Computational Visual Media
https://doi.org/10.1007/s41095-022-0287-3 Vol. 9, No. 4, December 2023, 787–806

Research Article

Stroke-GAN Painter: Learning to paint artworks using stroke-style generative adversarial networks

Qian Wang[1,2], Cai Guo[1,3], Hong-Ning Dai[4] (✉), and Ping Li[2,5] (✉)

Abstract It is a challenging task to teach machines to paint like human artists in a stroke-by-stroke fashion. Despite advances in stroke-based image rendering and deep learning-based image rendering, existing painting methods have limitations: they (i) lack flexibility to choose different art-style strokes, (ii) lose content details of images, and (iii) generate few artistic styles for paintings. In this paper, we propose a stroke-style generative adversarial network, called Stroke-GAN, to solve the first two limitations. Stroke-GAN learns styles of strokes from different stroke-style datasets, so can produce diverse stroke styles. We design three players in Stroke-GAN to generate pure-color strokes close to human artists' strokes, thereby improving the quality of painted details. To overcome the third limitation, we have devised a neural network named Stroke-GAN Painter, based on Stroke-GAN; it can generate different artistic styles of paintings. Experiments demonstrate that our artful painter can generate various styles of paintings while well-preserving content details (such as details of human faces and building textures) and retaining high fidelity to the input images.

Keywords AI painting; painting strokes; artistic style

1 School of Computer Science and Engineering, Macau University of Science and Technology, Macau, China. E-mail: anrogim@outlook.com.

2 Department of Computing, The Hong Kong Polytechnic University, Hong Kong, China.

3 School of Computing and Information Engineering, Hanshan Normal University, Chaozhou, China. E-mail: c.guo@hstc.edu.cn.

4 Department of Computer Science, Hong Kong Baptist University, Hong Kong, China. E-mail: hndai@ieee.org (✉).

5 School of Design, The Hong Kong Polytechnic University, Hong Kong, China. E-mail: p.li@polyu.edu.hk (✉).

Manuscript received: 2022-02-14; accepted: 2022-04-12

1 Introduction

Painting, as an important visual art, symbolizes human imagination and ingenuity. Human artists have used a variety of painting tools to create their artworks with specific characteristic styles. However, it is time-consuming for people to master painting skills, requiring much learning, imitating, and practising. Recent computer-aided painting methods generate non-photorealistic images similar to paintings, thereby offering effective painting-assistants for human painting learners, but it is still a challenging task to teach machines to paint artworks based on given images like human artists. Unlike directly generating a style-transfer image or photographic image [1–4], machine painting is carried out by a machine or computer in a stroke-by-stroke manner. The key to teaching a machine to mimic human artists lies in addressing the following three challenges: (i) painting artistic strokes on the canvas in a human-painting order, starting from a given input image, (ii) generating artistic strokes with textures like human artists' strokes, and (iii) preserving detailed contents of a given image while creating a painting instead of reconstructing a photorealistic image.

Some conventional methods include stroke-based rendering (SBR) methods [5–7], which have made contributions to stroke modeling. The quality of such stroke textures is good and mimics human strokes. However, these methods achieve a semi-automatic painting process which needs substantial user intervention. Furthermore, this is time-consuming and requires considerable painting skills of the user, and moreover, these SBR models have a *limited number of painting styles*. Unlike conventional SBR models, learning-based methods have flexible frameworks which can adapt to diverse artistic styles. In addition, they can create paintings without user intervention. Recently, researchers have typically used recurrent neural networks (RNNs) [8, 9] and reinforcement learning (RL) models [10–12] to generate stroke-by-stroke artworks. However, such unified frameworks *lack flexibility to choose different styles of strokes* and some paintings generated in particular artistic styles (e.g., pastel-like paintings) *lack fine details*.

To address limitations of existing painting methods, we have built a new model leveraging advantages of both conventional SBR methods and learning-based methods, as an extension of Ref. [13]. We first describe a novel stroke generative adversarial network (Stroke-GAN) which learns different stroke styles from stroke-style datasets and generates diverse stroke styles with adjustable parameters (stroke path, stroke size and shape, stroke color and transparency). Based on Stroke-GAN, we then describe a neural-network painter which learns to create different styles of paintings in a stroke-by-stroke paradigm. We call the entire framework Stroke-GAN Painter. Stroke-GAN Painter learns to generate a painting in a coarse-to-fine manner, as shown in Fig. 1. In particular, the painting quality becomes better by repeating multiple learning-to-paint processes, during which it proceeds from being a novice to being a veteran. In contrast to existing methods, such as sketching [8, 14], doodling [10], Neural Painter (NP) [15], and MDRL Painter (MDRLP) [12], our painter can generate diverse artistic styles of painting with different types of strokes. Moreover, the images generated by our painter also preserve key content details (such as face details in portraits) well, as shown in Fig. 1. In summary, our work makes the following three main contributions.

- We propose *a three-player-game model*, Stroke-GAN, to generate strokes in an artistic style, which are fully adjustable in terms of stroke path, stroke size and shape, stroke color and transparency, thereby greatly improving the stylization of generated paintings. Stroke-GAN has two generators and one discriminator; the second generator learns to purify the stained-color strokes generated by the first generator. Consequently, the generated strokes have pure

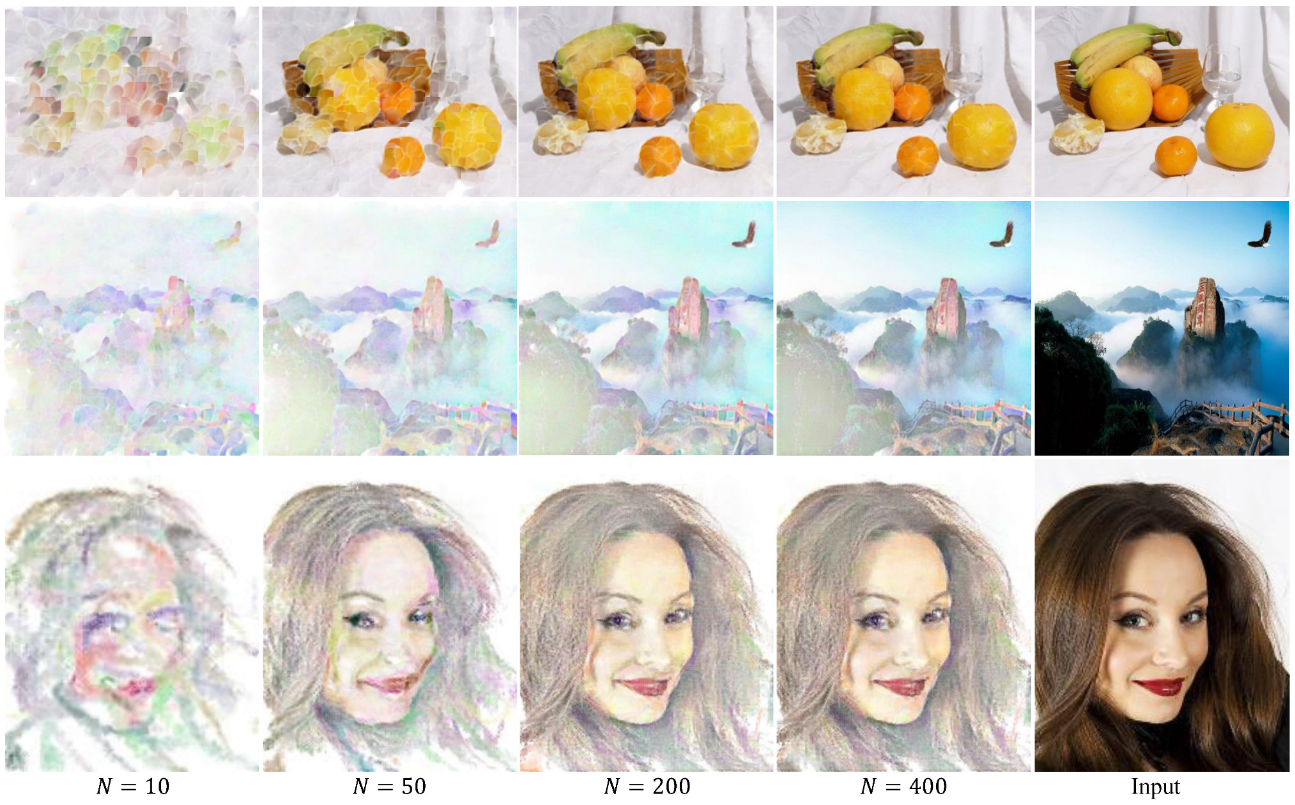

Fig. 1 Learning-to-paint process of Stroke-GAN Painter. Rightmost column: input. Top to bottom: oil painting, watercolor painting, pastel painting. N: number of painting times, rather than number of strokes.

colors and textures closer to human artists' strokes.

- We design a painter based on Stroke-GAN; it does not need to be trained on any image datasets. It learns to create *different artistic styles* of paintings based on stylized strokes, e.g., oil-painting-like artworks, watercolor-like artworks, pastel-like artworks, in a unified framework. Our generated paintings preserve content details of reference images well while offering stylistic diversity of paintings.
- Experimental results show that our painter generates diverse artistic styles of paintings for various types of the image-content, e.g., portraits, landscapes, animals, plants, and buildings. Paintings generated by Stroke-GAN Painter preserve content details well, such as eyes and teeth of portraits and fine details of buildings. User studies with a variety of participants demonstrate that paintings generated by Stroke-GAN Painter are preferred 77% of the time for pastel paintings, in comparison to another method, and 31% of the time for oil paintings, in comparison to three other methods, in terms of fidelity and stylization.

2 Related work

We briefly survey closely related studies on machine painting. We classify related studies into conventional stroke-based rendering (SBR), learning-based rendering, and image-style transfer (IST).

2.1 Conventional stroke-based rendering

SBR methods essentially reconstruct images as non-photorealistic imagery using stroke-based models. Researchers have adopted SBR methods to different types of artworks, e.g., paintings [5–7], pen-and-ink drawings [16, 17], and stippled drawings [18, 19]. In particular, Ref. [5] introduces a semi-automatic painting method based on a greedy algorithm; it needs substantial human intervention to control stroke shapes and select stroke locations. The authors of Ref. [6] propose a style design for their painting method by using spline-brush strokes to *render* the image, but this method requires a high degree of painting skill of its users. The work in Ref. [7] proposes a method to segment an image into areas with similar levels of salience to control the strokes. However, most of these methods require substantial human intervention to choose key parameters, and are thus inconvenient for ordinary users. Moreover, SBR methods only generate a limited number of artistic styles [20], consequently leading to inflexibility.

2.2 Learning-based rendering

Recently, researchers have adopted learning-based methods to improve the painting effects compared to traditional SBR methods. SPIRAL [10, 21] develops an adversarially-trained deep reinforcement learning (DRL) agent which learns structures of images. However, it does not reconstruct details of human portraits well. Sketch-RNN [8] constructs stroke-based drawings after training on human-drawn image datasets; it achieves excellent results for common objects, although it only makes simple-line paintings. StrokeNet [9] trains an agent to learn to paint based on a differentiable render and recurrent neural network (RNN); it generalizes poorly on color images. Computers are now able to generate quite realistic oil paintings [12, 22, 23], and pastel-like paintings [15]. MDRLP [12] paints oil-painting-like pictures with a small number of strokes, although it only mimics one style in a unified framework and loses brush-stroke textures. Methods such as those in Refs. [22, 23] improve stroke textures by redesigning their stroke rendering. NP [15] has a similar design to our method since both generate strokes using a GAN-based module. However, our approach differs from theirs in several ways. Firstly, we use a three-player GAN model to generate adjustable strokes while NP only uses a normal GAN to generate fixed strokes. Secondly, our method can learn from any stroke datasets while NP must use strokes produced by the MyPaint program as the stroke dataset. Thirdly, NP requires a massive manually-labelled stroke dataset to generate one-by-one action strokes. Their model requires the same strokes as the stroke dataset provided by MyPaint to ensure that the optimized strokes are those needed by the painting model. Our model has no data-labelling owing to that it can paint well by using the three-player Stroke-GAN to generate strokes similar to those in the stroke dataset.

