CartoonLossGAN: Learning Surface and Coloring of Images for Cartoonization

Yongsheng Dong, *Senior Member, IEEE*, Wei Tan, Dacheng Tao, *Fellow, IEEE*, Lintao Zheng, and Xuelong Li, *Fellow, IEEE*

***Abstract*— Cartoonization as a special type of artistic style transfer is a difficult image processing task. The current existing artistic style transfer methods cannot generate satisfactory cartoon-style images due to that artistic style images often have delicate strokes and rich hierarchical color changes while cartoon-style images have smooth surfaces without obvious color changes, and sharp edges. To this end, we propose a cartoon loss based generative adversarial network (CartoonLossGAN) for cartoonization. Particularly, we first reuse the encoder part of the discriminator to build a compact generative adversarial network (GAN) based cartoonization architecture. Then we propose a novel cartoon loss function for the architecture. It can imitate the process of sketching to learn the smooth surface of the cartoon image, and imitate the coloring process to learn the coloring of the cartoon image. Furthermore, we also propose an initialization strategy, which is used in the scenario of reusing the discriminator to make our model training easier and more stable. Extensive experimental results demonstrate that our proposed CartoonLossGAN can generate fantastic cartoon-style images, and outperforms four representative methods.**

***Index Terms*— Deep learning, generative adversarial networks, cartoonization.**

I. Introduction

CARTOON originated in the Netherlands in the seventeenth century, and has widely been used in a variety of fields including propaganda, movies, advertisements, children's education, and the like. Traditional cartoon images are generated from the real world by manual drawing. This requires the creator to have a very solid art foundation and the creation process is time-consuming. Cartoonization [1] is to generate cartoon images or cartoon-style images automatically from real images, and can be seen as a special type of artistic style transfer that generates automatically the artistic style images retaining the semantic content of the real-world scene.

Manuscript received June 7, 2021; revised September 8, 2021, October 15, 2021, and November 12, 2021; accepted November 13, 2021. Date of publication December 2, 2021; date of current version December 15, 2021. This work was supported in part by the National Natural Science Foundation of China under Grant 62071171 and Grant 61871470 and in part by the Program for Science and Technology Innovation Talents in Universities of Henan Province under Grant 19HASTIT026. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Ran He. *(Corresponding author: Yongsheng Dong.)*

Yongsheng Dong, Wei Tan, and Lintao Zheng are with the School of Information Engineering, Henan University of Science and Technology, Luoyang 471023, China (e-mail: dongyongsheng98@163.com).

Dacheng Tao is with the JD Explore Academy in JD.com, Beijing 100176, China (e-mail: dacheng.tao@gmail.com).

Xuelong Li is with the School of Artificial Intelligence, Optics and Electronics (iOPEN), Northwestern Polytechnical University, Xi'an 710072, China (e-mail: li@nwpu.edu.cn).

Digital Object Identifier 10.1109/TIP.2021.3130539

In the mid-1990s, the field of non-photorealistic rendering (NPR) [2] in computer graphics made initial attempts at artistic style transfer and made impressive advances. In the last decade, many artistic style transfer methods have been proposed by using deep neural network models. Gatys *et al.* [3] used the pre-trained visual geometry group network (VGG) [4] to extract features and perform optimization for artistic style transfer. Recently, generative adversarial networks (GAN) [5]–[7] have shown amazing image generation capabilities and have quickly become a popular research topic in the field of computer vision. Isola *et al.* [8] proposed a general framework for artistic style transfer through paired training data. The problem is that paired data acquisition is expensive and difficult. Zhu *et al.* [9] achieved artistic style transfer on unpaired training data in a cyclical manner, which is called cycle-consistent adversarial networks (CycleGAN). Since then, there have been a lot of studies to improve the CycleGAN model, including multi-domain image-to-image translation [10]–[12], improving the quality of generated images [13]–[15], controllable image-to-image translation [16], simplifing the network architecture [17], [18], and artistic style transfer [19], [20].

Although the above methods can generate impressive artistic style images, they often fail to produce satisfactory cartoon images. Because cartoon images have smooth surfaces without obvious color changes and clear and sharp edges. On the contrary, other types of artistic style images often have delicate strokes and dramatic, rich, very hierarchical color changes. Chen *et al.* [1] proposed a GAN-based method (named CartoonGAN) to achieve amazing cartoonization by proposing a simple edge-promoting adversarial loss. However, the training process of CartoonGAN is unstable and may fail in some cases. Pęśko *et al.* [21] improves the stability of CartoonGAN training by improving the network architecture, training strategy, and initialization strategy of CartoonGAN. Chen *et al.* [22] proposed a lightweight GAN-based method (named AnimeGAN) to achieve fast animation style transfer. However, the brightness of the image generated by AnimeGAN is dark. In addition, the color reconstruction loss in AnimeGAN compares the images generated by the generator with the reference content images, which may inhibit cartoonization. On the other hand, to construct a compact and informative translation framework, a generative adversarial network based on no independent component for encoding (NICE-GAN) reused the discriminator for encoding the images of the target domain [18]. However, NICE-GAN trained two generators and two discriminators, and thus the resulting translation framework maybe not compact enough.

Inspired by the cartoon creation process of sketching first and then coloring, in this paper we propose a generative adversarial network framework called CartoonLossGAN, which achieves cartoonization by learning the surface and color of cartoon-style images separately. We make a simple rule, i.e., color images are only compared with color images, and grayscale images are only compared with grayscale images, to prevent incorrect color mapping. According to this rule, we propose two novel loss functions to imitate the process of sketching and coloring in the cartoon creation process. Unlike NICE-GAN, we do not use the cyclic strategy and the multi-scale discriminator. We only use one generator and one simple discriminator (e.g., n layer discriminator of CycleGAN), and satisfactory results are observed. For the scene of reusing discriminator, we also propose an initialization strategy to make the training process more stable. We summarize our main contributions as follows:

- We propose a cartoon loss based generative adversarial network, CartoonLossGAN, trained on unpaired training data for cartoonization. We reuse the encoder part of the discriminator without using a cyclic manner and a complex multi-scale discriminator to build a compact generative adversarial network (GAN) based cartoonization architecture.
- We propose a novel cartoon loss function to imitate the process of sketching and coloring in the cartoon creation process. Our proposed cartoon loss can learn the smooth surface and the coloring of cartoon images.
- We propose an initialization strategy used for reusing discriminator, which makes the training process easier and more stable.
- Extensive experimental results demonstrate that our proposed CartoonLossGAN can generate fantastic cartoon-style images, and outperforms four representative methods.

This paper is organized as follows. Section II presents the related work. Section III presents our proposed CartoonLossGAN method. The experimental results are given in Section IV. Finally, Section V briefly concludes the paper.

II. Related Work

In this section, we review some related works including non-photorealistic rendering, neural style transfer, and GAN-based methods.

A. Non-Photorealistic Rendering

The earliest research on artistic style transfer was conducted by non-photorealistic rendering (NPR) and made impressive advances. Among them, stroke-based rendering [2] imitates the process of virtual strokes smearing on the canvas to render the image into a specific style. Image analogies [23] learns a mapping from content images to related style images through paired training data. Winnemöller *et al.* [24] use image filtering to render a given image to produce a cartoon image. However, stroke-based rendering can only learn a specific style at a time, and it is difficult to extend to other styles. The paired training data used in image analogies are usually difficult to collect. Image filtering is easy to implement, but the types of styles it can simulate are limited. In addition, all of the above NPR methods use low-level image features, which will cause distortion of the generated image structure.

B. Neural Style Transfer

In the last decade, deep neural network based artistic style transfer methods have been proposed. Gatys *et al.* [3] used the pre-trained VGG network [4] to extract image features from real-world scenes and single style images, and continuously optimized a noisy image in an iterative manner to achieve artistic style transfer. Gatys *et al.* [3] achieved neural style transfer [25] for the first time. However, their image optimization method requires a lot of time for each artistic style transfer of an image. In order to solve the above problems, Johnson *et al.* [26] proposed a model optimization method, i.e., by training a convolutional neural network instead of optimizing an image in real-time, achieving rapid artistic style transfer. Ulyanov *et al.* [27] proposed a new normalization method (i.e., instance normalization) specificantlly for artistic style transfer, which significantly improved the quality of stylization. Zhang *et al.* [28] realized multi-style transfer via a single model, which greatly improves the practicality of the model.

C. GAN-Based Methods

Generative adversarial networks (GAN) [29] is a powerful generative model that generates realistic images through adversarial training of two neural networks. Isola el al. [8] proposed a conditional adversarial networks [30] (named pix2pix) that achieves artistic style transfer by using paired training data. However, paired training data is difficult to be collected in some sense. To alleviate this issue, Zhu *et al.* [9] proposed a CycleGAN [9] by adopting a cyclic training strategy. It can generate vivid artistic images through unpaired training data. However, CycleGAN can only learn the style of one artist at a time, and simply training multiple pairs of CycleGAN models will lead to huge computational costs if multiple artists are needed. Anoosheh *et al.* [31] splits the encoder and decoder of the generator, and solves the above problems by reusing the encoder and decoder. Choi *et al.* [11] achieve multi-style transfer by using mask vector and a single model (i.e., using only one generator and one discriminator). Wang *et al.* [10] realize multi-domain image-to-image translation by sharing common information among multiple domains through a shared knowledge module. Li *et al.* [32] realize multi-domain weather translation through a segmentation module and a attention module.

