GAN-based Multi-Style Photo Cartoonization

Yezhi Shu, Ran Yi, Mengfei Xia, Zipeng Ye, Wang Zhao, Yang Chen, Yu-Kun Lai, *Member, IEEE,* Yong-Jin Liu, *Senior Member, IEEE*

Abstract—Cartoon is a common form of art in our daily life and automatic generation of cartoon images from photos is highly desirable. However, state-of-the-art single-style methods can only generate one style of cartoon images from photos and existing multi-style image style transfer methods still struggle to produce high-quality cartoon images due to their highly simplified and abstract nature. In this paper, we propose a novel multi-style generative adversarial network (GAN) architecture, called MS-CartoonGAN, which can transform photos into multiple cartoon styles. MS-CartoonGAN uses only unpaired photos and cartoon images of multiple styles for training. To achieve this, we propose to use (1) a hierarchical semantic loss with sparse regularization to retain semantic content and recover flat shading in different abstract levels, (2) a new edge-promoting adversarial loss for producing fine edges, and (3) a style loss to enhance the difference between output cartoon styles and make training process more stable. We also develop a multi-domain architecture, where the generator consists of a shared encoder and multiple decoders for different cartoon styles, along with multiple discriminators for individual styles. By observing that cartoon images drawn by different artists have their unique styles while sharing some common characteristics, our shared network architecture exploits the common characteristics of cartoon styles, achieving better cartoonization and being more efficient than single-style cartoonization. We show that our multi-domain architecture can theoretically guarantee to output desired multiple cartoon styles. Through extensive experiments including a user study, we demonstrate the superiority of the proposed method, outperforming state-of-the-art single-style and multi-style image style transfer methods.

Index Terms—Style transfer, cartoon styles, multi-style transfer, generative adversarial network

1 Introduction

Cartoons are a popular artistic form in our daily life. In addition to artistic interests, their applications range from publication in printed media to storytelling for children's education. However, manually creating cartoon images is very laborious and involves substantial artistic skills. Therefore, automatic generation of cartoon images from photos is highly desirable.

Stylizing images in an artistic manner has been widely studied in the domain of non-photorealistic rendering [1]. Traditional approaches develop dedicated computational tools such as various types of brushes to simulate specific styles and thus require substantial efforts on a comprehensive understanding of the artist's painting skills. Recently, learning-based style transfer methods (e.g. [2]), in which an image can be stylized based on provided examples, have drawn considerable attention. In particular, the power of generative adversarial networks (GANs) [3] formulated in a cyclic manner is explored to achieve high-quality style transfer, with the distinct feature that the model is trained using unpaired photos and stylized images.

As summarized in Section 2, although significant success has been achieved with learning based stylization including both single-style (e.g., [2], [3], [4]) and multi-style (e.g., [5], [6]) methods, they often fail to produce cartoonized images with acceptable quality. There are two reasons. First, instead of adding textures such as brush strokes in many other styles, cartoon images are highly simplified and abstracted from real-world photos. Second, despite variation of styles among artists, cartoon images have noticeable common appearance, i.e., clear edges, smooth color shading and relatively simple textures, which is very different from other forms of artworks. See Figure 1 for an illustration.

In our previous conference paper [8], we propose a GAN-based deep network model called CartoonGAN for learning single-style cartoonization. CartoonGAN takes a set of photos and a set of cartoon images for training. To produce high quality results while making the training data easy to obtain, CartoonGAN uses a dedicated GAN-based architecture together with two simple yet effective loss functions, which does not require pairing or correspondence between two sets of images. However, CartoonGAN can only learn one specific cartoon style for an individual artist and multiple CartoonGAN models have to be trained separately for multiple cartoon styles, with each style requiring to have sufficient training data. In this paper, we substantially extend the CartoonGAN into a novel multi-style GAN model called MS-CartoonGAN. In this model, the generator contains (1) an encoder shared by all styles, (2) multiple decoders corresponding to different cartoon styles, along with multiple discriminators used to promote generation of different cartoon styles, and (3) an auxiliary classifier to distinguish different cartoon styles, with which a style loss is introduced to enhance the difference between output styles and make the training process more stable. In particular, MS-CartoonGAN makes the following contributions:

- MS-CartoonGAN is a multi-domain GAN-based model. To learn multi-style cartoonization, MS-CartoonGAN uses a single encoder shared by all

- *Y. Shu, R. Yi, M. Xia, Z. Ye, W. Zhao, Y.-J. Liu are with BNRist, MOE-Key Laboratory of Pervasive Computing, the Department of Computer Science and Technology, Tsinghua University, Beijing, China. Y. Shu and R. Yi are joint first authors. Y.-J. Liu is the corresponding author. E-mail: liuyongjin@tsinghua.edu.cn.*
- *Y. Chen is with Tencent, China.*
- *Y.-K. Lai is with School of Computer Science and Informatics, Cardiff University, UK.*

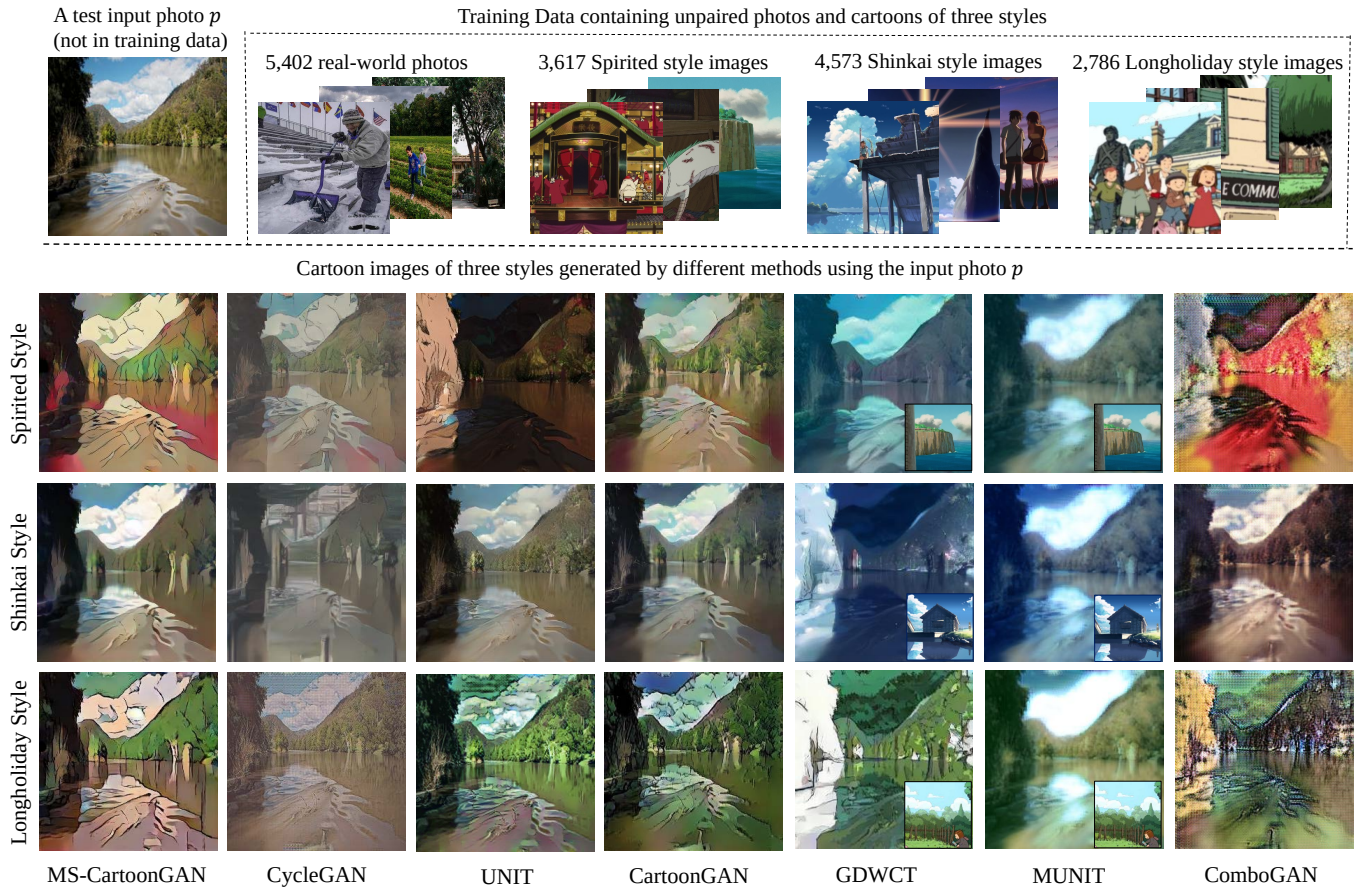


Fig. 1. We propose MS-CartoonGAN that can transform real-world photos into multiple cartoon styles with significant difference, using only unpaired photos and cartoon images of multiple styles for training. Although all training images are of size 256×256, MS-CartoonGAN can input photos of arbitrary size, thanks to the full convolutional network. MS-CartoonGAN generates better results than state of the arts, including CycleGAN, UNIT, CartoonGAN, GDWCT, MUNIT and ComboGAN. More comparisons with single-style methods (two versions of NST [2], CycleGAN [3] with identity loss, UNIT [4], GDWCT [7] and CartoonGAN [8])) and multi-style methods (MUNIT [5] and two versions of ComboGAN [6]) are presented in Section 5. For single-style methods, a single network model is trained for each cartoon style.

styles. Then the encoder exploits the common characteristics of different cartoon styles, leading to two advantages: (1) it can benefit from the large training data by combining the data for each style and (2) it makes MS-CartoonGAN achieve better and more stable stylization results than separately trained single-style models for individual styles.

- We propose three novel losses dedicated to our model: (1) a semantic loss for retaining semantic content and recovering flat shading, (2) an edge-promoting adversarial loss for generating clear edges and (3) a style loss for generating multiple styles with significant difference. Unlike CartoonGAN, MS-CartoonGAN uses a novel hierarchical semantic loss, which can effectively capture cartoon art forms in different abstract levels; therefore, together with style loss and an auxiliary style classifier, MS-CartoonGAN can generate more distinguishable results between different cartoon styles.