2.3 Image style transfer

Image style transfer (IST) methods are popular in both research projects and industrial applications [24–28], although few methods have been applied to stroke-based image rendering. PaintBot [29] based on a DRL network recreates the target image in a stroke-by-stroke manner while the painting style is only restricted to the style of the reference image. Neural Painter [15] uses a method to generate style strokes to recreate the target image without style reference images. However, it requires a large manually-labelled stroke dataset and also lacks flexibility to choose different styles of strokes. Moreover, the generated paintings lack fine details, e.g., in the human face and textures of buildings.

In this paper, we propose a new learning-based method, Stroke-GAN Painter. It integrates the advantages of SBR methods with learning-based methods to generate diverse styles of paintings with high quality. Specifically, Stroke-GAN can generate different styles of strokes, and stroke designer modules can generate different style stroke datasets for Stroke-GAN. Meanwhile, we use a rendering module to optimize stroke selection, to ensure Stroke-GAN generates well-behaved strokes, which are rendered onto the canvas to create high-quality paintings.

3 Stroke-GAN Painter

3.1 Overview

We propose a new painting model, Stroke-GAN Painter, to achieve stroke-by-stroke painting for machines or computers. The goal of Stroke-GAN Painter is to paint in *diverse artistic styles* of paintings in a unified framework. We mainly consider oil paintings, watercolor paintings, and pastel paintings here, although further painting styles could also be easily added to our framework. When given an image, our model can continually render style-strokes onto the canvas to create different artistic styles of paintings by choosing the style of the strokes. Figure 2 depicts our proposed Stroke-GAN Painter which consists of a stroke designer module, Stroke-GAN, the rendering module, and the canvas. The stroke designer module provides different styles of stroke datasets for training Stroke-GAN, which generates style strokes (see Fig. 2(a)). The rendering module feeds in both the reference image U_0 and canvas C_n to optimize stroke selection, which controls Stroke-GAN to generate well-behaved strokes. The states of the canvas during the learning-to-paint process are shown in Fig. 2(b). Stroke-GAN Painter learns

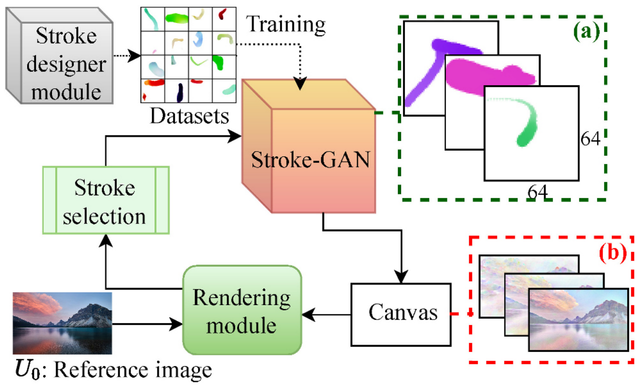

Fig. 2 The network architecture of Stroke-GAN Painter mainly comprises Stroke-GAN, the rendering module, and the canvas. (a) shows style strokes generated by Stroke-GAN. (b) shows states of the canvas during the learning-to-paint process.

to create paintings from novice to veteran ability and its painting quality improves with more painting time. Section 3.2 presents the stroke designer module to generate different styles of stroke datasets for training Stroke-GAN. Section 3.3 presents Stroke-GAN, which feeds in values obtained from stroke selection to generate style strokes, which are then painted on the canvas C_n. The rendering module feeds in both U_0 and C_n to optimize stroke selection, thereby finishing the painting process. Section 3.4 describes the painting process and stroke selection optimization.

3.2 Stroke designer module

3.2.1 Stroke modeling

The artistic style of a painting can be affected by the stroke style. The stroke designer module is used to create different styles of strokes to provide training datasets for Stroke-GAN. Inspired by previous studies [12, 15], our stroke designer modules consider the following variables: $\mathbb{P}$ denotes a set of control points of a Bézier curve, $\mathbb{S}$ denotes a set controlling size of a geometric shape, $\mathbb{T}$ denotes a set to control stroke transparency, and $\mathbb{V}$ denotes a set to control stroke color.

- Stroke path: We use different geometric shapes to represent the brush tip and Bézier curves to represent the path of a brush. The points in $\mathbb{P} = \{(x_i, y_i)|i = 0, \ldots, d\}$ control a Bézier curve, where d is the order of the Bézier curve. See Eq. (1) later.
- Stroke size and shape: We use $\mathbb{S} = \{s_0, s_1\}$ to define the size of a geometric shape to control the

stroke size and shape. The size of the brush tip varies with the values of $\mathbb{S}$.

- Stroke transparency: We use $\mathbb{T} = \{t_0, t_1\}$ to control the transparency of the stroke. The transparency of the stroke varies with the values of $\mathbb{T}$.
- Color: Primary colors denoted by $\mathbb{V} = \{r, g, b\}$ determine the color of the stroke.

3.2.2 *Stroke datasets*

The stroke designer module provides different stroke-style datasets for training Stroke-GAN. Each stroke dataset includes 200,000 stroke images, each with 64 × 64 resolution. We currently provide stroke datasets in three diverse styles, although others could be added: a watercolor-stroke dataset, an oil-painting-stroke dataset, and a pastel-stroke dataset. The pastel-stroke dataset is from Ref. [15] while both oil-painting-stroke dataset and watercolor-stroke dataset are provided by our stroke designer modules. We model the stroke as its path and tip, using a Bézier curve (BC) to construct the stroke path. Circles with variable radii are used to simulate the stroke tip. Each stroke is made from 100 variable circles moving along BC, which is given by

$$\boldsymbol{B}(t) = \sum_{i=0}^{d} \binom{d}{i} (1-t)^{(d-i)} t^i \boldsymbol{P}_i, \quad t \in [0,1] \qquad (1)$$

where $\boldsymbol{P}_i$ denotes the control point with coordinates $(x_i, y_i) \in \mathbb{P}$.

3.3 Stroke-GAN

3.3.1 *Basics*

In order to improve the painting quality and the fidelity of the stroke, we have designed a *three-player GAN model*, Stroke-GAN, to generate stylized strokes for the painting process. Stroke-GAN is the core component which allows our model to use a unified framework to produce diverse artistic styles of paintings from a given image. Its third player, the coloring module, allows preservation of high fidelity details. Paintings contain several elements: lines, textures, colors, and so on [30]. Our Stroke-GAN is designed to generate strokes containing these elements; the stroke style has a major influence on the painting style. As Stroke-GAN has an end-to-end training model, our painter framework has the flexibility to change the artistic style by choosing differently-trained Stroke-GAN models. We do not need to train the whole painter on any image datasets since the model is designed to learn to paint from novice to veteran ability.

3.3.2 *Motivation for Stroke-GAN*

Our design for Stroke-GAN follows the Deep Convolutional Generative Adversarial Network (DCGAN) [31], which offers more stable training than conventional GANs. However, the strokes generated by a normal DCGAN (i.e., a two-player game) have stained colors. It is unreasonable to try to reproduce the painting process of human artists by rendering strokes in stained colors. To address this problem, we design a second generator (the third player) to re-color strokes with three color parameters to generate pure-colored strokes, which are better for painting (closer to the strokes painted by human artists).

3.3.3 *Three-player-game of Stroke-GAN*

In order to obtain pure-colored strokes for reasonable painting strokes, we provide a coloring module as a second generator G' immediately following the normal generator G (being essentially a convolutional neural network), as shown in Fig. 3. Stroke-GAN consists of a normal generator G, a coloring module G', and a discriminator D. Let $\boldsymbol{h}_t$ denote the one-dimensional (1-D) vector used as the random noise to generated strokes. Let $\boldsymbol{h}_t^s$ and $\boldsymbol{h}_t^c$ denote 1-D vectors being fed into G and G', respectively, and $\boldsymbol{h}_t = [\boldsymbol{h}_t^s, \boldsymbol{h}_t^c]$. After being fed with $\boldsymbol{h}_t^s$ and $\boldsymbol{h}_t^c$, respectively, G learns to generate stroke images but stained-colored, and G' learns to generate pure-color strokes. The discriminator first determines whether a stroke generated by G is valid or not, and then determines the stroke image generated by G'. Both G and G' update during the adversary mode with D. Since the randomness of DCGAN leads to the unexpected stroke color, we build a second generator G' (the coloring module) to control the stroke color. Particularly, the vector $\boldsymbol{h}_t$ consists of 50 elements.

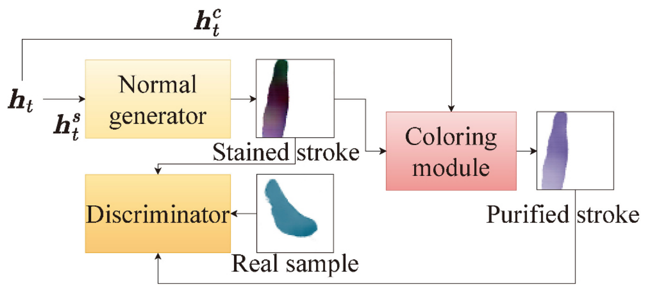

Fig. 3 Structure of Stroke-GAN. The stroke designer module provides stroke datasets. Stroke-GAN consists of a normal generator G, a coloring module G', and a discriminator D.

The normal generator (G) feeds the variable $\boldsymbol{h}_t^s$ with 47 elements and outputs the stroke image $G(\boldsymbol{h}_t^s)$ while $\boldsymbol{h}_t^c$ is used to control the stroke color $\{r, g, b\}$ fed into G'. Stroke-GAN improves the original DCGAN to purify the colors of strokes (see Fig. 3). The coloring module feeds $\boldsymbol{h}_t^c$ with the stroke $G(\boldsymbol{h}_t^s)$ to learn to purify the stroke.