On the other hand, to improve the quality of artistic style images generated by the CycleGAN model, Kim *et al.* [33] proposed a new attention module and adaptive layer-instance normalization (AdaLIN). Yi *et al.* [20] and Li *et al.* [13] implemented a novel asymmetric GAN in different ways to solve the problem that the images generated by CycleGAN will lose some local details (e.g., facial details). Saliency detail preservation generative adversarial networks (SDP-GAN) [14]

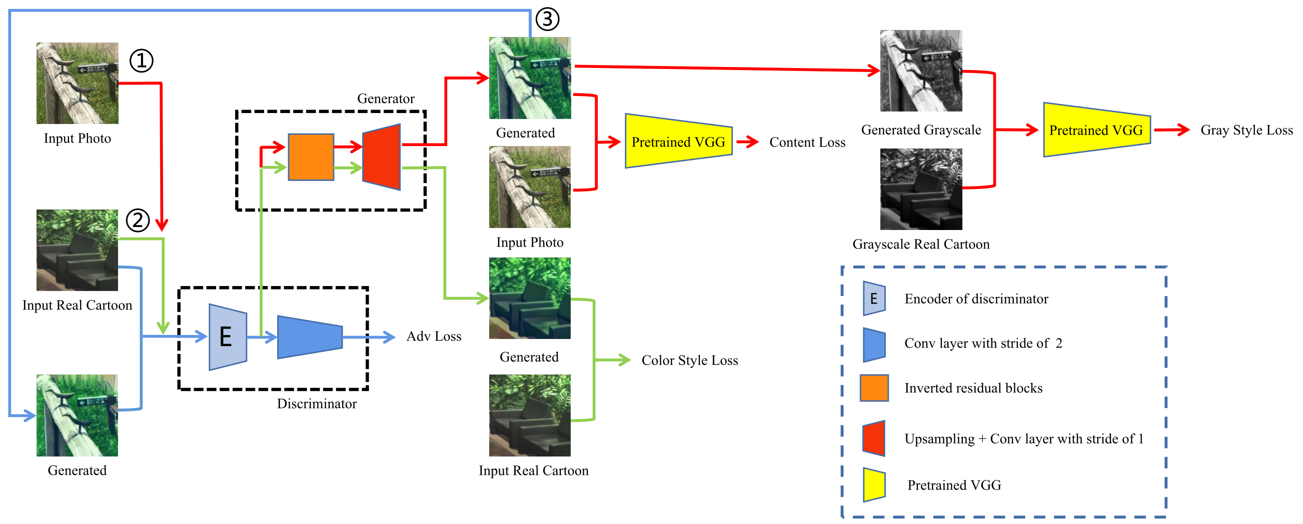


Fig. 1. The workflow of our proposed CartoonLossGAN. (1) "Input Photo" refers to the real-world scene image in the training data. "Input Real Cartoon" refers to a cartoon image of specific style in the training data. "Generated" refers to the cartoon style image generated by input photo or input real cartoon through the generator. "Generated Grayscale" refers to transforming the generated cartoon image into a grayscale image. "Grayscale Real Cartoon" refers to transforming input real cartoon into a grayscale image. (2) The encoder of the discriminator is also used for the generator. The encoder is only trained when training the discriminator and is fixed when training the generator. (3) The content loss is used to constrain the semantic content of content images. The gray style loss is used to generate a smooth surface. The color style loss is used to learn the coloring of real cartoon images. (4) We train the discriminator by adversarial loss, and we train the generator by adversarial loss, content loss, gray style loss, and color style loss.

achieves high perceptual quality style transfer by introducing a saliency network to retain the details of the salient regions. Gao *et al.* [15] use two complementary generators with different architectures and use a gated fusion network to combine the two generators to generate high-quality image-to-image translation results. Tang *et al.* [16] propose a novel controllable image-to-image translation method by providing additional target controllable structures as input. CartoonGAN [1] achieves amazing cartoonization by proposing a simple edge-promoting adversarial loss. Peśko *et al.* [21] improves the stability of CartoonGAN training by proposing a training strategy. AnimeGAN [22] is a lightweight GAN model that achieves impressive cartoonization through the proposed three novel loss functions.

III. Our Method

In this section, we describe our proposed CartoonLossGAN, which realizes cartoonization by learning the surface and color of cartoon-style images separately. In the following subsections, we first introduce our network architecture, and then give our proposed loss function, followed by our proposed initialization strategy.

A. CartoonLossGAN Architecture

The workflow of our proposed CartoonLossGAN is shown in Fig. 1. Given a real-world scene dataset $\{p_i\}_{i=1}^{N}$ where $p_i \in P \subset \mathbb{R}^{H\times W\times 3}$, and a cartoon dataset $\{c_j\}_{j=1}^{M}$ where $c_j \in C \subset \mathbb{R}^{H\times W\times 3}$. N is the number of photos in the real-world scene dataset. M is the number of cartoon images in the cartoon dataset. N is not equal to M, which means that the two datasets are unpaired. Our goal is to train a generator $G : P \to C$ that can automatically translate a given photo p into a desired cartoon-style image c. The discriminator D is used to determine whether the input image of D is a real cartoon image $c \in C$ or a cartoon image $\hat{c}$ generated by G.

The network architecture of CartoonLossGAN is shown in Fig. 2. In our proposed CartoonLossGAN, the generator and the discriminator share the encoder of the discriminator, and the encoder is trained by the decoupling strategy proposed by NICE-GAN [18] (i.e., the encoder only updates the parameters when training the discriminator, and fixes the parameters in other cases). As shown in Fig. 3, the difference between our proposed CartoonLossGAN and NICE-GAN is that we do not use the cyclic strategy (i.e., the discriminator *D_b* is paired with the generator *G_ab*, and the encoder *E_b* of *D_b* is used for the generator *G_ba*, vice versa) they emphasized and a powerful multi-scale discriminator. We only use the generator *G_ab* and the discriminator *D_b* and use the encoder *E_b* of *D_b* to replace the encoder part of *G_ab*, and satisfactory results are observed. We assume that the encoder *E_b* of the fully trained discriminator *D_b* can extract common image features for the generator *G_ab* (This view is similar to that proposed by Gatys *et al.* [3]). In addition, a simple discriminator (e.g., n layer discriminator of CycleGAN) can also achieve satisfactory results. In other words, the decoupled training strategy of NICE-GAN for reusing discriminator is a more general training strategy. For the scene of reusing discriminator, we propose an initialization strategy to make the training process more stable. This initialization strategy will be introduced in Section III-C.

In addition to simplifying the network architecture by reusing the discriminator, we also reduce the number of network parameters by replacing the residual block [34] with the inverted residual block [35]. The difference between our proposed CartoonLossGAN and AnimeGAN [22] is that we use common convolutional layer instead of depthwise separable convolution [36], [37]. Although the depthwise

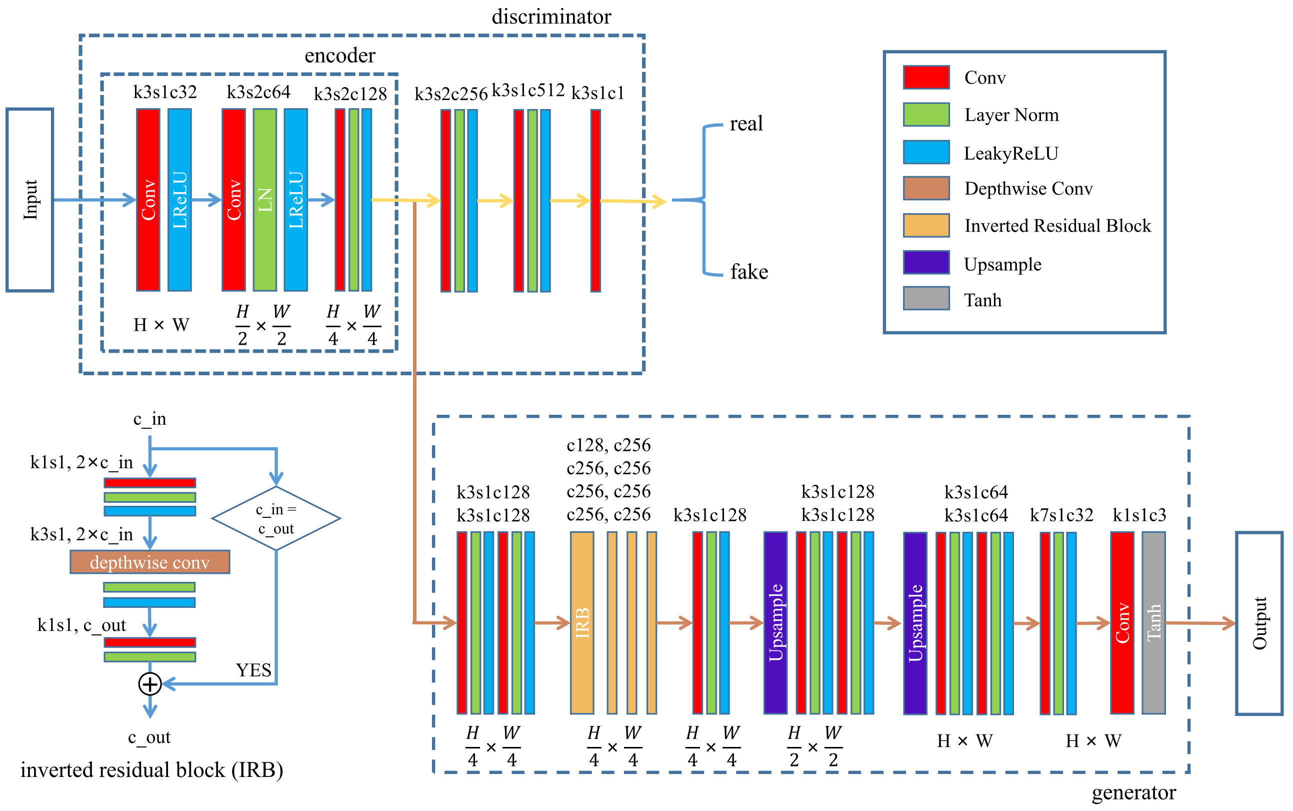
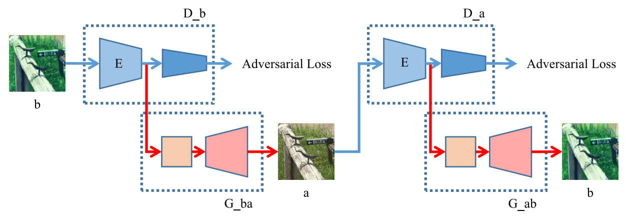
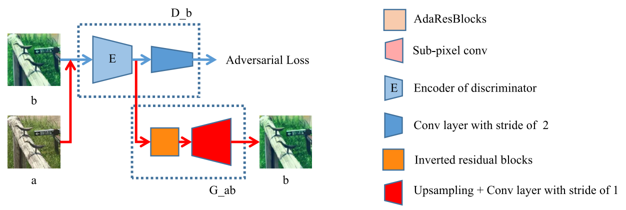