- Theoretical analysis is provided, showing that the multi-domain architecture of MS-CartoonGAN can guarantee to output desired multiple cartoon styles.

Extensive experiments show that MS-CartoonGAN can generate high-quality cartoons in multiple styles, which are substantially better than state-of-the-art style transfer methods.

2 Related Work

2.1 Non-photorealistic rendering (NPR)

Many NPR algorithms have been developed to mimic specific artistic styles including cartoons [1]. Typically, the cel shading techniques render 3D shapes in simple shading, which creates cartoon-like effect [9]. However, turning existing photos into cartoons with different styles such as the problem studied in this paper is much more challenging.

A variety of methods have been developed to create images with flat shading, mimicking cartoon styles. Such methods use either image filtering [10] or formulations in optimization problems [11]. However, it is difficult to capture rich artistic styles using simple mathematical formulas. In particular, applying filtering or optimization uniformly to the entirely image does not give the high-level abstraction that an artist would normally do. To improve the results, alternative methods rely on segmentation of images/videos [12], although at the cost of requiring some user interaction. Dedicated methods have also been developed for portrait cartoonization [13], [14]; however, such methods cannot cope with general images.

2.2 Stylization with neural networks

Instead of developing specific NPR algorithms which require substantial effort for each style, style transfer has been actively researched. Unlike traditional style transfer methods [15], [16] which require paired style/non-style images, recent studies [17], [18], [19], [20] show that the VGG network [21] trained for object recognition has good ability to extract semantic features of objects, which is very important in stylization. As a result, more powerful style transfer methods have been developed which do not require paired training images.