Since the stroke image just contains the background and the stroke, we can use *threshold segmentation* [32] to separate the stroke region from the background. Let $\boldsymbol{P} = [p_1, \ldots, p_c]$ denote the matrix of pixels in the stroke image, where c is the number of channels of the image. The average of $\boldsymbol{P}$ is denoted by $\overline{\boldsymbol{P}}$. Since the pixel values of the background in the stroke image generated by G are unknown, we should train G' to find the threshold of the background pixels. We denote the threshold by γ. We use threshold segmentation to eliminate the background region from the stroke image, and the result is denoted by $\widetilde{\boldsymbol{P}}$, which is obtained as Eq. (2):

$$\widetilde{\boldsymbol{P}} = \gamma - \overline{\boldsymbol{P}} \tag{2}$$

The values of the elements in $\widetilde{\boldsymbol{P}}$ are 0 or close to 0 in the background region. We use $\min(\cdot)$ and $\max(\cdot)$ to calculate the minimum and maximum values in $\widetilde{\boldsymbol{P}}$, respectively. We determine the stroke region and denote the results by $\boldsymbol{S}_P$, which is obtained by

$$\boldsymbol{S}_P = (\widetilde{\boldsymbol{P}} - \min(\widetilde{\boldsymbol{P}}))/(\max(\widetilde{\boldsymbol{P}}) - \min(\widetilde{\boldsymbol{P}})) \tag{3}$$

In particular, the values of the elements in $\boldsymbol{S}_P$ close to 1 are stroke pixels, and values close to 0 represent background pixels. We then use $\boldsymbol{h}_t^c$ to recolor the stroke and obtain the pure stroke image $\boldsymbol{P}_S$ as Eq. (4):

$$\boldsymbol{P}_S = [\boldsymbol{S}_P, \boldsymbol{S}_P, \boldsymbol{S}_P] \cdot \boldsymbol{h}_t^c \tag{4}$$

The coloring module endows our painter with more creativity in painting, e.g., outputting paintings with different colors even from the same input image. The reason is that the coloring module in Stroke-GAN takes in $\boldsymbol{h}_t^c$ directly to recolor the stroke image by learning the color information of the input. This is the reason that Stroke-GAN can be trained without the data-labelling restriction of generating an image the same as the labeled image. Even though Stroke-GAN generates strokes different from the dataset, the coloring module learns color information by analyzing the input image directly so as to ensure that the rendered canvas is close to the input image. The design of the coloring module plays an important role in ensuring the GAN generates realistic human strokes. Moreover, this design also allows Stroke-GAN to easily learn different styles of strokes. Therefore, while Stroke-GAN utilizes a unified framework, it can generate different styles of paintings.

3.3.4 Training of Stroke-GAN

We train Stroke-GAN to acquire different stroke models to endow our painter the ability to paint with different stroke styles. Figure 3 depicts the structure of Stroke-GAN. Stroke-GAN is trained with 128 images as a mini-batch for each stroke dataset containing 200,000 images. We use the whole dataset as the training set since the DCGAN model has no need for validation. When training Stroke-GAN, we directly feed the generator with a 1-D vector ($\boldsymbol{h}_t$) to generate a stroke image. The initial values of the elements in $\boldsymbol{h}_t$ are random. During training, the generator learns to produce images close to the dataset by optimizing the values of these elements. We use the Adam optimizer to train the Stroke-GAN model, with a learning rate of 0.0002, and values of betas of 0.5 and 0.999. Each pair comprising a generated stroke image and a real stroke image is then fed into the discriminator. The discriminator next determines whether the pair of strokes is valid or not. If the generated stroke image is similar to the real stroke image, the pair is valid, and invalid otherwise.

The training procedure for Stroke-GAN is given in Algorithm 1. We essentially train the normal generator G, the coloring module G', and the discriminator D by back-propagating the loss, to update the parameters θ_{g}, θ_{c}, and θ_{d}, respectively. In particular, the discriminator D has the loss ℓ_{d}, the generator G has the loss ℓ_{g}, and the generator G' has the loss ℓ_{c}. We use binary cross entropy (BCE) denoted by $\ell(\boldsymbol{z}, \boldsymbol{y})$ to measure the loss of the input sample $\boldsymbol{z}$ on conditional variable $\boldsymbol{y}$, with the number of the samples of T, as Eq. (5):

$$\ell(\boldsymbol{z}, \boldsymbol{y}) = \frac{1}{T} \sum_{t=1}^{T} \left[-\boldsymbol{y}_t \log(\boldsymbol{z}_t) - (1 - \boldsymbol{y}_t) \log(1 - \boldsymbol{z}_t) \right] \tag{5}$$

When training the discriminator on real stroke images (denoted by $\boldsymbol{x}$), $\boldsymbol{y} = 1$, so using Eq. (5), the loss ℓ_{dr} for real strokes is

$$\ell_{\mathrm{dr}} = \ell(D(\boldsymbol{x}), 1) \tag{6}$$

When training the discriminator on fake stroke images generated by G, we then have $\boldsymbol{y} = 0$, so the loss for

Algorithm 1 Training Stroke-GAN. Discriminator D, normal generator G, second generator G'. The number of iterations is m, and the mini-batch is k

```
1: for $m$ do
2:   for $k$ do
3:     Sample real data $\{x_1, \dots, x_k\}$ from stroke dataset;
4:     Set $label = 1$;
5:     Train $D$ with the real-loss function using
         $\mathrm{BCELoss}(D(x), label)$;
6:     Sample fake data $\{g_1, \dots, g_k\}$ generated by using
       random noise $\boldsymbol{h}_t = [\boldsymbol{h}_t^s, \boldsymbol{h}_t^c]$;
7:     Set $label = 0$;
8:     Train $D$ with the real loss added to the fake-loss
       function using
         $\mathrm{BCELoss}\big(D(x), 1\big) + \mathrm{BCELoss}\Big(D\big(G(\boldsymbol{h}_t^s)\big), label\Big)$;
9:     Set $label = 1$;
10:    Sample fake data $\{g_1, \dots, g_k\}$ generated by using
       random noise $\boldsymbol{h}_t^s$;
11:    Train $G$ with the loss function
         $\mathrm{BCELoss}\Big(D\big(G(\boldsymbol{h}_t^s)\big), label\Big)$;
12:    Sample fake data $\{g_1, \dots, g_k\}$ generated by using
       random noise $\boldsymbol{h}_t^s$;
13:    Input $\boldsymbol{h}_t^c$;
14:    Train $G'$ with the loss function:
         $\mathrm{BCELoss}\Big(D\big(G'(G(\boldsymbol{h}_t^s), \boldsymbol{h}_t^c)\big), label\Big)$;
15:  end for
16: end for
```

fake strokes, ℓ_{da}, is

$$\ell_{\mathrm{da}} = \ell\big(D(G(\boldsymbol{h}_t^s)), 0\big) \tag{7}$$

When training the discriminator on fake stroke images generated by G', we again have $\boldsymbol{y} = 0$, so the loss for fake strokes ℓ_{db}, is

$$\ell_{\mathrm{db}} = \ell\Big(D\big(G'(G(\boldsymbol{h}_t^s), \boldsymbol{h}_t^c)\big), 0\Big) \tag{8}$$

The entire loss of the discriminator is

$$\ell_{\mathrm{d}} = \ell_{\mathrm{dr}} + \ell_{\mathrm{da}} + \ell_{\mathrm{db}} \tag{9}$$

so

$$\ell_{\mathrm{d}} = \frac{1}{T}\sum_{t=1}^{T}\Big[-\log\Big(D(\boldsymbol{x}_t)\Big) - \log\Big(1 - D\big(G(\boldsymbol{h}_t^s)\big)\Big) - \log\Big(1 - D\big(G'(G(\boldsymbol{h}_t^s), \boldsymbol{h}_t^c)\big)\Big)\Big] \tag{10}$$

Similarly, the normal generator G has the loss:

$$\ell_{\mathrm{g}} = \frac{1}{T}\sum_{t=1}^{T} -\log\Big(D\big(G(\boldsymbol{h}_t^s)\big)\Big) \tag{11}$$

and the loss function of the coloring module is

$$\ell_{\mathrm{c}} = \frac{1}{T}\sum_{t=1}^{T} -\log\Big(D\big(G'(G(\boldsymbol{h}_t^s), \boldsymbol{h}_t^c)\big)\Big) \tag{12}$$

where the strokes in $G(\boldsymbol{h}_t^s)$ are visually stained. The coloring module learns to compute the content of the stroke region in $G(\boldsymbol{h}_t^s)$ and recolor the stroke by $\boldsymbol{h}_t^c$.

Figure 4(a) compares sample strokes generated by Stroke-GAN with and without the coloring module (CM) for various stroke datasets. Figure 4(b) shows convergence of Stroke-GAN for different stroke datasets. The stained-color strokes generated by Stroke-GAN W/O CM cannot mimic human artists' strokes. The pure-color strokes generated by Stroke-GAN W CM are close to human artists' strokes, benefiting the quality of the content. Although both Stroke-GAN W/O CM and Stroke-GAN W CM can generate strokes similar to the given strokes, the former generates parti-colored strokes while the latter generates pure-colored strokes, which are better as they are closer to those of human artists. Stroke-GAN can learn any styles of strokes given an appropriate training dataset. In this paper, we use three stoke datasets: watercolor, oil-painting, and pastel stroke datasets, and save the trained models as Style 1, Style 2, and Style 3, respectively. We then choose the corresponding model to give a certain artistic style.

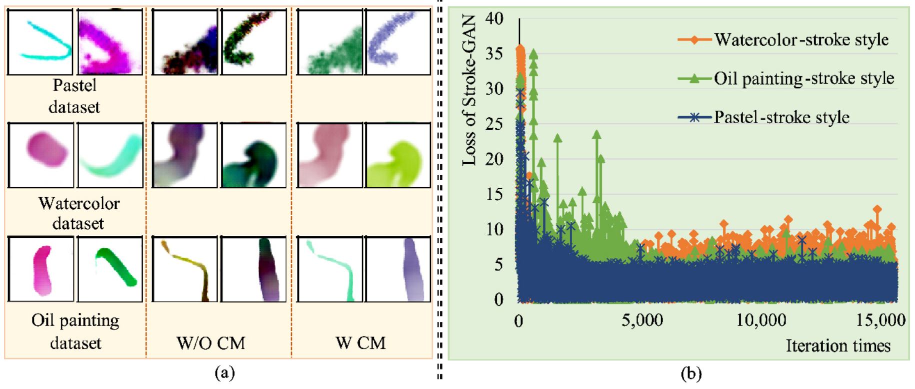

Fig. 4 Training Stroke-GAN. (a) Stroke samples generated by two comparative methods: Stroke-GAN with coloring module (W CM) and without coloring module (W/O CM). (b) Loss of Stroke-GAN versus iterations for three different stroke datasets.

In summary, Stroke-GAN has the following merits. Firstly, it can recolor the stroke according to the design of the coloring module, thereby improving the artistic creativity of the painter. For example, the color of the painting can be recreated close to but not the same as the input reference image. Secondly, it can flexibly learn any style of strokes as long as a stroke dataset is available, owing to its completeness and independence. Thirdly, it enables end-to-end training so that it can be easily applied in a painting models for various artistic styles, by choosing different Stroke-GAN models.

3.4 Rendering module

3.4.1 Basics

We endow our painter with the capability to paint in diverse artistic styles. Besides using diverse stroke-styles generated by Stroke-GAN, we also consider the feature-extraction network (FEN) to extract contents of reference images. After processing by the FEN, the original reference image may lose some content but retaining the core information of the image. We design the rendering module with an FEN and an optimization algorithm. We use the FEN in our rendering module to process the features of the reference image and the canvas. We use the optimization algorithm to "pick" well-behaved strokes for rendering the canvas.