Fig. 2. The network architecture of CartoonLossGAN. In the figure, k is the kernel size, s is the stride, c is the number of channels, H and W represent the height and width of the input image. Upsampling doubles the image size through bilinear interpolation. The plus sign in the inverted residual block means elementwise sum.



(a) NICE-GAN



(b) Our CartoonLossGAN

Fig. 3. Difference between NICE-GAN and our proposed CartoonLossGAN. NICE-GAN uses two generators and two discriminators, and thus the resulting translation framework maybe not compact enough. We only use one generator and one discriminator.

separable convolution is powerful and the number of our network parameters can be significantly reduced by using the depthwise separable convolution. But in our experiment, the depthwise separable convolution will cause poor visual effects. AnimeGANv2[1] also uses the common convolutional layer instead of depthwise separable convolution. In addition, we also try to reduce the number of network parameters by reducing the number of channels, but a fuzzy cartoon image is generated.

Note that we use spectral normalization [38] in each convolutional layer of the discriminator network (including the convolutional layer of the encoder) to make the training of the discriminator more stable. Finally, we use resize-convolution layers [39] instead of fractionally strided convolutions [40] to prevent the cartoon image generated by our model from appearing checkerboard artifacts [39].

B. Cartoon Loss Function

To automatically generate cartoon-style images, we propose a novel loss function for our proposed network architecture. It can be used to imitate the process of sketching to learn the smooth surface of the cartoon image, and imitate the coloring process to learn the coloring of the cartoon images. So we call it cartoon loss function and describe it as follows.

1) Adversarial Loss: The adversarial loss $\mathcal{L}_{adv}$ [29] is mainly used to make the generator network generate cartoonized images through minimax training. We use the edge-promoting adversarial loss [1] proposed by CartoonGAN to generate clear edges. For real-world photo $p_i \in P$, we first extract features through encoder E, and then generate cartoon-style image through generator G (i.e., the generated cartoon image $\hat{c} = G(E(p_i))$). The adversarial loss function

[1]https://tachibanayoshino.github.io/AnimeGANv2/

is as follows:

$$\mathcal{L}_{adv}(G, D) = \mathbb{E}_{c_j \sim S_{data}(c)}\left[\log D\left(c_j\right)\right] + \mathbb{E}_{p_i \sim S_{data}(p)}\left[\log\left(1 - D\left(G\left(E\left(p_i\right)\right)\right)\right)\right] + \omega \mathbb{E}_{e_k \sim S_{data}(e)}\left[\log\left(1 - D\left(e_k\right)\right)\right], \quad (1)$$

where e_k is a smooth cartoon image obtained by smoothing the edges of the cartoon image $c_j \in C$, and $\omega = 0.1$. It should be noted that the encoder E only updates the parameters when training the discriminator D, and is fixed when training the generator G.

2) Content Loss: The content loss $\mathcal{L}_{content}$ [1] extracts high-level features from the real-world photo $p_i \in P$ and the generated cartoon image $G(E(p_i))$ through a pre-trained VGG19[2] network, and then compares the extracted high-level features to constrain the semantic content of the generated image. The content loss is as follows:

$$\mathcal{L}_{content} = \mathbb{E}_{p_i \sim S_{data}(p)}\left[\| VGG_l\left(G\left(E\left(p_i\right)\right)\right) - VGG_l\left(p_i\right) \|_1\right], \quad (2)$$

where l represents the lth layer of the pre-trained VGG19 network. Like CartoonGAN, we use the layer 'conv4_4'. Note that the encoder E is fixed here.

3) Gray Style Loss: Our proposed gray style loss $\mathcal{L}_{gray}$ is used to imitate the process of sketching to learn the smooth surface of the cartoon images. Firstly, we transform the cartoon image $c_j \in C$ and the generated image $G(E(p_i))$ into 3-channel grayscale images and then use the pre-trained VGG19 network to extract high-level features from the transformed grayscale images. Finally, we calculate the Gram matrix of the extracted features and compare them. The gray style loss is as follows:

$$\mathcal{L}_{gray} = \mathbb{E}\left[\| Gram\left(VGG_l\left(gray\_g\right)\right) - Gram\left(VGG_l\left(gray\_c\right)\right) \|_1\right], \quad (3)$$

where $gray\_g = Gray(G(E(p_i)))$, $gray\_c = Gray\left(c_j\right)$, l represents the lth layer of the pre-trained VGG19 network, we use the layer 'conv4_4'. Note that the encoder E is fixed here. The difference between our gray style loss and AnimeGAN's grayscale style loss [22] is that we transform the generated color images into grayscale images, and use grayscale generated images to compare with grayscale cartoon images. And AnimeGAN's grayscale style loss uses color generated images to compare with grayscale cartoon images. We do this to avoid the influence of colors and only learn the smooth surface of cartoon images.

4) Color Style Loss: Our proposed color style loss $\mathcal{L}_{color}$ is used to imitate the coloring process to learn the coloring of cartoon-style images. First, we use cartoon image $c_j \in C$ to generate image $G\left(E\left(c_j\right)\right)$. Then, we transform the cartoon image c_j and the generated image $G\left(E\left(c_j\right)\right)$ from RGB format to YUV format. Finally, we compare the transformed YUV images, where the Y channel uses the L_1 loss, and the U and V channels use the Huber loss. The Y channel represents luminance, and the image of the Y channel looks like an uncolored sketch. We use the L_1 loss to constrain the Y channel to help our model learn other features of cartoon images except for the color. The U and V channels represent chrominance. We use Huber loss to constrain the U and V channels to help our model learn the coloring of cartoon images. When the color difference between the input cartoon image and the generated cartoon image is small (i.e., when the absolute error between the U channel and V channel of the input cartoon image and the generated cartoon image is less than the Huber loss threshold), the Huber loss will become very small. At this time, the color style loss pays more attention to the constraint on the Y channel. In other cases, Huber loss is equivalent to the L_1 loss. The color style loss is as follows:

$$\mathcal{L}_{color} = \mathbb{E}_{c_j \sim S_{data}(c)}\left[\| Y\left(c\_g\right) - Y\left(c_j\right) \|_1 + \| U\left(c\_g\right) - U\left(c_j\right) \|_H + \| V\left(c\_g\right) - V\left(c_j\right) \|_H\right], \quad (4)$$

where $c\_g = G\left(E\left(c_j\right)\right)$. Note that the encoder E is fixed here. The difference between our color style loss and AnimeGAN's color reconstruction loss [22] is that we use cartoon images to generate cartoon images, and compare the generated cartoon images with the input cartoon images. And AnimeGAN's color reconstruction loss uses real-world photos to generate cartoon images and compares the generated cartoon images with the real-world photos. Through our color style loss, images with colors more similar to the input cartoon images can be generated.

5) Total Variation Regularizer: Total variation regularizer $\mathcal{L}_{tv}$ [26] is used to generate smooth cartoon-style images and reduces high-frequency noises. We refer to the implementation of $\mathcal{L}_{tv}$ in PaddleGAN.[3]

In summary, our proposed cartoon loss function $\mathcal{L}_{CartoonLoss}$ is as follows:

$$\mathcal{L}_{CartoonLoss} = \lambda_1 \mathcal{L}_{adv}(G, D) + \lambda_2 \mathcal{L}_{content} + \lambda_3 \mathcal{L}_{gray} + \lambda_4 \mathcal{L}_{color} + \lambda_5 \mathcal{L}_{tv}. \quad (5)$$

Note that we are motivated by the cartoon creation process of sketching first and then coloring, and thus divide the learning of cartoon style into two parts: learning the smooth surface of cartoon style images and learning the coloring of cartoon style images. Specifically, our proposed gray style loss adopts the idea of Gatys et al [3]. By comparing the grayscale image of the generated cartoon style image with the grayscale style image of the input cartoon style image, we can imitate the sketching process in the cartoon creation process. We compare grayscale images to prevent the interference of color information, which is also very similar to the sketching process. Through our proposed gray style loss $\mathcal{L}_{gray}$, our CartoonLossGAN can generate smooth surfaces of cartoon images. Our proposed color style loss is essentially L_1 loss. In order to focus on learning the coloring of cartoon style images, we transform the input cartoon style image and the cartoon style image generated by the input cartoon image to YUV format because of the U and V channels mainly contains chrominance information (i.e., color information). Through

[2]https://paddlegan.bj.bcebos.com/models/vgg19_no_fc.npy

[3]https://github.com/PaddlePaddle/PaddleGAN

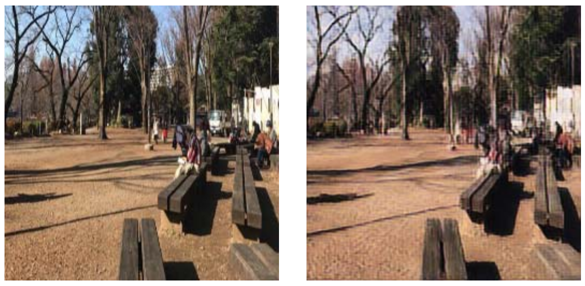

(a) Real-world photo (b) Initialization result

Fig. 4. Our initialization phase result.

our proposed color style loss $\mathcal{L}_{color}$, our CartoonLossGAN can generate bright colors that are more similar to cartoon images. The hyper-parameters we use in the total loss $\mathcal{L}_{CartoonLoss}$ will be introduced in Section IV.