Given a style image and a content image, Gatys et al. [2] first proposed a neural style transfer (NST) method based on convolutional neural networks (CNNs) that transfers the style from the style image to the content image. It produces nice results for transferring a variety of artistic styles automatically. However, it requires the content and style images to be reasonably similar. Furthermore, when images contain multiple objects, it may transfer styles to semantically different regions. The results for cartoon style transfer are more problematic, as they often fail to reproduce clear edges or smooth shading.

Li and Wand [22] obtained style transfer by local matching of CNN feature maps and using a Markov Random Field for fusion (CNNMRF). However, local matching can make mistakes, resulting in semantically incorrect output. Liao et al. [23] proposed a deep analogy method which keeps semantically meaningful dense correspondences between the content and style images while transferring the style. They also compare and blend patches in the VGG feature space. Chen et al. [24] proposed a method to improve comic style transfer by training a dedicated CNN to classify comic/non-comic images. All these methods use a single style image for a content image, and the result heavily depends on the chosen style image, as there is inevitable ambiguity regarding the separation of styles and content in the style image.

2.3 Image synthesis with GANs

An alternative promising approach to image synthesis is to use generative adversarial networks (GANs) [25], [26]. Several works [27], [28], [29] have provided GAN solutions to pixel-to-pixel image synthesis problems. However, these methods require paired image sets for the training process which is impractical for stylization due to the challenge of obtaining such corresponding image sets.

To address this fundamental limitation, CycleGAN [3] was recently proposed, which is a framework able to perform image translation with unpaired training data. However, as well as some other GAN models (e.g., UNIT [4], GDWCT [7]), CycleGAN does not consider the special characteristics (i.e., high-level abstraction and clear edges) of cartoon images, and then produces poor results for cartoon stylization. Recently, CartoonGAN [8] is proposed, which learns a one-way mapping from the photo to cartoon domain from unpaired data using a content loss to preserve semantic content. In this paper, we substantially extend CartoonGAN to transfer photos to multiple cartoon styles using a single network model.

2.4 Multi-domain image-to-image synthesis

All above mentioned methods are limited in image-to-image synthesis between only two domains. To tackle the multi-domain image-to-image synthesis problem, StarGAN [30] is proposed to use a single generator for learning mappings between all available domains by taking information of both the images and target domains as input. StarGAN is designed for facial images of different styles, in which the semantic content can be automatically detected by facial components. Other multi-domain synthesis models include SG-GAN [31] and StyleGAN [32]. The former is designed to train with sparsely-grouped datasets, i.e., most training data are unlabeled and only a few are labeled. The latter is mainly designed for facial attribution modification, which makes use of face semantics in a latent space and are not suitable for real-world photo cartoonization.

CD-GAN [33], RG-UNIT [34] and MUNIT [5] are proposed for multi-modal unsupervised image-to-image translation. They all use a latent-code-sharing assumption. CD-GAN [33] learns high-level features across different domains and can generate diverse and realistic results. RG-UNIT [34] takes the power of a retrieval system to complete translation. But the fixed encoder-decoder design restricts its capability of transforming images in different style domains within one model. MUNIT [5] recombines the content code from an input image with the style code sampled from the set of style images. This design strategy puts high demands on the network structure. MUNIT works well on natural images of different styles, such as similar scenes at different seasons. However, for the highly simplified and abstract images such as cartoons, fully decoupling various images into the content and style codes is very difficult, and thus MUNIT works poorly on photo cartoonization.

Another related work is the ComboGAN [6] that simplifies n CycleGAN or UNIT models (each model corresponds to a pair of image domains) into a model with n components (each component corresponds to an image domain). Similar to MUNIT, ComboGAN applies a cycle loss to make use of unpaired training data. However, due to information unbalance between natural image domain A and the highly simplified and abstract cartoon domain B, the cycle loss from A to B and back to A does not work well. Therefore, ComboGAN is not suitable for multi-style photo cartoonization. Our experiments in Section 5 demonstrate this observation.