3.4.2 Painting process

We design our painter to mimic the painting process used by human artists painting in a given style. Painting is conducted in a coarse-to-fine manner, in which our painter learns to paint from scratch to a finely-detailed painting after multiple times of painting. The Stroke-GAN generates a sequence of strokes at one time and the rendering module optimizes stroke selection (see Section 3.4.3) to "render" these strokes on the canvas. The painting process is shown in Fig. 5. The painting model consists of the rendering module, stroke selection, the canvas, and the Stroke-GAN. The rendering module optimizes the stroke selection and the Stroke-GAN generates continually strokes used to paint. Stroke-GAN Painter learns to paint from novice to veteran ability. We observe that the painting quality improves by repeating multiple learning-to-paint processes. The painting quality becomes better with increased n. The reference image is denoted by

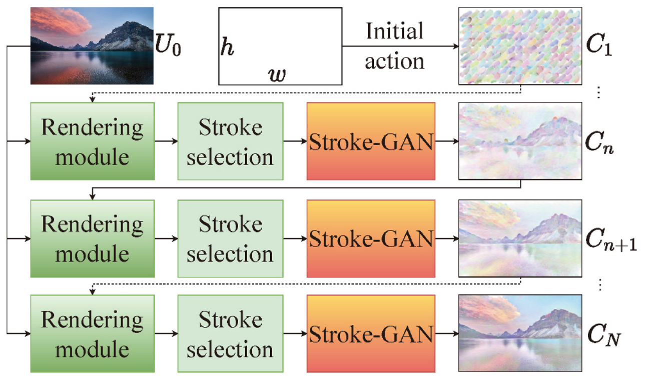

Fig. 5 Coarse-to-fine learning-to-paint process. The reference image is U_0, while C_n denotes the current state of the canvas, and N is the number of painting processes. The width and the height of the canvas are w and h, respectively.

U_0. The content state of a certain canvas and the set of stroke selections are denoted by C_n and H_n, respectively. The height h and the width w of the canvas are automatically configured according to the aspect ratio of the reference image.

We model our painting process as a stroke state optimization process with a canvas state set $\mathcal{S}$, a stroke selection $\mathcal{H}$, and a mapping f: $\mathcal{S} \to \mathcal{H}$. Let $\mathcal{S} = \{C_n | n = 1, \ldots, N\}$, $\mathcal{H} = \{H_n | n = 1, \ldots, N\}$, where N is the number of iterations of the painting model (also the number of times of painting). Let the total number of strokes needed to complete the painting be T. Since each H_n has T elements, we then choose $H_n = \{\boldsymbol{h}_t | t = 1, \ldots, T\}$, where $\boldsymbol{h}_t$ is also the input of Stroke-GAN when training the Stroke-GAN (mentioned in Section 3.3.3). One $\boldsymbol{h}_t$ in the stroke selection is used to *pick* one stroke (generated by Stroke-GAN). For each painting iteration, the stroke selection outputs a set of H_n of size T to let Stroke-GAN generate T strokes. One painting process is finished when T strokes have all been rendered onto the canvas, i.e., C_n is completed for some n.

Each stroke generated by the Stroke-GAN is essentially an image, with 64×64 pixels. Thus, the canvas is divided into grid cells for rendering convenience, with the size of each cell also 64×64 pixels. We render T strokes onto the canvas cell-by-cell to finish one painting process. In each cell, the content at the stroke position in the new stroke image replaces that at the corresponding position. Stroke-GAN produces T strokes at a time, where T equals the number of strokes in each cell multiplied by $h \times w$ cells. The strokes are sequentially rendered in a grid order. Stroke-GAN runs once in one painting process. The mapping $f\colon \mathcal{S} \to \mathcal{H}$ uses the transition function $C_{n+1} = f(C_n, H_{n+1})$. Stroke selection first outputs an initial set H_1; each element in H_1 denotes an effect for rendering a stroke at a certain position of the canvas. The rendering module then optimizes stroke selection via the stroke-selection-optimization algorithm and generates a new set of elements H_n, which are used to render strokes on the canvas to get C_n. We continue the coarse-to-fine process from C_n to C_{n+1}, where C_{n+1} denotes a finer-grained painting (with optimized strokes). Continuing the above process, we finally obtain the best painting C_N.

3.4.3 Stroke selection optimization

A rendering module is used to optimize stroke selection; it consists of an FEN and the optimization algorithm. During the painting process, the rendering module first feeds in both the reference image and the painted canvas to compute the distance between them, and then optimizes the stroke selection. Stroke selection *picks* well-behaved strokes generated by the Stroke-GAN. This ensures that the state of the canvas C_{n+1} is better than C_n. This procedure continues until one painting process is complete, and the painting C_N is generated after N painting process. The learning-to-paint process works in a coarse-to-fine manner.

It is a key task to optimize the stroke generated by the Stroke-GAN in the stroke selection step. The rendering module first extracts the feature maps of U_0 and C_n, and then processes the difference of the input and the canvas by computing their ℓ_1-distance. We denote the extracted feature maps of the input reference image U_0 and those of the painted canvas C_n by $\mathcal{F}(U_0) = \{I_j | j = 1, \ldots, M\}$ and $\mathcal{F}(C_n) = \{c_j | j = 0, \ldots, M\}$, respectively, where M is the number of features extracted by the neural rendering module. In particular, I_j and c_j denote the features of U_0 and those of a certain state of the canvas C_n, respectively. We calculate the ℓ_1-distance loss function $\mathcal{L}(U_0, C_n)$ as

$$\mathcal{L}(U_0, C_n) = \frac{1}{M} \sum_{j=1}^{M} |I_j - c_j| \tag{13}$$

This essentially computes the distance between the features of the reference image U_0 and those of canvas C_n. The stroke selection algorithm optimizes the generated stroke by resetting the values of the elements in $\boldsymbol{h}_t$ based on the ℓ_1-distance loss function. The function $f(C_n, H_{n+1})$ is computed by the backpropagation algorithm for $\mathcal{L}(U_0, C_n)$. Each C_n (the state of the canvas) is rendered by T strokes, and each stroke is produced according to the values in $\boldsymbol{h}_t$. Therefore, the values of the elements in each $\boldsymbol{h}_t$ can be updated to the ones needed by backpropagation for $\mathcal{L}(U_0, C_n)$.

3.5 Style reconstruction

3.5.1 Basics

As explained in Section 3.3, Stroke-GAN is the core component for generating the style. The rendering module also contributes to the stylization of the whole painting. In particular, we use the FEN in the rendering module to extract features of the reference image as well as the painted canvas. This process loses some content of the original image but mimics paintings close to the original image. This step can make the generated painting differ from but be similar to the reference image, providing artistic mimesis or "realism". We reconstruct the painting style by using the Stroke-GAN and FEN in the rendering module. In particular, Stroke-GAN endows the painter with diverse styles of strokes and the FEN in the rendering module creates the artistic style.

3.5.2 Artistic style

Since Ref. [15] indicates that the content objective preserves only high-level features while the parameterization can fill in details, we also only take high-level features as inputs. We adopt two of the most representative deep neural networks: GoogleNet [33] or residual nets (ResNets) [34] for the digital rendering module. The design of using FEN to process the original reference image, rather than directly using the original image, endows our painter with artistic creativity while retaining a high similarity to the reference image. We focus on realism in oil paintings, so we use ResNet to build the FEN in the rendering module. ResNets have high accuracy when extracting information [34], so can keep a high fidelity in the extracted features. On the other hand, features extracted by GoogleNet are relatively sparse so that may offer more space for our painterly creativity. Therefore, we use GoogleNet for watercolor and pastel paintings.

3.5.3 Stroke style

Different strokes can produce different styles of artwork even when used by the same human artist. We provide three kinds of strokes to endow our painter with more creativity. Figure 6 shows different stroke styles for watercolors, oil-painting, and pastels. The watercolor strokes have smooth, soft contours and the brush paths are simple and pure. In contrast, oil-painting strokes have sharp contours and volatile paths. When stacking multiple strokes on the canvas, the oil-painting texture can be recognised easily due to these characteristics. The pastel strokes seem to be accumulated from many uneven points (mimicking the granular textures of pastel paintings). These different styles of strokes cause the canvas to show different styles of painting. After utilizing different FENs to process the reference image in conjunction with different types of strokes, we obtain different styles of paintings.

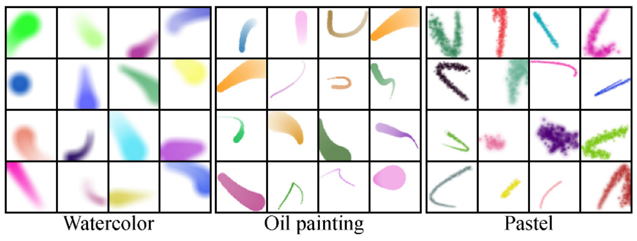

Fig. 6 Samples from three datasets used to mimic the styles of watercolor, oil-painting, and pastel strokes

4 Experimental results

We have evaluated our approach with several experiments. We first describe the implementation. Then, we evaluate three styles of paintings generated by our painter and compare the output of the proposed Stroke-GAN Painter to state-of-the-art methods. Finally, we study alternatives to understand how our Stroke-GAN Painter generates different styles of paintings by fine-tuning the design.

4.1 Implementation

Our experiments were conducted on a workstation with an i7-7700k CPU and an NVIDIA Titan RTX GPU. We evaluated our painter on three image datasets: CelebA [35], ImageNet [36], and real-world photos. These images cover various types of content including portraits, landscapes, animals, plants, and buildings. All images used in experiments are labelled by “Img No.”. Style 1 (Stroke-GAN model 1) and the rendering module using GoogleNet were used to generate watercolor paintings, Style 2 and ResNet were used to generate oil paintings, and Style 3 and GoogleNet were used to generate pastel paintings.

4.2 Comparison of stroke styles

We compare paintings generated by different styles of strokes on CelebA, ImageNet, and real-world photos. In Fig. 7 and Fig. 8, the top row shows the inputs, and the images in successive rows show oil painting, watercolor painting, and pastel painting results. Img 39 and Img 12 were randomly selected from CelebA [35], Img 32 and Img 35 were randomly selected from ImageNet [36], and the others are real-world photos. Figures 7 and 8 show that all generated paintings exhibit different styles in contrast to the reference images. In particular, the oil-painting-stroke paintings in the second row well preserve textures, lines, and color features, consequently capturing fine details in the reference images. Meanwhile, we observe stroke textures from oil paintings, demonstrating oil-painting stylization. The watercolor-stroke paintings in the third row exhibit a style between pastel paintings and oil paintings; this style is good at expressing details for scenery and building images (see Imgs 26, 15, 11, 21, and 23). The pastel-stroke paintings in the last row preserve some textures and lines while losing some color features.

Figure 9 plots ℓ_1-distances between the generated images and the reference images. Specifically, Fig. 9(a) plots the ℓ_1-distance versus painting times for the three stroke styles. We observe that all three styles nearly converge after 300 painting times, although the oil-painting stroke style converges faster than the other two. The pastel-stroke style paintings converge the slowest since they lose more content detail. Figure 9(b) compares convergence for three types of image datasets using the same watercolor-stroke style. It takes 200 painting iterations to recreate the images in CelebA, 300 iterations for images from ImageNet, and 400 for real-world photos. The portrait images of CelebA are relatively easier to learn than those of ImageNet and real-world photos as they have fewer features.