C. Initialization Strategy

We propose an initialization strategy to be used in the case of reusing discriminator. Like CartoonGAN, the goal of our initialization strategy is to enable the generator to reconstruct the content of the input real-world image. Fig. 4 shows an example of our initialization strategy after training 2 epochs. The training process of GAN is not stable under the condition of random initialization because of its unique adversarial loss. The previous cartoonization training is to generate the semantic content of the input image and transform it into cartoon style. However, through our proposed initialization strategy, the subsequent cartoonization training is to directly transform the semantic content of the input image into cartoon style. That is, the task is simplified. It is due to that the generator network can generate the input content image after the training of the initialization strategy. So, based on the condition that the input content image can be generated, the resulting stylizing training process is stable than the process that the generator network first generates an image completely randomly and then stylizes it. The initial adversarial loss is as follows:

$$\mathcal{L}_{adv\_i} = \mathbb{E}_{p_i \sim S_{data}(p)}\left[\log D\left(p_i\right)\right] + \mathbb{E}_{p_i \sim S_{data}(p)}\left[\log\left(1 - D\left(G\left(E\left(p_i\right)\right)\right)\right)\right]. \quad (6)$$

Specifically, we train the discriminator D (including the encoder E) through our proposed initial adversarial loss $\mathcal{L}_{adv\_i}$, and train the generator G only through the content loss $\mathcal{L}_{content}$ (encoder E is fixed at this time).

Note that through our proposed cartoon loss function, our CartoonLossGAN can imitate the process of sketching to learn the smooth surface of the cartoon image and imitate the coloring process to learn the coloring of the cartoon image. Our proposed initialization strategy can make our model training easier and more stable.

IV. Experiments

In this section, we test our proposed CartoonLossGAN and compare it with four representative methods on the cartoon dataset to demonstrate its effectiveness. We implement our CartoonLossGAN through PaddlePaddle [41] on Baidu's AI Studio platform. In the process of implementation, we refer to the code of PaddleGAN[3]. We train 2 epochs in our initialization strategy, and then we train another 30 epochs for learning cartoonization. We use the Adam [42] optimizer to optimize the total loss function $\mathcal{L}_{total}$. We set the batch size to 4 and the learning rate to 0.0002. Our model is trained on an NVIDIA Tesla V100 GPU.

The hyper-parameters used in Eq. (5) are: $\lambda_1 = 300$, $\lambda_2 = 1.5$, $\lambda_3 = 2.5$, $\lambda_4 = 100$, $\lambda_5 = 1$. The reasons are as follows. When the weights of all loss functions are set as 1, the value of the adversarial loss and the color style loss is much smaller than the content loss and the gray style loss, and the adversarial loss and the color style loss play a crucial role in the cartoon style transfer. To balance their loss function values and make them all play a role in the total loss function, we set much larger weights for the adversarial loss and the color style loss than the content loss and the gray style loss. In addition, the total variation regularizer loss is a loss often used in artistic style transfer to generate smooth artistic style images. But the cartoon style transfer is mainly done by the other loss functions other than the total variation regularizer loss, so we set a small weight for the total variation regularizer loss to prevent it from causing interference to other loss functions.

A. Data

We are using the cartoon dataset[4] provided by AnimeGAN. Among them, the training data includes real-world photos and cartoon images, while the test data only has real-world photos, and all training data is processed as 256 × 256.

1) Real-World Photos: In our experiment, we use 6656 real-world photos for training and 790 images for testing, and they are all processed as 256 × 256. In addition, we also use 45 high-resolution images to show the cartoonized results of our model.

2) Cartoon Images: Cartoon images include color cartoon images, grayscale cartoon images, and smooth cartoon images. Cartoon images are used for training only. First, we process all cartoon images into 256 × 256. Then, we smooth the edges of the cartoon images (this processing can refer to the implementation of CartoonGAN[5]). In addition, the process of transforming color cartoon images to grayscale cartoon images is performed when reading the data, but it is a better choice to process the data in advance.

We select four cartoon styles for training. All cartoon images are obtained by extracting keyframes from cartoon movies (please choose high-definition video for extraction as much as possible). Among them, 1792 Miyazaki Hayao style images are extracted from the movie ***The Wind Rise***, 1650 Makoto Shinkai style cartoon images are extracted from the movie ***Your Name***, 1357 Mamoru Hosoda style cartoon images are extracted from the movie ***Summer Wars***, and

[4]https://github.com/TachibanaYoshino/AnimeGAN/releases/download/dataset-1/dataset.zip

[5]https://github.com/mnicnc404/CartoonGan-tensorflow/blob/master/scripts/

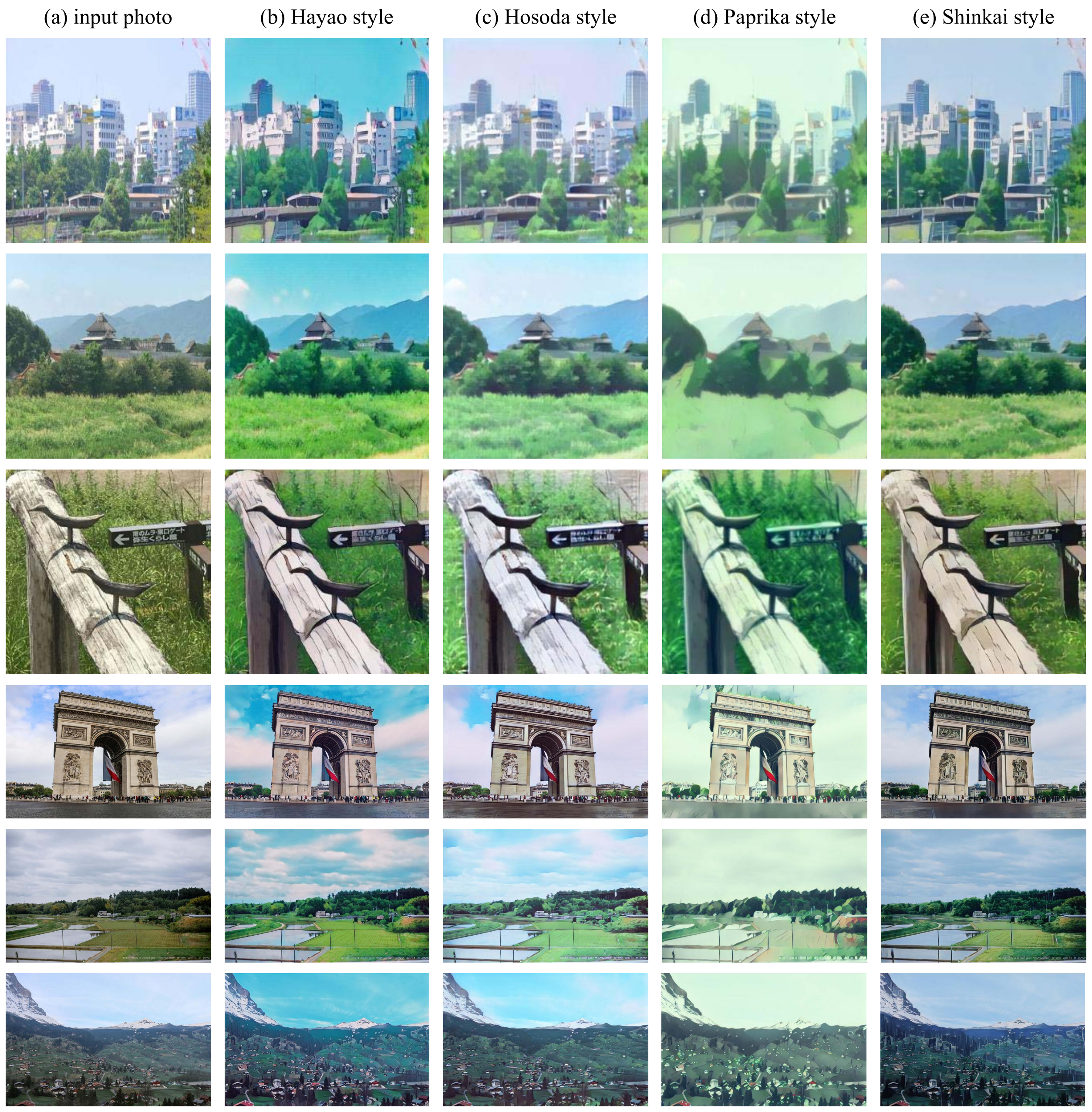


Fig. 5. Some results generated by our proposed CartoonLossGAN. (a) Input real-world photos. (b) Miyazaki Hayao style. (c) Mamoru Hosoda style. (d) Paprika style. (e) Makoto Shinkai style.