3 MS-CartoonGAN

We propose MS-CartoonGAN for the multi-style photo cartoonization task. MS-CartoonGAN follows the standard GAN model, which consists of two CNNs. One is the generator G trained to produce output that fools the discriminator. The other is the discriminator D that classifies whether the output image is from the real cartoons or synthetic. To learn multiple cartoon styles, we decouple the generator G into one encoder for real-world photos and multiple decoders for multiple cartoon styles. We also use multiple discriminators corresponding to each cartoon style. To generate significant difference between output styles, we further introduce an auxiliary classifier and a style loss.

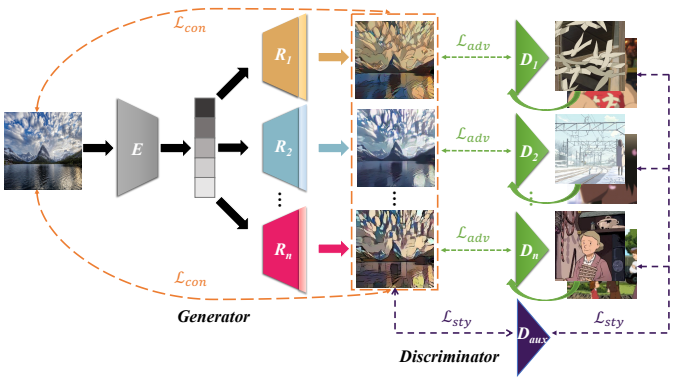


Fig. 2. Network architecture in MS-CartoonGAN. The generator G consists of a shared encoder E (for photo contents) and multiple decoders R_i (for different cartoon styles). The composite discriminator D consists of a set of discriminators $\{D_i\}_{i=1}^n$ (for different cartoon styles) and an auxiliary classifier D_{aux} (to define a style loss).

Figure 2 shows an overview of our GAN architecture. We denote the shared encoder as E and multiple decoders as R_i, $i = 1, 2, ..., n$, where R_i is the ith decoder for the ith cartoon style and n is the number of total styles that need to be learned. The composite discriminator D consists of two parts: one is $\{D_i\}_{i=1}^n$, in which each discriminator D_i judges whether an image is a real cartoon image of ith style or a synthesized one, and the other is a classifier D_{aux}, which assign style labels to real or generated cartoon images.

In this paper, we assume that the cartoon images of each distinguishable style lies in a cartoon manifold[1]. We formulate the process of learning to transform real-world photos into cartoon images of different styles as a one-to-many mapping function. This function maps the photo manifold $\mathcal{P}$ to the cluster of cartoon manifolds $\{\mathcal{C}^i\}_{i=1}^n$, and is learned using training data $S_{data}(\mathcal{P}) = \{p_k \,|k = 1 \dots n_p\} \subset \mathcal{P}$ and $S_{data}(\mathcal{C}^i) = \{c_j^i \,|j = 1 \dots n_i\} \subset \mathcal{C}^i$, $i = 1, 2, \cdots, n$, where n_p and n_i are the numbers of photos and cartoon images of the ith-style, respectively. For example, the training data shown in Figure 1 contain $n_p = 5402$ photos, three cartoon styles with $n_1 = 3617$, $n_2 = 4573$ and $n_3 = 2786$.

To learn multiple cartoon styles, the composite discriminator $D = (\{D_i\}_{i=1}^n, D_{aux})$ is trained for pushing G to reach its goal by distinguishing images in the cartoon manifold $\mathcal{C}^i$ from other images, $i = 1, 2, \cdots, n$, and providing the adversarial loss for G. Let $\mathcal{L}$ be the loss function, G^* and D^* be the weights of the networks. Our objective is to solve the min-max problem:

$$(G^*, D^*) = \arg \min_G \max_D \mathcal{L}(G, D) \tag{1}$$

We present the network architecture of MS-CartoonGAN in Section 3.1 and propose three dedicated loss terms for G and D in Section 3.2. To further improve the network convergence, we propose an initialization phase and incorporate it into MS-CartoonGAN, which is summarized in Section 3.3.