4.3 Comparison to prior methods

4.3.1 Methods

We further evaluate our painter by comparing it to various state-of-the-art learning-based methods, including Neural Painter (NP) [15], MDRL Painter (MDLRP) [12], SNP [22], and PaintTF [23]; these outperform other learning methods and traditional

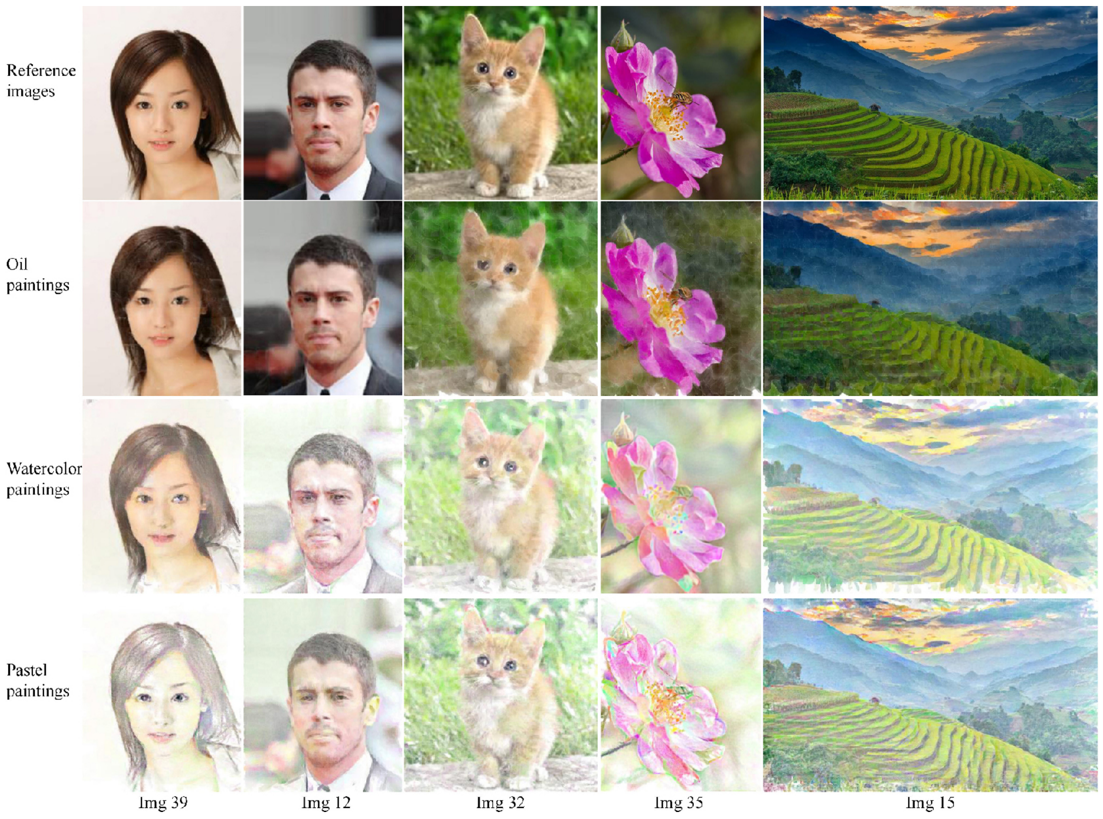

Fig. 7 Three artistic styles of paintings generated by Stroke-GAN Painter using images from CelebA [35], ImageNet [36], and a real-world photo (Img 15).

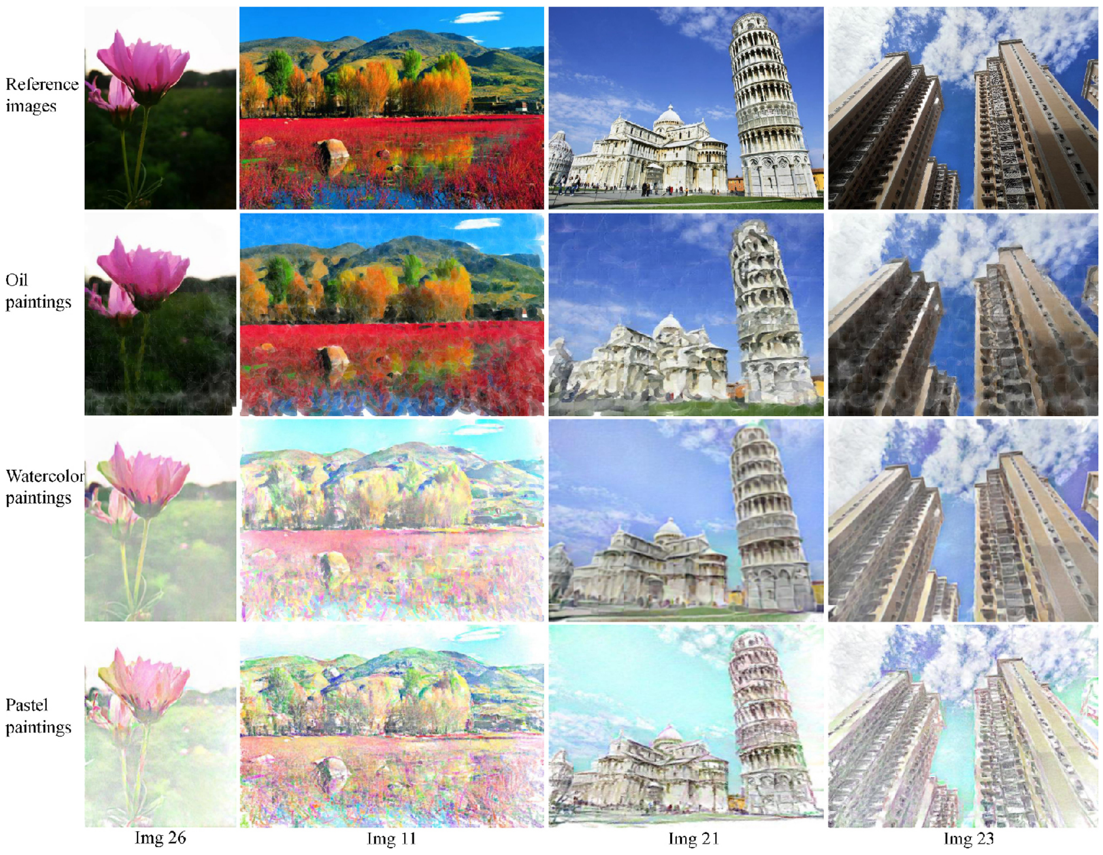

Fig. 8 Three artistic styles of paintings generated by Stroke-GAN Painter using images from real-world photos.

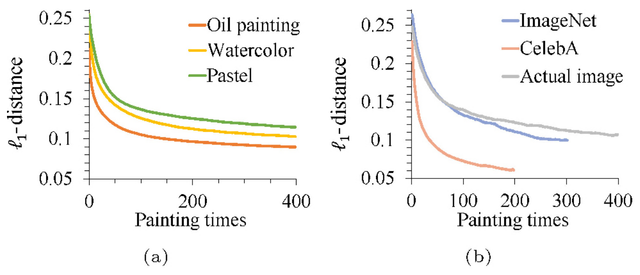

Fig. 9 The ℓ_1-distance between the generated images and reference images. (a) ℓ_1-distance of different stroke styles for real-world photos. (b) ℓ_1-distance of different datasets for the watercolor-stroke style.

SBR methods. We do so using two representative artistic styles for our model: pastel-stroke painting and oil-stroke painting. We use NP for comparative pastel-stroke paintings, and MDRLP, SNP, and PaintTF for comparative oil paintings. In order to obtain the best paintings generated by the compared methods, we use the authors' pre-trained models and default parameter values .

4.3.2 Qualitative comparison

Figures 10 and 11 compare paintings generated by our painter and these compared models. Figure 10 compares pastel paintings generated by NP and our painter. Our painter generates images with more details and textures than NP. For example, we cannot see facial texture and teeth in the woman's portrait generated by NP while the image generated by our painter well preserves those details, thereby looking more vivid. NP [15] only generates fixed strokes, while our Stroke-GAN generates variable strokes thanks to the coloring module. This allows strokes to be tuned according to the input image, thus retaining more details. Figure 11 compares oil paintings generated by MDRLP, SNP, PaintTF, and our painter. Images generated by our painter suffer less content loss than those generated by MDRLP, SNP, and PaintTF. It is quite obvious when comparing inset close-ups, e.g., our painter well preserves details of the man's eyes and mouth, and textures of the motorcycle and the cloud. Our model uses the independent Stroke-GAN to generate strokes with diverse shapes and variable sizes so can depict detailed contents. However, brushstrokes used in SNP and PaintTF have few shape variants as they are directly generated by their entire models. In particular, their models have only two shapes of strokes despite variant stroke size and angles. On the other hand, the stroke-texture representation differs between all compared methods. MDRLP presents stroke textures while losing some content since it has no special process during stroking to mimic the stroke textures. Adding more strokes and painting steps can make the result more similar to the input photo instead of a painting. SNP and PaintTF present stroke textures by adding a stroke-texture mask after stroke generation. In other words, the stroke contains no textures when generated and the textures have no affect on optimizing the stroke. Our Stroke-GAN painter renders the stroke texture by generating a sharp contour that mimics the thick edge of oil paints in a stroke. Therefore, the painting results present irregular-line textures instead of the brush textures.

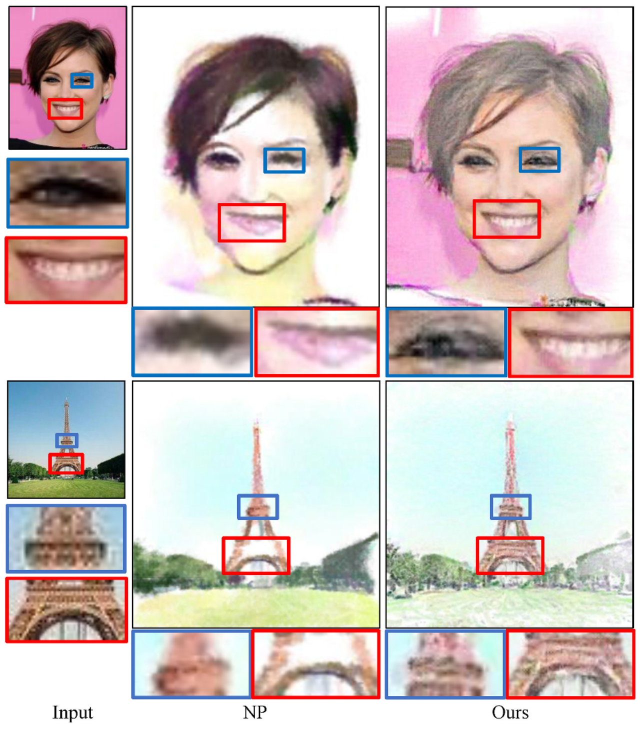

Fig. 10 Comparison to the prior method: pastel-stroke paintings generated by our painter and NP [15].

4.3.3 Quantitative comparison

To further compare the quality of paintings generated by Stroke-GAN Painter and the other methods, we conducted a two-step user study inspired by Refs. [37, 38]. For fairness, the experiments were blind trials, in which users did not know which paintings were generated by which methods. User Study I investigated relative preferences for paintings generated by these methods. User Study II investigated preservation of detailed content and stroke textures in paintings generated by these methods.