1553 Paprika style cartoon images are extracted from the movie ***Paprika***.

B. Performance Evaluation and Ablation Experiment

1) Performance Evaluation: We first apply our CartoonLossGAN on the cartoon dataset to evaluate its cartoonization performance. Fig. 5 shows part of the experimental results generated by our model. It can be seen that our model can generate vivid cartoon-style images. Through our proposed gray style loss, our CartoonLossGAN can generate a smooth surface of cartoon images without obvious distortion. Through our proposed color style loss, our CartoonLossGAN can generate bright colors that are more similar to cartoon images, without producing dark images. Through reusing the encoder part of the discriminator, our CartoonLossGAN can extract image features efficiently and avoid the waste of the discriminator. Our proposed initialization strategy can make our model training easier and more stable.

2) Ablation Experiment: We reuse the discriminator mainly to build a compact generative adversarial network based cartoonization architecture and avoid the waste of the discriminator, but the reuse of the discriminator enables our CartoonLossGAN to generate better cartoon style images. As shown in Fig. 6(b), when the discriminator is not reused, some strange vertical lines are generated in the lawn part of the image. As shown in Fig. 6(f), in the case of reusing the

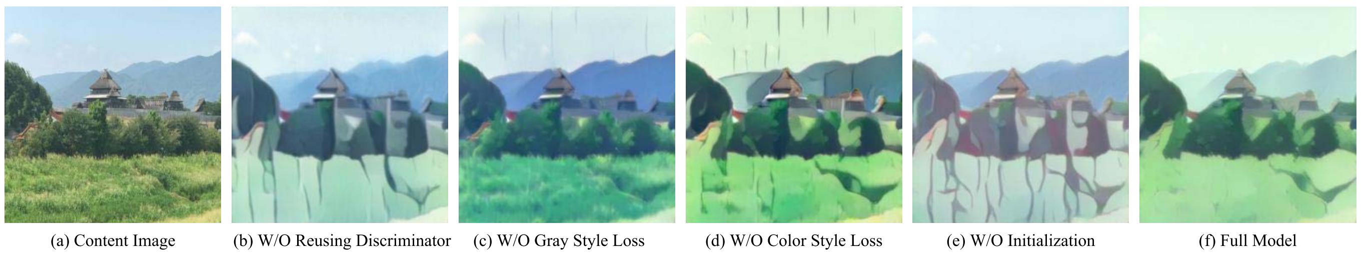


Fig. 6. Ablation experiment by removing each component.

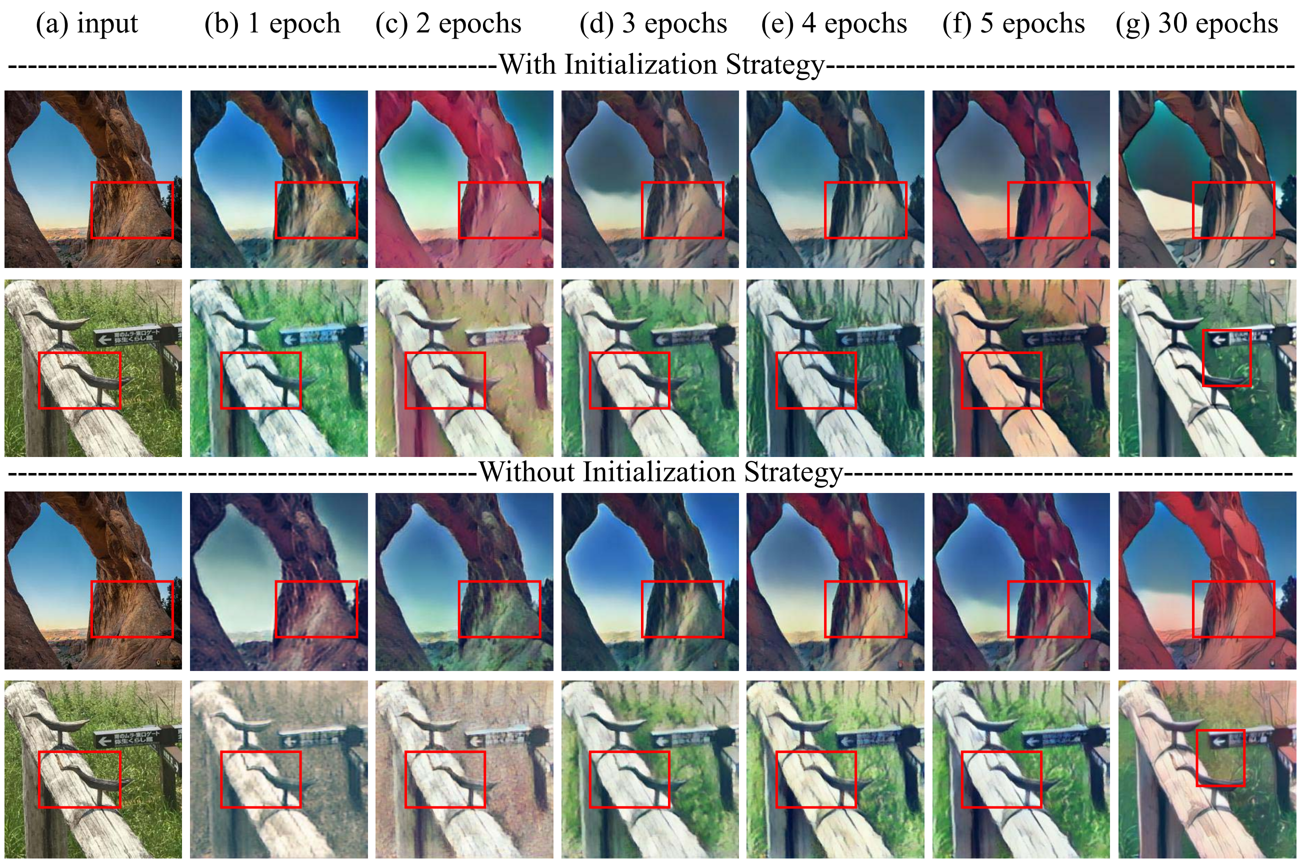


Fig. 7. Some ablation experiment results of our proposed initialization strategy. The upper part indicates that the initialization strategy is used, and the lower part indicates that the initialization strategy is not used. (a) Input real-world photos; (b)-(f) The experimental results of the first 5 epochs; (g) Final experimental results.

discriminator, our model generates a complete and smooth lawn. The discriminator is used to determine whether the input cartoon image is a cartoon image generated by the generator or a real cartoon image. By training the discriminator, the encoder of the discriminator can extract more important information for distinguishing cartoon images. We believe that such an encoder helps us generate better cartoon style images. Next, let's compare Fig. 6(c) and Fig. 6(d). As shown in Fig. 6(d), the lawn part marked by the red box generates a smoother surface under the effect of the gray style loss. As shown in Fig. 6(c), when the gray style loss is not used, although the lawn part marked by the red box is not as smooth as the cartoon style image, a color closer to the cartoon style image is generated under the effect of the color style loss. As shown in Fig. 6(e), a lot of vertical lines are generated in the lawn part without using the initialization strategy. We think this is because the semantic content of the input content image is not well preserved. Through our proposed initialization strategy, the generator can perform cartoon style transfer training when it can generate the input content image, which helps to preserve the semantic content of the input content image. As shown in Fig. 6(f), when the initialization strategy is used, the semantic content of the input content image is well preserved, and we can easily see that the part marked by the red box is a lawn.

To further show the effectiveness of our proposed initialization strategy and cartoon loss function, we perform more ablation experiments as follows.

Our proposed initialization strategy can make our model easier to cartoonize training. We next show the ablation experimental results of the first 5 epochs to illustrate it. As shown in Fig. 7, the upper part indicates that the initialization strategy is used, and the lower part indicates that the initialization strategy is not used. Please note the changes in the red box. In the