In Section 4 we present a theoretical analysis to show that the MS-CartoonGAN architecture with the dedicated loss function can guarantee to generate cartoon images of desired multiple styles.

1. In our theoretic analysis in Section 4, we do not require that different cartoon manifolds for different styles need to be disjoint; i.e., they can be intersected.

3.1 Network architecture

MS-CartoonGAN is inspired by the multimodal image-to-image translation method MUNIT [5]. MUNIT is built upon the assumption that the image representation can be decomposed into a domain-invariant content code and a domain-specific style code. MS-CartoonGAN follows the same assumption. However, MS-CartoonGAN uses a different network architecture with the following reason.

MUNIT requires a full decomposition of all images with different styles into the content and style codes, and recombine the content code from an input image with the style code sampled from the space of style images. This design strategy puts high demands on the network structure. In particular, for the highly simplified and abstract form such as cartoons, fully decoupling various cartoon images into the content and style codes is very difficult. Our experiments in Section 5 show that MUNIT works poorly on multi-style photo cartoonization.

In contrast, the network architecture of MS-CartoonGAN is designed to extract the content code from real-world photos and the multiple style codes from cartoon images of multiple styles. By recombining the content code of input photo with a style code learned from training data, MS-CartoonGAN can obtain high-quality cartoon images. Due to the rich content information in real-world photos and rich style information in cartoons, the network architecture of MS-CartoonGAN is affordable. Figure 2 illustrates this architecture and details are explained below.

3.1.1 *Generator architecture*

The generator G in MS-CartoonGAN is represented as $(E, \{R_i\}_{i=1}^n)$. For a given photo p, the encoder E learns a mapping to transform p into a shared latent space Θ that encodes both the image content of p and common characteristics shared by all cartoon styles. Each decoder R_i learns a mapping from the space Θ to the target cartoon manifold $\mathcal{C}^i$ of the ith style. We denote the combination of (E, R_i) as the generator sub-network G_i and denote the output of $G_i : \mathcal{P} \to \mathcal{C}^i$ as $G_i(p)$. Then the generator G has n outputs $\{G_i(p)\}_{i=1}^n$, each of which corresponds to a cartoon style in the training data.

Detailed structures of E and each R_i are as follows. The encoder E begins with a flat convolution stage followed by two down-convolution blocks to spatially compress and encode the images. Useful local signals are extracted in this stage for downstream transformation. Afterwards, 5 residual blocks with an identical layout are used to construct the content and manifold feature. We employ the residual block layout proposed in [35]. The decoder R_i begins with 4 residual blocks with an identical layout. Then, the output cartoon images of the ith style are reconstructed by two up-convolution blocks which contain fractionally strided convolutional layers with stride $1/2$. To this end, an additional layer, which consists of two more flat-convolution blocks with 32 and 16 channels respectively, is applied to enhance details (such as edges) of the output, followed by a final convolutional layer with a 7×7 kernel.

Compared to the single-style CartoonGAN [8], the generator design $(E, \{R_i\}_{i=1}^n)$ has the following advantages. First, given cartoon images of n styles, MS-CartoonGAN

needs to be trained only once, while n CartoonGAN networks need to be trained independently. Second, since mappings to different cartoon manifolds share the same encoder, more and diverse training data by merging cartoon images of different styles can be used to train the shared encoder network. Therefore in addition to extracting photo contents, the encoder can better characterize the common cartoon style shared by different cartoon images and then MS-CartoonGAN can generate better cartoon images than those from CartoonGAN. Third, It is easy to extend our architecture to include more cartoon styles. Given a set of cartoon images $\mathcal{C}^{n+1}$ of a new style, the encoder E in MS-CartoonGAN can be reused without re-training. Experiments in Section 5.3 demonstrate this property.

