[69] D. Chen, L. Yuan, J. Liao, N. Yu, and G. Hua, "Stereoscopic neural style transfer," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 10

[70] A. Selim, M. Elgharib, and L. Doyle, "Painting style transfer for head portraits using convolutional neural networks," *ACM Transactions on Graphics*, vol. 35, no. 4, p. 129, 2016. 10

[71] M. Ruder, A. Dosovitskiy, and T. Brox, "Artistic style transfer for videos," in *German Conference on Pattern Recognition*, 2016, pp. 26–36. 10

[72] ——, "Artistic style transfer for videos and spherical images," *International Journal of Computer Vision*, 2018. 10

[73] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid, "Deepflow: Large displacement optical flow with deep matching," in *Proceedings of the IEEE International Conference on Computer Vision*. IEEE, 2013, pp. 1385–1392. 10

[74] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid, "Epicflow: Edge-preserving interpolation of correspondences for optical flow," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1164–1172. 10

[75] H. Huang, H. Wang, W. Luo, L. Ma, W. Jiang, X. Zhu, Z. Li, and W. Liu, "Real-time neural style transfer for videos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 783–791. 10

[76] A. Gupta, J. Johnson, A. Alahi, and L. Fei-Fei, "Characterizing and improving stability in neural style transfer," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4067–4076. 10

[77] D. Chen, J. Liao, L. Yuan, N. Yu, and G. Hua, "Coherent online video style transfer," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1105–1114. 10

[78] G. Atarsaikhan, B. K. Iwana, A. Narusawa, K. Yanai, and S. Uchida, "Neural font style transfer," in *Proceedings of the IAPR International Conference on Document Analysis and Recognition*, vol. 5. IEEE, 2017, pp. 51–56. 11

[79] S. Yang, J. Liu, Z. Lian, and Z. Guo, "Awesome typography: Statistics-based text effects transfer," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 7464–7473. 11

[80] S. Azadi, M. Fisher, V. Kim, Z. Wang, E. Shechtman, and T. Darrell, "Multi-content gan for few-shot font style transfer," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 11, 13

[81] F. Luan, S. Paris, E. Shechtman, and K. Bala, "Deep photo style transfer," in *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*. IEEE, 2017, pp. 6997–7005. 11

[82] Y. Li, M.-Y. Liu, X. Li, M.-H. Yang, and J. Kautz, "A closed-form solution to photorealistic image stylization," *arXiv preprint arXiv:1802.06474*, 2018. 11

[83] L. Zhang, Y. Ji, and X. Lin, "Style transfer for anime sketches with enhanced residual u-net and auxiliary classifier gan," in *Proceedings of the Asian Conference on Pattern Recognition*, 2017. 11, 13

[84] J. Liao, Y. Yao, L. Yuan, G. Hua, and S. B. Kang, "Visual attribute transfer through deep image analogy," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, p. 120, 2017. 11

[85] S. Jiang and Y. Fu, "Fashion style generator," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. AAAI Press, 2017, pp. 3721–3727. 11

[86] P. Verma and J. O. Smith, "Neural style transfer for audio spectograms," in *Proceedings of the NIPS Workshop on Machine Learning for Creativity and Design*, 2017. 11

[87] P. K. Mital, "Time domain neural audio style transfer," in *Proceedings of the NIPS Workshop on Machine Learning for Creativity and Design*, 2018. 11

[88] J. Johnson, "neural-style," https://github.com/jcjohnson/neural-style, 2015. 13

[89] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2223–2232. 13

[90] X. Huang, M.-Y. Liu, S. Belongie, and J. Kautz, "Multimodal unsupervised image-to-image translation networks," *arXiv preprint arXiv:1804.04732*, 2018. 13

[91] "DeepArt," 2016. [Online]. Available: https://deepart.io/ 16

[92] R. Sreeraman, "Neuralstyler: Turn your videos/photos/gif into art," 2016. [Online]. Available: http://neuralstyler.com/ 16

[93] A. Semmo, M. Trapp, J. Döllner, and M. Klingbeil, "Pictory: Combining neural style transfer and image filtering," in *ACM SIGGRAPH 2017 Appy Hour*. ACM, 2017, pp. 5:1–5:2. 16

[94] S. Pasewaldt, A. Semmo, M. Klingbeil, and J. Döllner, "Pictory - neural style transfer and editing with coreml," in *SIGGRAPH Asia 2017 Mobile Graphics & Interactive Applications*. ACM, 2017, pp. 12:1–12:2. 16

[95] T. Dürschmid, M. Söchting, A. Semmo, M. Trapp, and J. Döllner, "Prosumerfx: Mobile design of image stylization components," in *SIGGRAPH Asia 2017 Mobile Graphics & Interactive Applications*. ACM, 2017, pp. 1:1–1:8. 16

[96] Y. Jia and P. Vajda, "Delivering real-time ai in the palm of your hand," 2016. [Online]. Available: https://code.facebook.com/posts/196146247499076/delivering-real-time-ai-in-the-palm-of-your-hand 16

[97] B. J. Joshi, K. Stewart, and D. Shapiro, "Bringing impressionism to life with neural style transfer in come swim," *ArXiv e-prints*, Jan. 2017. 16

[98] Q.-s. Zhang and S.-C. Zhu, "Visual interpretability for deep learning: a survey," *Frontiers of Information Technology & Electronic Engineering*, vol. 19, no. 1, pp. 27–39, 2018. 17

[99] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *International Conference on Learning Representations*, 2014. 17

[100] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International Conference on Learning Representations*, 2015. 17

[101] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *arXiv preprint arXiv:1801.00553*, 2018. 17

[102] Y. Deng, C. C. Loy, and X. Tang, "Image aesthetic assessment: An experimental survey," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 80–106, 2017. 17

[103] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Deep image prior," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 18

[104] P. Upchurch, J. Gardner, G. Pleiss, R. Pless, N. Snavely, K. Bala, and K. Weinberger, "Deep feature interpolation for image content changes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 7064–7073. 18

[105] C. Chen, X. Tan, and K.-Y. K. Wong, "Face sketch synthesis with style transfer using pyramid column feature," in *IEEE Winter Conference on Applications of Computer Vision*. Lake Tahoe, USA, 2018. 18

[106] I. Korshunova, W. Shi, J. Dambre, and L. Theis, "Fast face-swap using convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3677–3685. 18

[107] A. Atapour-Abarghouei and T. Breckon, "Real-time monocular depth estimation using synthetic data with domain adaptation via image style transfer," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 18