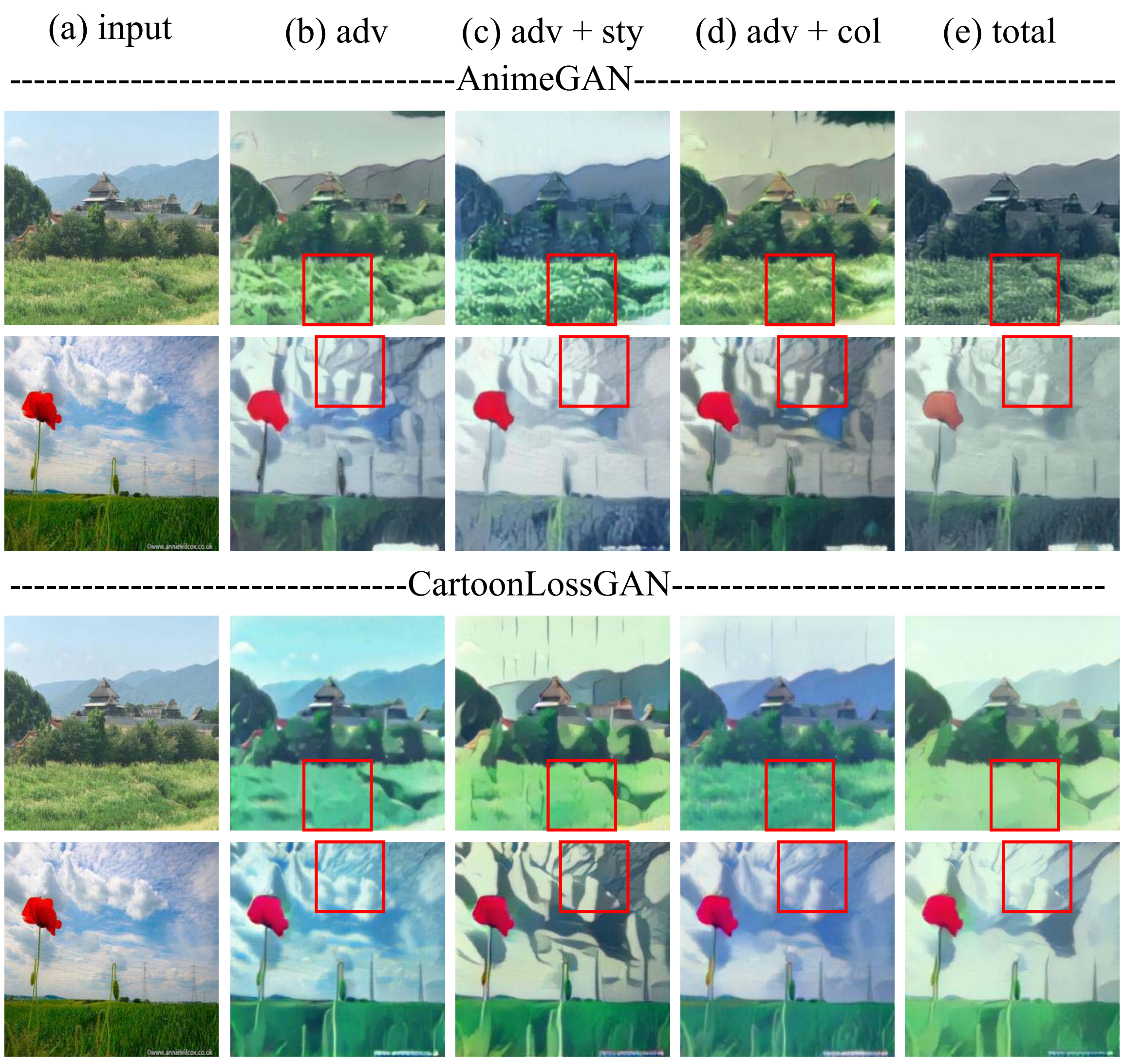


Fig. 8. Ablation experiment and comparisons of the loss functions in AnimeGAN and our CartoonLossGAN. (a) Input content image; (b) The adversarial loss, where AnimeGAN uses their proposed grayscale adversarial loss, and we use the edge-promoting adversarial loss proposed by CartoonGAN; (c) The adversarial loss and the style loss, where AnimeGAN uses the grayscale style loss they proposed, and we use the improved gray style loss; (d) The adversarial loss and the color loss, where AnimeGAN uses the color reconstruction loss they proposed, and we use the improved color style loss; (e) The total loss.

case of using the initialization strategy, our model quickly produces a cartoonized effect and produces a relatively good cartoonized result in the third epoch (i.e., a smooth surface and sharp edges are produced). This is because after the initialization strategy, the generator network has been able to generate input content images, and subsequent training only needs to be cartoonized training. In the case of not using the initialization strategy, the first 3 epochs seem to be struggling to generate the semantic content of the input image, and it is not until the 5th epoch that a good cartoonized result is produced. In addition, as shown in Fig. 7(g), our model generates a clearer cartoon image with clear edge lines and smooth surfaces when using the initialization strategy. We can easily observe the above viewpoints in the rock parts marked by the red box. In short, our proposed initialization strategy can help our model to generate better cartoon images more easily.

To further demonstrate the superiority of our proposed cartoon loss function, we perform ablation experiment and comparisons with the loss function in AnimeGAN. Fig. 8 shows the ablation experiment and comparisons of the loss functions in AnimeGAN and our CartoonLossGAN. As shown in Fig. 8(b) and Fig. 8(c), through our proposed gray style loss, our model generates smooth surfaces and sharp edges of cartoon style images. The above point of view can be easily seen in the part of the cloud marked in the red box. As shown in Fig. 8(b) and Fig. 8(d), through our proposed color style loss, our model adjusts the color of the generated cartoon image to make the color of the generated cartoon image closer to the color of the reference cartoon style. As shown in Fig. 8(b) and Fig. 8(e), after using our proposed gray style loss and color style loss, the cartoon image generated by our model generates a smoother surface and clear edges. The color of the generated cartoon image has been adjusted to make it closer to the reference cartoon style coloring.

Furthermore, we perform a detailed comparison to verify the superiority of our proposed cartoon loss over the loss function used in AnimeGAN although the two loss functions are similar. It can be clearly seen from Fig. 8 that the color of AnimeGAN's results is dark. We think this is due to their mixed use of color images and grayscale images for comparison in their grayscale adversarial loss and grayscale style loss. This phenomenon can be seen in Fig. 8(b) and Fig. 8(c). They try to eliminate this phenomenon through their color reconstruction loss, but from Fig. 8(d), the effect is not satisfactory. To solve the above problems, we make a very simple and effective rule, i.e., color images are only compared with color images, and grayscale images are only compared with grayscale images, to prevent incorrect color mapping. From the comparative experimental results of Fig. 8(b) - Fig. 8(d), it is obvious that the color of the cartoon image generated by our model is more vivid and bright under the combination of various loss functions.

C. Qualitative Comparison

To further show the effectiveness of our proposed method, we first compare it with the method of Gatys *et al.* [3] and CycleGAN [9]. These two methods are typical methods for traditional artistic style transfer (e.g., oil painting), and represent the neural style transfer (NST) [25] method and the GAN-based method, respectively. The method of Gatys *et al.* [3] is an online image optimization method. Given an input image and a reference cartoon-style image, their method uses an iterative method to continuously optimize a noise image until the noise image has the content of the input image and the style of the reference image. In our experiment, we use the PyTorch version of the NST code[6] implemented by others. CycleGAN is a typical unsupervised image-to-image translation method. In our experiment, we use the code[7] provided by the original author, train by default, and only use the generator $G : P \rightarrow C$ that generates cartoon images for the comparative experiment.

Then, we compare with two state-of-the-art cartoonization methods (i.e., CartoonGAN [1] and AnimeGAN [22]). We use the pre-trained model[8] provided by the original author of CartoonGAN to perform the comparative experiment. Our experiment and CartoonGAN use the same four cartoon styles, but since Hayao style training data are extracted from different movies, we only select Shinkai style and Paprika style for comparison. Since AnimeGAN does not provide a pre-trained model, we use the code[9] provided by the authors of AnimeGAN for training through default configuration.

We give comparative experiments with four representative methods to demonstrate the superiority of our proposed CartoonLossGAN. The experimental results are shown in Fig. 9. Obviously, the mothed of Gatys *et al.* [3] cannot generate satisfactory cartoon-style images. The method of Gatys *et al.* [3] requires special care in selecting reference cartoon-style images, which must be highly consistent with the semantic content of the input content image. When the reference style image we choose is not highly correlated with the input content image, it will suffer obvious failure. CycleGAN [9] can generate impressive cartoon images on the whole, but the cartoon results of CycleGAN have obvious distortions in local details and produce artifacts that do not exist in the input content image (we annotate these distortions and artifacts by using the red box in Fig. 9(c)).

CartoonGAN [1] is a powerful cartoonization model. However, as shown in Fig. 9(d), CartoonGAN produces slight distortion in local details, and also produces disharmonious artifacts. AnimeGAN [22] is the state-of-the-art method for cartoonization, which can generate realistic cartoon images. However, the brightness of the cartoon images generated by AnimeGAN is generally dark, and there are blurs and artifacts in local areas. Compare with the above methods, our proposed CartoonLossGAN generates vivid cartoon images with bright colors and smooth surfaces and achieves the best visual effects.

TABLE I
FID Scores Calculated Under Shinkai Style

Model	FID to Cartoon Images	FID to Content Images
Content Images	135.82	N/A
CycleGAN [9]	**108.29**	91.95
CartoonGAN [1]	117.62	58.71
AnimeGAN [22]	116.33	77.36
CartoonLossGAN	118.79	**53.95**

D. Quantitative Evaluation

How to evaluate the results of artistic style transfer is a difficult problem, because **people's evaluation of the results of artistic style transfer is very subjective and difficult to measure with specific metrics**. In our experiments, we use Frechet Inception Distance (FID) [43] to evaluate our model. FID extracts image features through a pre-trained Inception-V3 model [44], and calculates the distance between two image distributions through the extracted image features, so as to measure the similarity of the two image distributions. The lower the FID score, it means that the distribution of the generated image is closer to the distribution of the reference real image. In other words, the generated image is more like the real image. In our experiment, we use the PyTorch implementation[10] recommended by the original author of FID.

We calculate FID scores for cartoon images and content images respectively. Although we and CartoonGAN [1] use the same four styles, since the images of the two styles are extracted from different movies, we only compare the FID scores calculated under the Shinkai style and the Paprika Style. We calculate the FID score between the generated images and the cartoon images to determine whether the generated images have the style of the cartoon images. In addition, we also calculate the FID score between the generated images and the content images to determine whether the generated images well retain the semantic content of the content images.