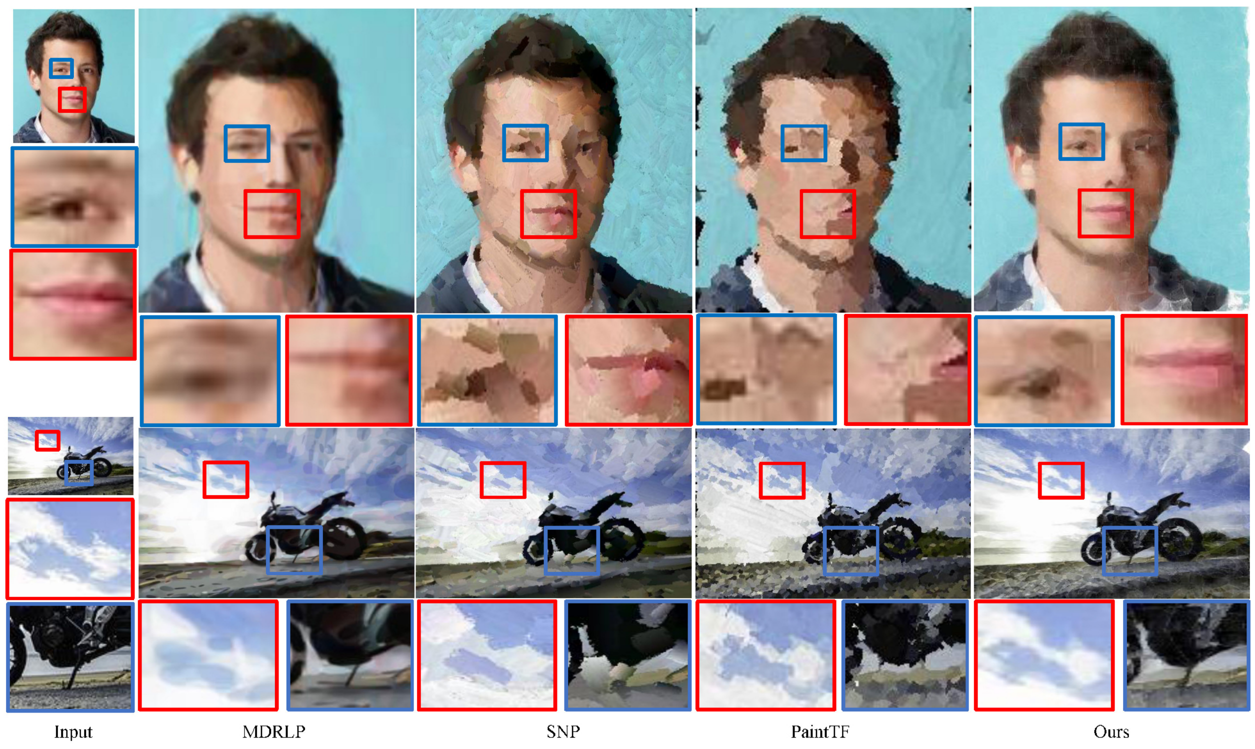

Fig. 11 Comparison results: oil-painting-stroke paintings generated by our painter, MDRLP [12], SNP [22], and PaintTF [23].

User Study I. User Study I used two questionnaires, the first one to compare pastel style artworks, and the second one to compare oil-painting style artworks. Since User Study I was designed to evaluate the preferences of people for artworks generated by different methods, we did not emphasize the backgrounds of users in the comparison, although they came from both artistic and non-artistic backgrounds.

In the first questionnaire, we arbitrarily choose 20 images from CelebA [35] (3 images), ImageNet [36] (6 images), and real-world photos (11 photos); the images included various types of contents including landscapes, buildings, animals, and portraits. The first group of participants had various backgrounds (10% with artistic training), age groups (17–50), and gender (44 females, 43 males). We evaluated pastel-stroke paintings generated by our painter and NP [15]. For each reference image (images numbered 1–20 in Fig. 12), we obtained a pair of images painted by our painter and NP. We evaluated the user *preference* for and *stylization* of generated images: we asked participants to choose which image better represented a pastel-stroke painting and which they preferred in each pair of images. Figure 12(a) depicts the results. Most users picked the results created by Stroke-GAN Painter as their preference, for all pairs of paintings; our paintings gained 77% of all votes on average. These high votes imply that paintings from Stroke-GAN Painter present pastel-painting style better than the compared ones.

Similarly, the second questionnaire evaluated the oil-painting effect, comparing our painter, MDRLP [12], SNP [22], and PaintTF [23]. The second group of participants was also chosen to have diverse backgrounds, age, and gender (40 females, 32 males). We also select 20 images from CelebA, ImageNet, and real-world photos to cover different content types (images numbered 21–40 in Fig. 12). We asked participants to choose which image is closer to an oil painting and which they preferred in each set of images. Again, more participants (31% among four methods) voted paintings generated by our method presenting better oil-painting style than those from other methods.

User Study II. We used the second user study to compare paintings generated by our Stroke-GAN Painter and other methods in terms of content detail and stroke textures. We again used two questionnaires (on a Likert scale [39]) for pastel paintings and oil paintings, separately. The participants were divided into two groups: users with and without an artistic

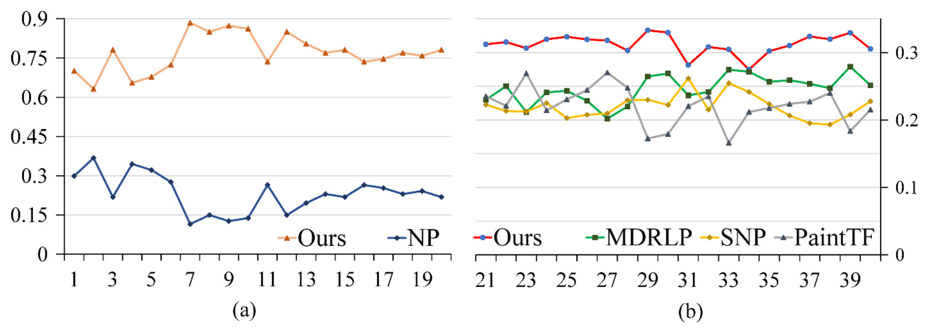

Fig. 12 User Study I. (a) Pastel-stroke-painting results generated by Stroke-GAN Painter and NP [15]. (b) Oil-painting results created by Stroke-GAN Painter, MDRLP [12], SNP [22], and PaintTF [23]. Vertical axis: percentage of users' preferences for an image. Horizontal axis: numbered image pairs.

background. All participants were chosen from various age groups (17–50) and gender (20 females, 5 males) for each questionnaire. We compared the average score (μ), variance (σ), and the 95% confidence interval for paintings generated by each method. Tables 1 and 2 give results for the two user groups, respectively.

In Table 1 (users without an artistic background), the content details of paintings generated by MDRLP [12], PaintTF [23], and SNP [22] gained low scores (lower than 3). One reason lies in the fact that their paintings lose too many details, and the stroke textures generated by MDRLP [12] are also difficult to recognize for most users. In contrast, the paintings generated by Stroke-GAN Painter have better scores for both content details and stroke textures. Similarly, in Table 2 (users with an artistic background), our method again gained higher evaluation scores than the other methods. Comparing Table 1 to Table 2, users lacking an artistic background gave higher scores than users with an artistic background in most cases. Nevertheless, both users with and without artistic backgrounds evaluated our paintings higher than those of other methods. In particular, the average score (μ) for the content details reaches 3.940 and the upper bound is 4.060 (in the 95% confidence interval) in Table 1. In Table 2, the average score of the stroke texture reaches 3.956 with upper bound 4.141. Interestingly, for stroke textures, users without artistic background gave a higher score (2.816) for MDRLP [12] than users with an artistic background (2.583), and the highest score given by users without artistic background is 3.468 (our method). However, opposite scores are given by

Table 1 User Study II. Scores of paintings generated by our method and other methods for content details and stroke textures without artistic background. CI = confidence interval. LB = lower bound. UB = upper bound

Item	Method	μ	σ	95% CI	
				LB	UB
Content	NP	3.626	0.181	3.530	3.723
	Ours	**3.940**	0.227	3.820	**4.060**
	MDRLP	2.839	0.308	2.683	2.996
	PaintTF	2.471	0.386	2.274	2.668
	SNP	2.374	0.298	2.153	2.595
	Ours	3.361	0.214	3.251	3.611
Stroke	NP	3.605	0.265	3.465	3.746
	Ours	3.722	0.258	3.585	3.859
	MDRLP	2.816	0.286	2.670	2.962
	PaintTF	2.582	0.368	2.394	2.769
	SNP	2.576	0.294	2.358	2.794
	Ours	3.468	0.201	3.366	3.571

Table 2 User Study II. Scores of paintings generated by our methods and other methods for content details and stroke textures with artistic background. CI = confidence interval. LB = lower bound. UB = upper bound

Item	Method	μ	σ	95% CI	
				LB	UB
Contents	NP	3.446	0.225	3.258	3.635
	Ours	3.775	0.229	3.584	3.967
	MDRLP	2.717	0.446	2.459	2.974
	PaintTF	2.397	0.559	2.074	2.719
	SNP	2.237	0.567	1.909	2.564
	Ours	3.803	0.313	3.623	3.984
Strokes	NP	3.594	0.286	3.355	3.833
	Ours	**3.956**	0.221	3.772	**4.141**
	MDRLP	2.583	0.398	2.353	2.813
	PaintTF	3.668	0.214	3.544	3.791
	SNP	3.682	0.153	3.594	3.770
	Ours	3.604	0.239	3.466	3.742

users with artistic background. The scores given for SNP [22] and PaintTF [23] are higher than those for our method although our score is 3.604, close to these two methods.

In summary, both pastel and oil paintings from our method contain more detailed contents than do the compared methods. For non-artistic users, the stroke textures are not well presented by most methods, while artistic users think that PaintTF [23], SNP [22], and our methods (both pastel and oil-painting) present stroke textures well.

User Study III. In order to evaluate the aesthetics of the results, we further conducted User Study III to evaluate color tone and aesthetic beauty of the output paintings. We again used two user-study questionnaires (using a Likert scale [39]) for pastel-stroke paintings and oil-painting-stroke paintings, separately. The input images were those used in User Study II. The participants were divided into two groups: users with an artistic background (15) and users without an artistic background (19). The participants came from various age groups (21–40), with 18 females and 16 males, for each questionnaire. We compare the average score (μ), variance (σ), and the 95% confidence interval for paintings generated by each method. Tables 3 and 4 give results for the two user groups.

In Table 3, scores for color tone and aesthetic beauty given by users without artistic background for pastel paintings (NP [15] and our method) are higher

Table 3 User Study III. Scores of paintings generated by our methods and SOTA methods for color tone and aesthetic beauty without artistic background. CI = confidence interval. LB = lower bound. UB = upper bound

Item	Method	μ	σ	95% CI	
				LB	UB
Color tone	NP	**3.703**	0.316	3.535	**3.870**
	Ours	3.688	0.306	3.526	3.851
	MDRLP	3.211	0.274	3.071	3.350
	PaintTF	2.934	0.363	2.749	3.119
	SNP	2.845	0.346	2.588	3.101
	Ours	3.708	0.190	3.611	3.805
Beauty	NP	3.782	0.386	3.577	3.986
	Ours	3.833	0.293	3.678	3.988
	MDRLP	2.963	0.327	2.796	3.130
	PaintTF	2.537	0.380	2.343	2.731
	SNP	2.484	0.356	2.220	2.748
	Ours	3.508	0.188	3.412	3.604

Table 4 User Study III. Scores of paintings generated by our methods and SOTA methods for color tone and aesthetic beauty with artistic background. CI = confidence interval. LB = lower bound. UB = upper bound

Item	Method	μ	σ	95% CI	
				LB	UB
Color tone	NP	3.857	0.243	3.654	4.060
	Ours	3.913	0.250	3.704	4.121
	MDRLP	3.823	0.238	3.686	3.961
	PaintTF	3.607	0.290	3.439	3.774
	SNP	3.743	0.237	3.606	3.880
	Ours	**4.217**	0.357	4.010	4.423
Beauty	NP	3.707	0.346	3.418	3.996
	Ours	3.753	0.218	3.571	3.935
	MDRL	3.550	0.327	3.361	3.739
	PaintTF	3.267	0.555	2.946	3.587
	SNP	3.470	0.454	3.208	3.732
	Ours	**3.937**	0.400	3.706	4.167

than 3. On the other hand, oil-paintings obtained lower scores. In particular, paintings generated by PaintTF [23] and SNP [22] gained much lower scores than our method and MDRLP [12] for aesthetic beauty. As the scores for content in Table 1 (given by users without artistic background) are also lower than 3, we see that the paintings generated by PaintTF [23] and SNP [22] lose too much content detail, so users without an artistic background give low scores for beauty. Although users without artistic background did not give high scores, the ranking of the compared methods for aesthetic beauty item and color-tone remain the same.