We give quantitative evaluations with four representative methods to prove the superiority of our method. As shown in Table I and Table II, the FID score between the cartoon images

[6]https://github.com/gordicaleksa/pytorch-neural-style-transfer
[7]https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix
[8]https://github.com/FlyingGoblin/CartoonGAN
[9]https://github.com/TachibanaYoshino/AnimeGAN
[10]https://github.com/mseitzer/pytorch-fid

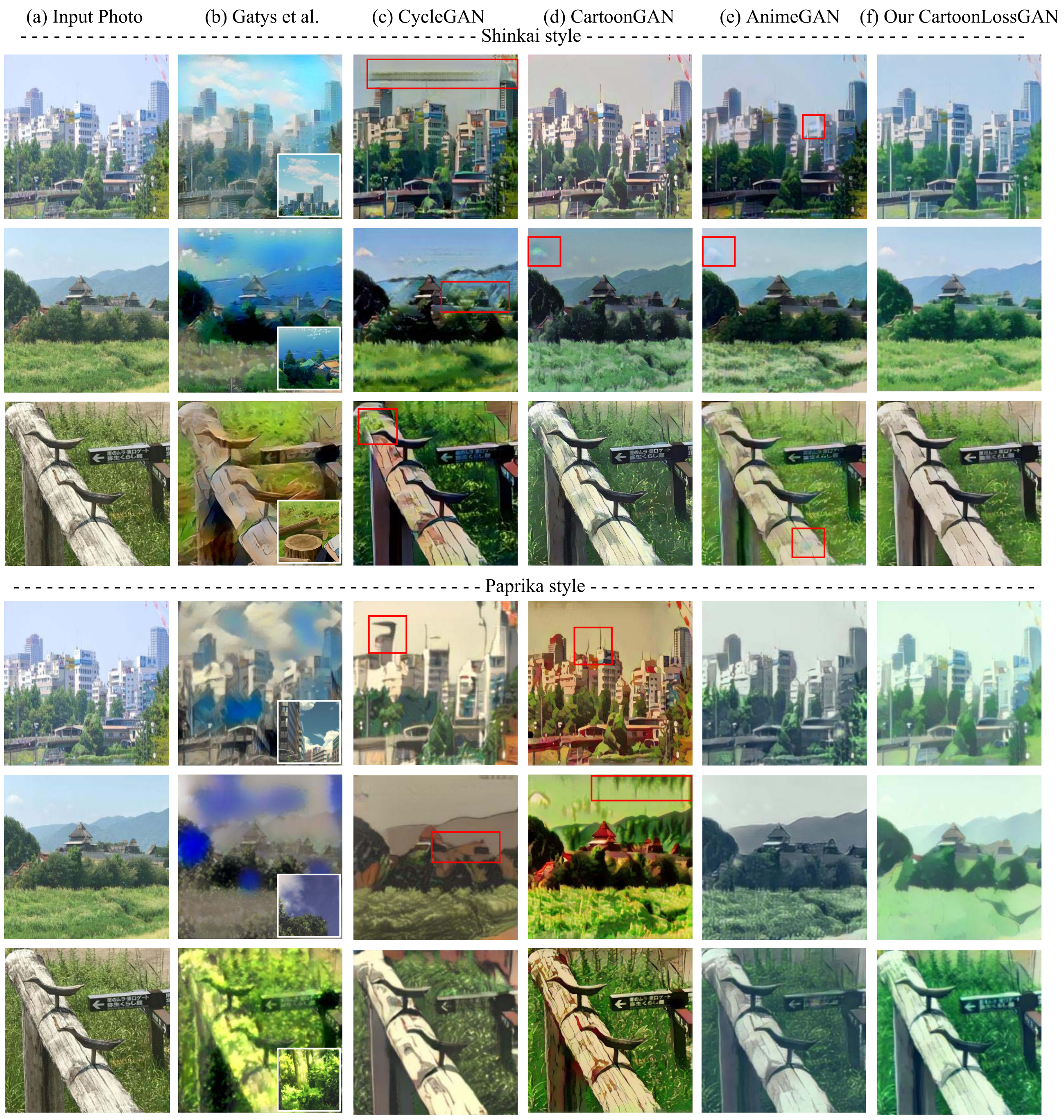


Fig. 9. Qualitative comparison with other methods. We use red boxes to mark obvious distortions and artifacts. We give the reference cartoon image in the lower right corner of the result of Gatys *et al.* [3].

generated by CycleGAN [9] and the real cartoon images is the lowest, and it is significantly lower than other methods. But as shown in Fig. 10, there are very obvious distortions and artifacts in the results of CycleGAN. We assume that this is due to the excessive cartoonization of the cartoon images generated by CycleGAN. The FID score between the cartoon images generated by CycleGAN and the real content images is significantly higher than other methods, which also proves our point that the results of CycleGAN do not well preserve the semantic content of the content images. Compare with CartoonGAN [1] and AnimeGAN [22], our proposed CartoonLossGAN has little difference in FID scores on cartoon images. But the FID score of our proposed CartoonLossGAN on the content images is significantly better than CartoonGAN and AnimeGAN, which shows that our proposed CartoonLossGAN better preserves the semantic content of the content images.

To further show the effectiveness of our proposed CartoonLossGAN, we perform user study experiments. The experimental setting is similar to RPD-GAN [19]. We number

TABLE II
FID SCORES CALCULATED UNDER PAPRIKA STYLE

Model	FID to Cartoon Images	FID to Content Images
Content Images	174.27	N/A
CycleGAN [9]	**122.60**	141.39
CartoonGAN [1]	143.98	107.81
AnimeGAN [22]	150.71	95.67
CartoonLossGAN	147.88	**79.95**

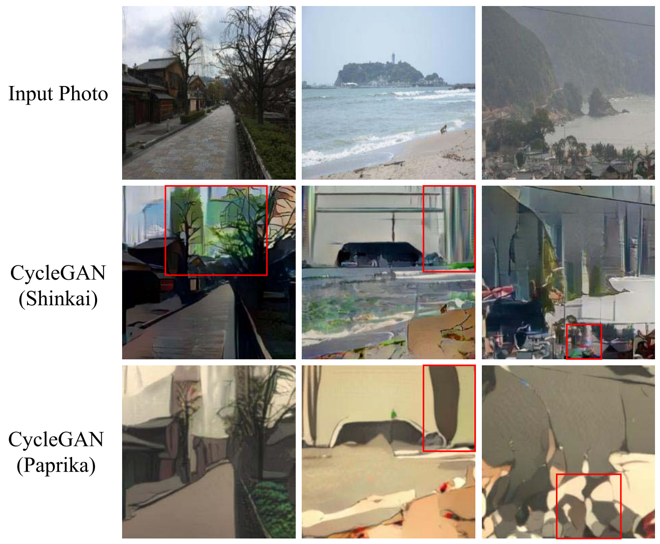


Fig. 10. Some cartoon-style images generated by CycleGAN. Many obvious distortions and artifacts can be seen in these generated cartoon-style images.

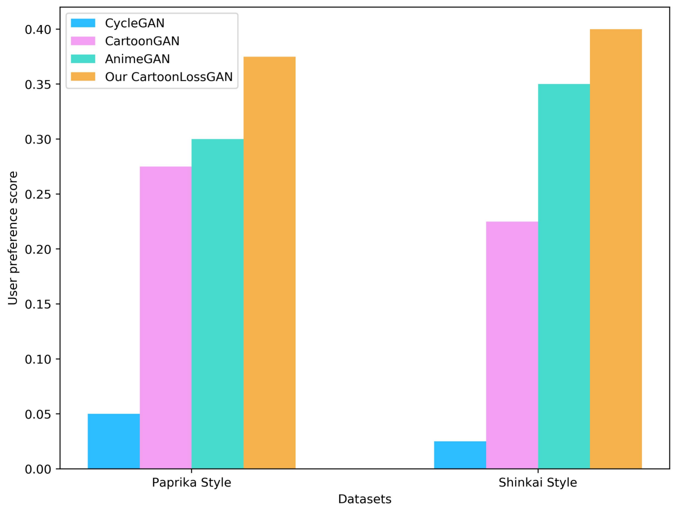


Fig. 11. User preference scores for cartoon style images generated by CycleGAN, CartoonGAN, AnimeGAN and our proposed CartoonLossGAN.

the input content images, and transform the input content images into cartoon style images by using the four models including CycleGAN [9], CartoonGAN [1], AnimeGAN [22], and our proposed CartoonLossGAN. For the sake of fairness, we number the generated cartoon style images, that is, the users participating in user study do not know by which model each cartoon style image is generated. We do not compare Gatys *et al.* [3], because for each input content image, Gatys *et al.* needs to find a reference cartoon style image very similar to it. We randomly select a set of images for different cartoon styles, and then let users evaluate the randomly selected set of images (i.e., select the cartoon style image they think is the best). We count the user evaluation data and calculate the ratio of the number of times each model is selected to the total number. The ratio is used as the user preference score. Fig. 11 shows the user preference scores for cartoon style images generated by the four different models. It can be seen from Fig. 11 that our CartoonLossGAN achieved the highest scores on different datasets. In addition, CycleGAN and CartoonGAN also generate impressive cartoon style images. However, the cartoon style images generated by CycleGAN and CartoonGAN are obviously distorted, which affects the user's choice for them.

V. CONCLUSION

In this paper, we propose CartoonLossGAN, a generative adversarial network framework, which generates vivid cartoon images by learning surface and coloring of images to imitate the cartoon creation process of sketching first and then coloring. The proposed cartoon loss function can imitate the process of sketching to learn the smooth surface of the cartoon image, and imitate the coloring process to learn the coloring of the cartoon image. In addition, we reuse the encoder part of the discriminator without using a cyclic manner and a complex multi-scale discriminator to build a compact generative adversarial network (GAN) based cartoonization architecture. Furthermore, we also propose an initialization strategy, which is used in the scenario of reusing the discriminator to make our model training easier and more stable. Extensive experimental results demonstrate that our proposed CartoonLossGAN can generate fantastic cartoon-style images, and outperforms four representative methods. The ablation experiment demonstrates the effectiveness and rationality of the proposed cartoon loss function. In the future, we will consider to solve the problem that cartoonization is not obvious. Moreover, we also hope to use the fusion network [45] and feature selection methods [46] to fuse the InfoGAN's [47] ideas for achieving multi-style transfer.