On the other hand, users with an artistic background gave higher scores than users without an artistic background. In Table 4, paintings generated by all compared methods obtain scores higher than 3. In particular, our paintings score 4.217 for color tone and 3.937 for aesthetic beauty. Comparing Tables 4 and 3, our method, MDRLP [12], PaintTF [23], and SNP [22] achieve much higher scores than NP [15]. Evaluation of pastel-stroke paintings generated by NP differ little between users with and without artistic backgrounds. However, the difference between these two kinds of users is obvious when evaluating the oil-painting style paintings. For example, for aesthetic beauty, users without artistic background give higher scores for paintings by PaintTF than by SNP while users with artistic background give lower scores. However, all users give consistent evaluations.

In particular, for both color tone and aesthetic beauty, our method is better than the others, while PaintTF [23] and SNP [22] rank lowest. Considering Tables 1–4 overall, our method scores the highest among the state-of-the-art methods. NP performs well for both content and color tone. MDRLP performs well for color tone. PaintTF and SNP perform well for stroke texture.

4.4 Alternatives

In this section, we investigate how our painter generates different styles of paintings when using alternative FENs and strokes.

4.4.1 Feature-extraction network

Recall that we chose GoogleNet as the FEN for watercolor and pastel-stroke images, and ResNet as the FEN for oil-painting images. We consider a new FEN with a combination of GoogleNet and ResNet, namely (G+R) to generate paintings. We denote the ℓ_1-distance of features extracted by GoogleNet and the ℓ_1-distance of features extracted by ResNet by $\mathcal{L}_{\mathrm{G}}(U_0, C_n)$ and $\mathcal{L}_{\mathrm{R}}(U_0, C_n)$, respectively: see Eq. (13). The loss function of the G+R network can be written as

$$\mathcal{L}(U_0, C_n) = 0.5\mathcal{L}_{\mathrm{G}}(U_0, C_n) + 0.5\mathcal{L}_{\mathrm{R}}(U_0, C_n) \quad (14)$$

We only perform backpropagation for the final $\mathcal{L}(U_0, C_n)$. Figure 13 compares results generated by the three different FENs and four stroke styles, Styles 1–4. Style 4 is a new stroke designed with a hollow circle and a cubic Bézier curve; its stroke dataset is generated using a similar method to that for Style 1. We see from Fig. 13 that the

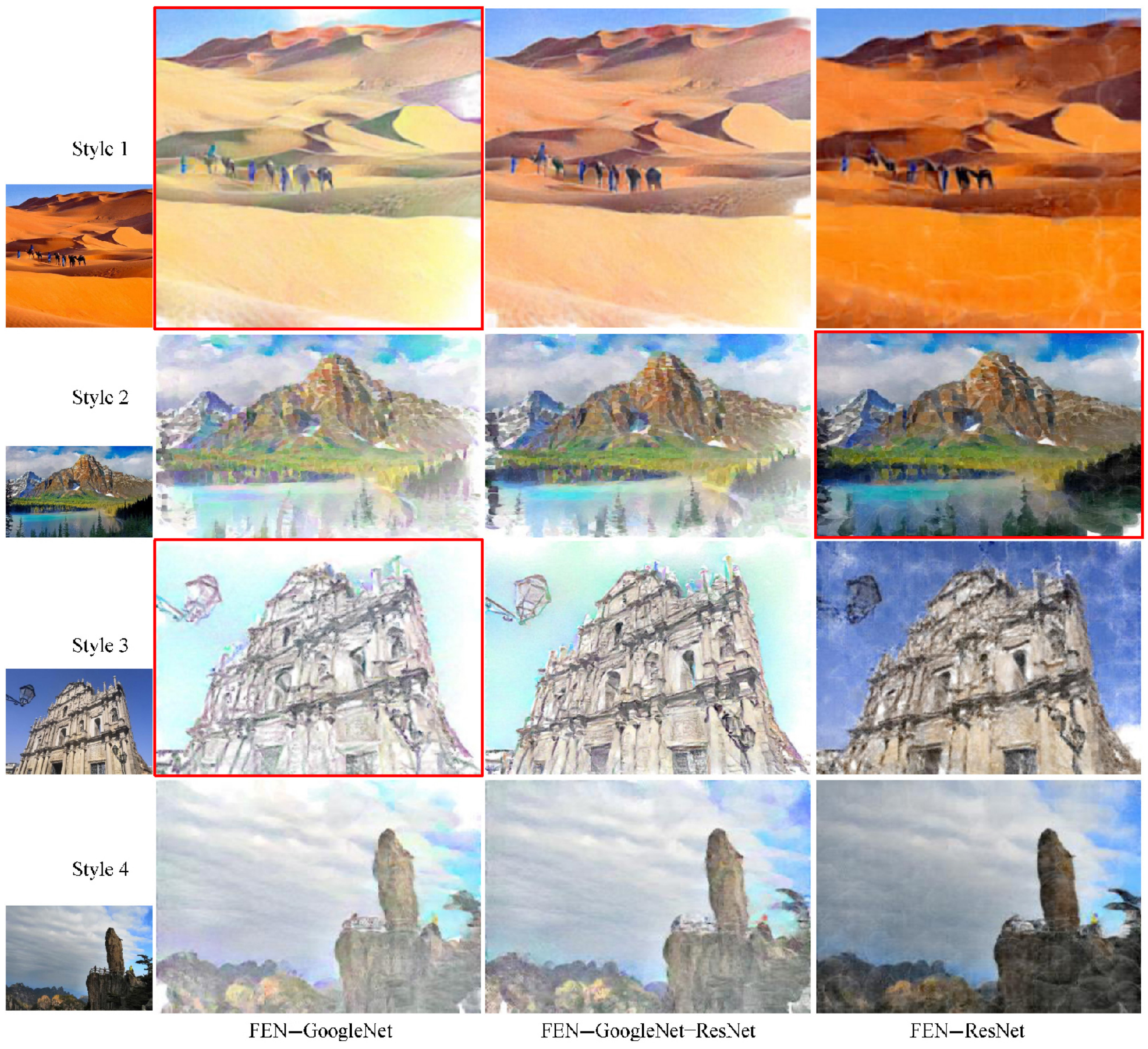

Fig. 13 Paintings by various feature-extracting networks (FENs) and different stroke styles. Each row contains images created by models with one style of strokes and an FEN (GoogleNet, GoogleNet+ResNet, or ResNet), with input images in the first column.

model using GoogleNet+ResNet as the FEN can generate a style of artwork combining the characters of pastel and oil-painting styles. Interestingly, the FEN essentially affects the *artistic style* of a painting, and the stroke style affects the *painting style* of the painting. Therefore, we confidently infer that various combinations of FENs and stroke styles can create a diversity of artistic styles of paintings.

4.4.2 Number of strokes

We next investigate the impact of the number of strokes. In particular, the canvas is divided into $h \times w$ cells. We then generate images with various number of strokes in each cell. Figure 14 depicts paintings generated using different numbers of strokes with the same FEN. The image generated using 2 strokes looks colorful and artistically creative although it also loses much content detail. Larger numbers of strokes (8 or 10) lead to an exquisite image, much closer to the reference image than images generated by fewer strokes (2 or 5). Having an adjustable number of strokes provides the users with artistic choices.

5 Conclusions and future work

In this paper, we have presented a stroke-based image rendering approach to mimic the human painting process and generate different styles of paintings. In particular, we designed Stroke-GAN to generate various styles of strokes. We model the painting process as a stroke-state-optimization process, which can be optimized by a deep convolutional neural network. Our artistic painter can generate different styles of paintings in a coarse-to-fine fashion, like a human painter. User studies of the paintings generated by Stroke-GAN Painter and competing methods demonstrate that our painter gained most votes for the closeness to pastel paintings and oil paintings. Moreover, images generated by our painter also preserve more content detail than existing methods.

A deep learning algorithm and Stroke-GAN are used to decompose the reference image into a grid of cells to allow rendering of a sequence of strokes to achieve the stroke-by-stroke effect. We designed Stroke-GAN to generate style strokes by learning from stroke datasets. Our Stroke-GAN can learn any stroke style, providing the painting agent with creativity and flexibility. Although our generated paintings do not compare with masters' artworks, we have made an important step for learning-based AI painting with creative and flexible artistic styles. Meanwhile, there is much that can be done to improve

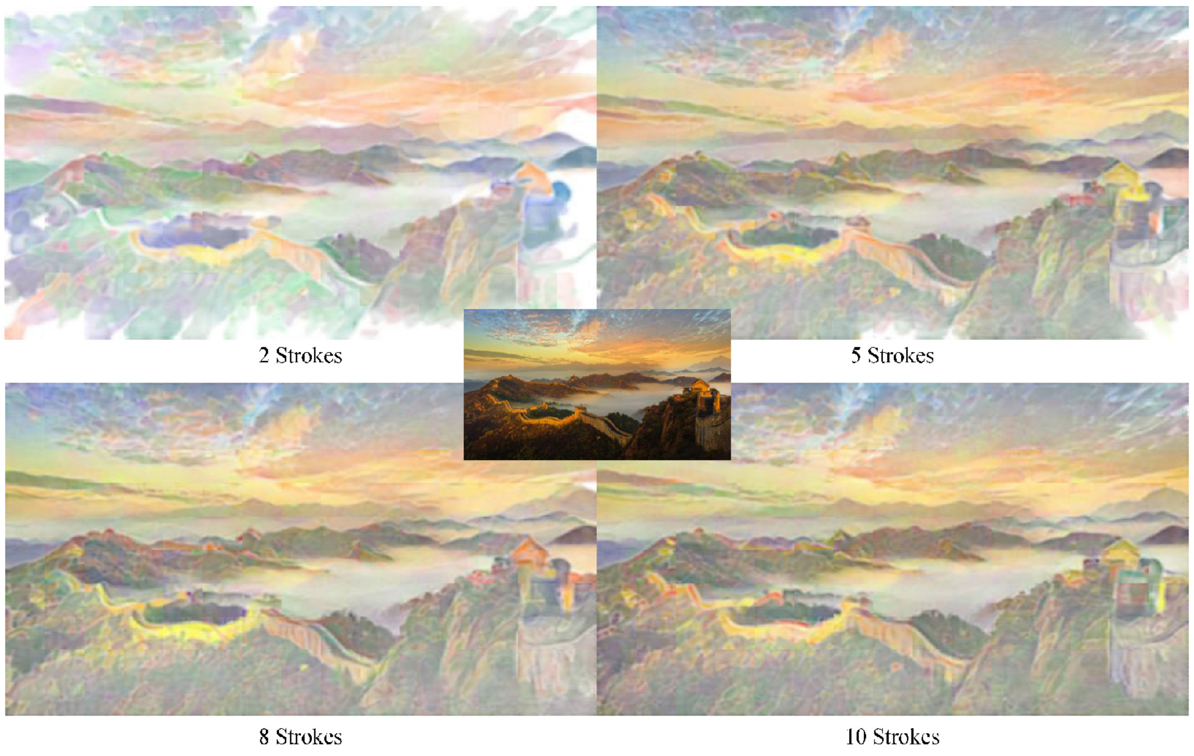

Fig. 14 Paintings generated using 2, 5, 8, or 10 strokes in each cell. Center: reference image.

the quality of the output, and other artistic styles not mentioned in this paper could also be emulated. In future, we can combine the advantages of conventional stroke-based methods with learning-based methods to improve painting quality. On the other hand, we hope to develop new style transfer methods in a stroke-by-stroke manner to enrich the artistic styles of AI painting.

Availability of data and materials

The data and materials generated during the study are available from the corresponding authors on reasonable request.

Author contributions

Q. Wang designed the study, performed experiments, and wrote the manuscript. P. Li and H.-N. Dai helped to design the study and experiments and supervised the project. C. Guo provided comments and feedback on the study and the results. All authors reviewed the manuscript.

Acknowledgements

The authors would like to thank the anonymous reviewers for their helpful suggestions and comments. This work was supported in part by the Hong Kong Institute of Business Studies (HKIBS) Research Seed Fund under Grant HKIBS RSF-212-004, and in part by The Hong Kong Polytechnic University under Grant P0030419, Grant P0030929, and Grant P0035358.

Declaration of competing interest

The authors have no competing interests to declare that are relevant to the content of this article.

References

[1] Wang, L.; Wang, Z.; Yang, X. S.; Hu, S. M.; Zhang, J. J. Photographic style transfer. *The Visual Computer* Vol. 36, No. 2, 317–331, 2020.

[2] Gatys, L. A.; Ecker, A. S.; Bethge, M. Image style transfer using convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2414–2423, 2016.

[3] Zhao, Y. R.; Deng, B.; Huang, J. Q.; Lu, H. T.; Hua, X. S. Stylized adversarial AutoEncoder for image generation. In: Proceedings of the 25th ACM International Conference on Multimedia, 244–251, 2017.

[4] Zhou, W. Y.; Yang, G. W.; Hu, S. M. Jittor-GAN: A fast-training generative adversarial network model zoo based on Jittor. *Computational Visual Media* Vol. 7, No. 1, 153–157, 2021.

[5] Haeberli, P. Paint by numbers: Abstract image representations. *ACM SIGGRAPH Computer Graphics* Vol. 24, No. 4, 207–214, 1990.

[6] Hertzmann, A. Painterly rendering with curved brush strokes of multiple sizes. In: Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques, 453–460, 1998.

[7] Lee, H.; Seo, S.; Ryoo, S.; Ahn, K.; Yoon, K. A multi-level depiction method for painterly rendering based on visual perception cue. *Multimedia Tools and Applications* Vol. 64, No. 2, 277–292, 2013.

[8] Ha, D.; Eck, D. A neural representation of sketch drawings. In: Proceedings of the International Conference on Learning Representations, 1–16, 2018.

[9] Zheng, N.; Jiang, Y.; Huang, D. StrokeNet: A neural painting environment. In: Proceedings of the International Conference on Learning Representations, 1–12, 2019.

[10] Ganin, Y.; Kulkarni, T.; Babuschkin, I.; Eslami, S. M. A.; Vinyals, O. Synthesizing programs for images using reinforced adversarial learning. In: Proceedings of the 35th International Conference on Machine Learning, 1666–1675, 2018.

[11] Xie, N.; Hachiya, H.; Sugiyama, M. Artist agent: A reinforcement learning approach to automatic stroke generation in oriental ink painting. *IEICE Transactions on Information and Systems* Vol. E96.D, No. 5, 1134–1144, 2013.

[12] Huang, Z. W.; Zhou, S. C.; Heng, W. Learning to paint with model-based deep reinforcement learning. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, 8708–8717, 2019.

[13] Wang, Q.; Guo, C.; Dai, H. N.; Li, P. Self-stylized neural painter. In: Proceedings of the SIGGRAPH Asia 2021 Posters, Article No. 9, 2021.

[14] Song, J. F.; Pang, K. Y.; Song, Y. Z.; Xiang, T.; Hospedales, T. M. Learning to sketch with shortcut cycle consistency. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 801–810, 2018.

[15] Nakano, R. Neural painters: A learned differentiable constraint for generating brushstroke paintings. In: Proceedings of the 33rd Conference on Neural Information Processing Systems, 2019.

[16] Deussen, O.; Strothotte, T. Computer-generated pen-and-ink illustration of trees. In: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, 13–18, 2000.

[17] Wilson, B.; Ma, K. L. Rendering complexity in computer-generated pen-and-ink illustrations. In: Proceedings of the 3rd International Symposium on Non-photorealistic Animation and Rendering, 129–137, 2004.

[18] Deussen, O.; Hiller, S.; Van Overveld, C.; Strothotte, T. Floating points: A method for computing stipple drawings. *Computer Graphics Forum* Vol. 19, No. 3, 41–50, 2000.

[19] Deussen, O.; Isenberg, T. Halftoning and stippling. In: *Image and Video-based Artistic Stylisation. Computational Imaging and Vision, Vol. 42.* Rosin, P.; Collomosse, J. Eds. Springer London, 45–61, 2013.

[20] Hertzmann, A. A survey of stroke-based rendering. *IEEE Computer Graphics and Applications* Vol. 23, No. 4, 70–81, 2003.

[21] Mellor, J.; Park, E.; Ganin, Y.; Babuschkin, I.; Kulkarni, T.; Rosenbaum, D.; Ballard, A.; Weber, T.; Vinyals, O.; Eslami, S. M. A. Unsupervised doodling and painting with improved SPIRAL. In: Proceedings of the Neural Information Processing Systems Workshops, 2019.

[22] Zou, Z. X.; Shi, T. Y.; Qiu, S.; Yuan, Y.; Shi, Z. W. Stylized neural painting. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 15684–15693, 2021.

[23] Liu, S. H.; Lin, T. W.; He, D. L.; Li, F.; Deng, R. F.; Li, X.; Ding, E.; Wang, H. Paint transformer: Feed forward neural painting with stroke prediction. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, 6578–6587, 2021.

[24] Gatys, L.; Ecker, A.; Bethge, M. A neural algorithm of artistic style. *Journal of Vision* Vol. 16, No. 12, 326, 2016.

[25] Jing, Y. C.; Yang, Y. Z.; Feng, Z. L.; Ye, J. W.; Yu, Y. Z.; Song, M. L. Neural style transfer: A review. *IEEE Transactions on Visualization and Computer Graphics* Vol. 26, No. 11, 3365–3385, 2020.

[26] Dutta, T.; Singh, A.; Biswas, S. StyleGuide: Zero-shot sketch-based image retrieval using style-guided image generation. *IEEE Transactions on Multimedia* Vol. 23, 2833–2842, 2021.

[27] Xu, M. L.; Su, H.; Li, Y. F.; Li, X.; Liao, J.; Niu, J. W.; Lv, P.; Zhou, B. Stylized aesthetic QR code. *IEEE Transactions on Multimedia* Vol. 21, No. 8, 1960–1970, 2019.

[28] Chu, W. T.; Wu, Y. L. Image style classification based on learnt deep correlation features. *IEEE Transactions on Multimedia* Vol. 20, No. 9, 2491–2502, 2018.

[29] Jia, B.; Fang, C.; Brandt, J.; Kim, B.; Manocha, D. PaintBot: A reinforcement learning approach for natural media painting. *arXiv preprint* arXiv: 1904.02201, 2019.

[30] Justin, R. Elements of art: Interpreting meaning through the language of visual cues. Ph.D. Thesis. Stony Brook University, 2018.

[31] Radford, A.; Metz, L.; Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. In: Proceedings of the International Conference on Learning Representations, 1–16, 2016.

[32] Donoho, D. L. De-noising by soft-thresholding. *IEEE Transactions on Information Theory* Vol. 41, No. 3, 613–627, 1995.

[33] Szegedy, C.; Liu, W.; Jia, Y. Q.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1–9, 2015.

[34] He, K. M.; Zhang, X. Y.; Ren, S. Q.; Sun, J. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 770–778, 2016.

[35] Liu, Z. W.; Luo, P.; Wang, X. G.; Tang, X. O. Deep learning face attributes in the wild. In: Proceedings of the IEEE International Conference on Computer Vision, 3730–3738, 2015.

[36] Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S. A.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision* Vol. 115, No. 3, 211–252, 2015.

[37] Huang, H. Z.; Zhang, S. H.; Martin, R. R.; Hu, S. M. Learning natural colors for image recoloring. *Computer Graphics Forum* Vol. 33, No. 7, 299–308, 2014.

[38] Tong, Z. Y.; Chen, X. H.; Ni, B. B.; Wang, X. H. Sketch generation with drawing process guided by vector flow and grayscale. *Proceedings of the AAAI Conference on Artificial Intelligence* Vol. 35, No. 1, 609–616, 2021.

[39] Liddell, T. M.; Kruschke, J. K. Analyzing ordinal data with metric models: What could possibly go wrong? *Journal of Experimental Social Psychology* Vol. 79, 328–348, 2018.

Qian Wang received her B.Eng. degree in electronic information engineering from Yangtze University, Jingzhou, China, in 2012, and her M.Eng. degree in educational technology from Zhejiang University of Technology, Hangzhou, China, in 2016. She is currently pursuing a Ph.D. degree in computer technology and applications in the School of Computer Science and Engineering, Macau University of Science and Technology. She is also a research assistant with The Hong Kong Polytechnic University. Her current research interests include image and video stylization, and AI drawing.

Cai Guo received his M.Eng. degree in software engineering from Guangdong University of Technology, Guangzhou, China, in 2011. He is currently pursuing a Ph.D. degree in computer technology and applications in the School of Computer Science and Engineering, Macau University of Science and Technology. He is also with Hanshan Normal University, Chaozhou, China. His current research interests include deep learning, motion deblurring, and AI drawing.

Hong-Ning Dai received his Ph.D. degree in computer science and engineering from The Chinese University of Hong Kong, in 2008. He is currently an associate professor in the Department of Computer Science, Hong Kong Baptist University, Hong Kong, China. He was in the Faculty of Information Technology at Macau University of Science and Technology as an assistant/associate professor from 2010 to 2021, and the Department of Computing and Decision Sciences, Lingnan University, Hong Kong, China, as an associate professor from 2021 to 2022. His current research interests include Internet of Things, big data analytics, and blockchains. He has co-authored or co-edited 3 monographs and published more than 150 papers in top-tier journals and conferences.

Ping Li received his Ph.D. degree in computer science and engineering from The Chinese University of Hong Kong, in 2013. He is currently an assistant professor with The Hong Kong Polytechnic University. He has published many top-tier scholarly research papers and has one excellent research project reported worldwide by *ACM TechNews*. His current research interests include artistic rendering and synthesis, stylization, colorization, and creative media.