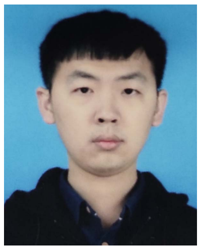
REFERENCES

[1] Y. Chen, Y.-K. Lai, and Y.-J. Liu, "CartoonGAN: Generative adversarial networks for photo cartoonization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 9465–9474.
[2] Y. Jing *et al.*, "Stroke controllable fast style transfer with adaptive receptive fields," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 238–254.
[3] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2414–2423.
[4] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Represent.*, 2015, pp. 1–14.
[5] D. Bhattacharjee, S. Kim, G. Vizier, and M. Salzmann, "DUNIT: Detection-based unsupervised image-to-image translation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 4786–4795.
[6] H. Xu, J. Ma, and X.-P. Zhang, "MEF-GAN: Multi-exposure image fusion via generative adversarial networks," *IEEE Trans. Image Process.*, vol. 29, pp. 7203–7216, 2020.

[7] A. Brock, J. Donahue, and K. Simonyan, “Large scale GAN training for high fidelity natural image synthesis,” in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–35.
[8] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, “Image-to-image translation with conditional adversarial networks,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5967–5976.
[9] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, “Unpaired image-to-image translation using cycle-consistent adversarial networks,” in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2242–2251.
[10] Y. Wang, Z. Zhang, W. Hao, and C. Song, “Multi-domain image-to-image translation via a unified circular framework,” *IEEE Trans. Image Process.*, vol. 30, pp. 670–684, 2021.
[11] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo, “StarGAN: Unified generative adversarial networks for multi-domain image-to-image translation,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8789–8797.
[12] Y. Choi, Y. Uh, J. Yoo, and J.-W. Ha, “StarGAN V2: Diverse image synthesis for multiple domains,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 8185–8194.
[13] Y. Li, S. Tang, R. Zhang, Y. Zhang, J. Li, and S. Yan, “Asymmetric GAN for unpaired image-to-image translation,” *IEEE Trans. Image Process.*, vol. 28, no. 12, pp. 5881–5896, Jun. 2019.
[14] R. Li *et al.*, “SDP-GAN: Saliency detail preservation generative adversarial networks for high perceptual quality style transfer,” *IEEE Trans. Image Process.*, vol. 30, pp. 374–385, 2021.
[15] F. Gao, X. Xu, J. Yu, M. Shang, X. Li, and D. Tao, “Complementary, heterogeneous and adversarial networks for image-to-image translation,” *IEEE Trans. Image Process.*, vol. 30, pp. 3487–3498, 2021.
[16] H. Tang, H. Liu, and N. Sebe, “Unified generative adversarial networks for controllable image-to-image translation,” *IEEE Trans. Image Process.*, vol. 29, pp. 8916–8929, 2020.
[17] L. Dai and J. Tang, “iFlowGAN: An invertible flow-based generative adversarial network for unsupervised image-to-image translation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, early access, Mar. 2, 2021, doi: 10.1109/TPAMI.2021.3062849.
[18] R. Chen, W. Huang, B. Huang, F. Sun, and B. Fang, “Reusing discriminators for encoding: Towards unsupervised image-to-image translation,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 8165–8174.
[19] X. Gao, Y. Tian, and Z. Qi, “RPD-GAN: Learning to draw realistic paintings with generative adversarial network,” *IEEE Trans. Image Process.*, vol. 29, pp. 8706–8720, 2020.
[20] R. Yi, Y.-J. Liu, Y.-K. Lai, and P. L. Rosin, “Unpaired portrait drawing generation via asymmetric cycle mapping,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 8214–8222.
[21] M. Pęśko, A. Svystun, P. Andruszkiewicz, P. Rokita, and T. Trzciński, “Comixify: Transform video into comics,” *Fundamenta Informaticae*, vol. 168, nos. 2–4, pp. 311–333, Sep. 2019.
[22] J. Chen, G. Liu, and X. Chen, “AnimeGAN: A novel lightweight GAN for photo animation,” in *Proc. Int. Symp. Intell. Comput. Appl.*, 2019, pp. 242–256.
[23] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin, “Image analogies,” in *Proc. 28th Annu. Conf. Comput. Graph. Interact. Techn.*, 2001, pp. 327–340.
[24] H. Winnemöller, S. C. Olsen, and B. Gooch, “Real-time video abstraction,” *ACM Trans. Graph.*, vol. 25, no. 3, pp. 1221–1226, 2006.
[25] Y. Jing, Y. Yang, Z. Feng, J. Ye, Y. Yu, and M. Song, “Neural style transfer: A review,” *IEEE Trans. Vis. Comput. Graphics*, vol. 26, no. 11, pp. 3365–3385, Nov. 2020.
[26] J. Johnson, A. Alahi, and L. Fei-Fei, “Perceptual losses for real-time style transfer and super-resolution,” in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 694–711.
[27] D. Ulyanov, A. Vedaldi, and V. Lempitsky, “Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4105–4113.
[28] H. Zhang and K. Dana, “Multi-style generative network for real-time transfer,” in *Proc. Eur. Conf. Comput. Vis. Workshops*, 2018, pp. 349–365.
[29] I. Goodfellow *et al.*, “Generative adversarial nets,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.
[30] M. Mirza and S. Osindero, “Conditional generative adversarial nets,” 2014, *arXiv:1411.1784*.
[31] A. Anoosheh, E. Agustsson, R. Timofte, and L. Van Gool, “ComboGAN: Unrestrained scalability for image domain translation,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2018, pp. 783–790.
[32] X. Li, K. Kou, and B. Zhao, “Weather GAN: Multi-domain weather translation using generative adversarial networks,” *CoRR*, vol. abs/2103.05422, pp. 1–10, Mar. 2021.
[33] J. Kim, M. Kim, H. Kang, and K. H. Lee, “U-GAT-IT: Unsupervised generative attentional networks with adaptive layer-instance normalization for image-to-image translation,” in *Proc. Int. Conf. Learn. Represent.*, 2020.
[34] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[35] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “MobileNetV2: Inverted residuals and linear bottlenecks,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520.
[36] A. G. Howard *et al.*, “MobileNets: Efficient convolutional neural networks for mobile vision applications,” 2017, *arXiv:1704.04861*.
[37] F. Chollet, “Xception: Deep learning with depthwise separable convolutions,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1251–1258.
[38] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, “Spectral normalization for generative adversarial networks,” in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–26.
[39] A. Odena, V. Dumoulin, and C. Olah, “Deconvolution and checkerboard artifacts,” *Distill*, vol. 1, no. 10, p. e3, Oct. 2016.
[40] V. Dumoulin and F. Visin, “A guide to convolution arithmetic for deep learning,” 2016, *arXiv:1603.07285*.
[41] Y. Ma, D. Yu, T. Wu, and H. Wang, “PaddlePaddle: An open-source deep learning platform from industrial practice,” *Frontiers Data Domputing*, vol. 1, no. 1, pp. 105–115, 2019.
[42] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *Proc. Int. Conf. Learn. Represent. (Poster)*, 2015, pp. 1–15.
[43] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, “GANs trained by a two time-scale update rule converge to a local nash equilibrium,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6629–6640.
[44] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2818–2826.
[45] X. Li, D. Song, and Y. Dong, “Hierarchical feature fusion network for salient object detection,” *IEEE Trans. Image Process.*, vol. 29, pp. 9165–9175, 2020.
[46] X. Li, H. Zhang, R. Zhang, and F. Nie, “Discriminative and uncorrelated feature selection with constrained spectral analysis in unsupervised learning,” *IEEE Trans. Image Process.*, vol. 29, pp. 2139–2149, 2020.
[47] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, “InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2180–2188.

Yongsheng Dong (Senior Member, IEEE) received the Ph.D. degree in applied mathematics from Peking University in 2012.

He was a Postdoctoral Research Fellow with the Center for Optical Imagery Analysis and Learning, Xi’an Institute of Optics and Precision Mechanics, Chinese Academy of Sciences, Xi’an, China, from 2013 to 2016. From 2016 to 2017, he was a Visiting Research Fellow at the School of Computer Science and Engineering, Nanyang Technological University, Singapore. His current research interests include pattern recognition, machine learning, and computer vision. He has authored or coauthored over 50 papers at famous journals and conferences, including IEEE TRANSACTIONS ON IMAGE PROCESSING, IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, IEEE TRANSACTIONS ON CYBERNETICS, IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, and IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING. He is a Senior Member of CCF. He is an Associate Editor of *Neurocomputing*.



Wei Tan received the B.Eng. degree in software engineering from the Henan University of Science and Technology in 2017, where he is currently pursuing the M.Sc. degree in computer science and technology. His research interests include computer vision and machine learning.

Dacheng Tao (Fellow, IEEE) is currently the Inaugural Director of the JD Explore Academy and a Senior Vice President of JD.com. He mainly applies statistics and mathematics to artificial intelligence and data science, and his research is detailed in one monograph and over 200 publications in prestigious journals and proceedings at leading conferences. He received the 2015 Australian Scopus-Eureka Prize, the 2018 IEEE ICDM Research Contributions Award, and the 2021 IEEE Computer Society McCluskey Technical Achievement Award. He is a fellow of the Australian Academy of Science, AAAS, and ACM.

Lintao Zheng received the Ph.D. degree in computer science from Zhejiang University in 2012. He is currently a Lecturer with the School of Information Engineering, Henan University of Science and Technology, Luoyang, China. His current research interests include image processing and computer vision.

Xuelong Li (Fellow, IEEE) is currently a Full Professor with the School of Artificial Intelligence, Optics and Electronics (iOPEN), Northwestern Polytechnical University, Xi’an, China.