3.1.2 Discriminator architecture

We design the structure of composite discriminator D in two parts: (1) a set of multiple classification networks $\{D_i\}_{i=1}^{n}$ and (2) an auxiliary classifier D_{aux}.

Complementary to the generator sub-network G_i, each D_i judges whether an image is a real cartoon image of the ith style. Since judging whether an image is cartoon of the ith style or not is a less demanding task, instead of a regular full-image discriminator, we use a simple patch-level architecture with fewer parameters for each D_i. Different from object classification, cartoon style discrimination relies on local features of the image. Accordingly, the sub-network D_i is designed to be shallow. After the stage with flat layers, the network employs two strided convolutional blocks to reduce the resolution and encode essential local features for classification. Afterwards, a feature construction block and a 3×3 convolutional layer are used to obtain the classification response. Leaky ReLU (LReLU) [36] with $\alpha = 0.2$ is used after each normalization layer.

To help the generator better present the difference between cartoon styles, we construct an auxiliary classifier D_{aux}. It assigns a style label to input cartoon image, which defines a style loss devoting to the final objective function for every style. D_{aux} contains 4 down-convolution blocks, followed by a flat convolution stage and a softmax activation function. The output of D_{aux} is the possibility of input image belonging to each style.

3.2 Loss function

The loss function $\mathcal{L}(G, D)$ in Eq. (1) consists of three parts: (1) the hierarchical content loss $\mathcal{L}_{con}(G)$ (Section 3.2.1), which preserves the image content in different abstract levels during cartoon stylization, (2) the adversarial loss $\mathcal{L}_{adv}(G, D_{dom})$ (Section 3.2.2), which drives the generator network to achieve the desired manifold transformation, and (3) the style loss $\mathcal{L}_{sty}(G, D_{aux})$ (Section 3.2.3), which helps the network generate significant difference between cartoon styles and stabilize training process. Finally, the loss function is:

$$\mathcal{L}(G, D) = \mathcal{L}_{adv}(G, D) + \omega_1 \mathcal{L}_{con}(G) + \omega_2 \mathcal{L}_{sty}(G, D_{aux}) \quad (2)$$

where ω_1 and ω_2 are weights to balance the three given losses. Larger ω_1 leads to more content information from the input photos to be retained, results in stylized images with more detailed textures. In all our experiments, we set $\omega_1 = 0.2$ and $\omega_2 = 0.5$, which achieves a good balance of style and content preservation.

3.2.1 Hierarchical content loss $\mathcal{L}_{con}(G)$

One important goal in cartoon stylization is to ensure the resulting cartoon images retain semantic content of the input photos. Note that different styles have their own meticulous stroke forms. In MS-CartoonGAN, we adopt the high-level feature maps in the VGG network [21] pre-trained by [37], to control content preservation in different resolutions of generated images. Combining high resolution images (which keep fine details) with low resolution images (which retain important structure information) enables our network preserve content in different levels. Accordingly, we define the hierarchical content loss for each cartoon style $i \in \{1, 2, ..., n\}$ with hierarchical level $h \in \{1, 2, ..., m\}$ as:

$$\mathcal{L}_{con}(G_i) = \sum_{h=1}^{m} \mathbb{E}_{p \sim S_{data}(\mathcal{P})} [||VGG_l(H_h(G_i(p))) - VGG_l(H_h(p))||_1] \quad (3)$$

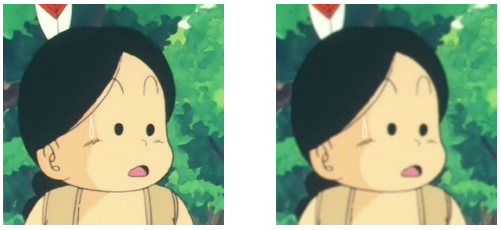
where l refers to the feature maps of a specific VGG layer, and H_h is the image downsampling operation. Note that after downsampling images, we interpolate low resolution images back to the original size and then feed them into the VGG network, since it takes an image of fixed size as input. In our experiments, we use $m = 3$, i.e., three resolutions 256×256, 128×128 and 64×64 are used to compose the hierarchical content loss. Then the overall content loss is the sum of content losses for all cartoon styles, defined as:

$$\mathcal{L}_{con}(G) = \sum_{i=1}^{n} \mathcal{L}_{con}(G_i) \quad (4)$$

Unlike other image generation methods [2], [17], we define our semantic content loss using the ℓ_1 sparse regularization of VGG feature maps between the input photo and the generated cartoon image. This is due to the fact that cartoon images have very different characteristics (i.e., clear edges and smooth shading) from photos. We observe that even with a suitable VGG layer that intends to capture the image content, the feature maps may still be affected by the massive style difference. Such differences often concentrate on local regions where the representation and regional characteristics change dramatically. ℓ_1 sparse regularization is able to cope with such changes much better than the standard ℓ_2 norm. As we will show later, this is crucial to reproduce the cartoon style. We use the feature maps in the layer 'conv4_4' to compute our semantic content loss.

3.2.2 Adversarial loss $\mathcal{L}_{adv}(G, D)$

The adversarial loss is applied to both networks G and D, which affects the cartoon transformation process in the generator network G. Its value indicates to what extent the output image of the generator G looks like a cartoon image. In previous GAN frameworks [3], [25], [28], the task of the discriminator D is to figure out whether the input image is synthesized from the generator or from the real target manifold. However, we observe that simply training the discriminator D to separate generated and true cartoon images is not sufficient for transforming photos to cartoons. This is because the presentation of clear edges is an important



(a) A cartoon image (b) Edge-smoothed version

Fig. 3. Edge smooth example. (a) A cartoon image. (b) The corresponding image by removing clear edges from (a).

characteristic of cartoon images, but the proportion of these edges is usually very small in the whole image. Therefore, an output image without clearly reproduced edges but with correct shading is likely to confuse the discriminator trained with a standard loss.

To circumvent this problem, for each cartoon style, from the training cartoon images $S_{data}(\mathcal{C}^i) \subset \mathcal{C}^i$, we automatically generate a set of images $S_{data}(\mathcal{E}^i) = \{e^i_j \ | j = 1 \dots n_i\} \subset \mathcal{E}^i$ by removing clear edges in $S_{data}(\mathcal{C}^i)$, where $\mathcal{C}^i$ and $\mathcal{E}^i$ are the cartoon manifold and the manifold of cartoon-like images without clear edges, respectively. In more detail, for each image $c^i_j \in S_{data}(\mathcal{C}^i)$, we apply the following three steps: (1) detect edge pixels using a standard Canny edge detector [38], (2) dilate the edge regions, and (3) apply a Gaussian smoothing in the dilated edge regions.

Figure 3 shows an example of a cartoon image and a modified version with edges smoothed out. Recall that for each photo p in the photo manifold $\mathcal{P}$, the generator sub-network G_i outputs a generated image $G_i(p)$ in the ith cartoon style. In MS-CartoonGAN, the goal of training each discriminator sub-network D is to maximize the probability of assigning the correct label to $G_i(p)$, the cartoon images without clear edges (i.e., $e^i_j \in S_{data}(\mathcal{E}^i)$) and the real cartoon images (i.e., $c^i_j \in S_{data}(\mathcal{C}^i)$), such that the generator sub-network G_i can be guided correctly by transforming the input photo to the correct cartoon manifold $\mathcal{C}^i$. Therefore, we define the edge-promoting adversarial loss for each cartoon style $i \in \{1, 2, ..., n\}$ as:

$$\begin{aligned}\mathcal{L}_{adv}(G_i, D_i) = &\mathbb{E}_{c^i_j \sim S_{data}(\mathcal{C}^i)}[\log D_i(c^i_j)] \\ &+ \mathbb{E}_{e^i_k \sim S_{data}(\mathcal{E}^i)}[\log(1 - D_i(e^i_k))] \\ &+ \mathbb{E}_{p_t \sim S_{data}(\mathcal{P})}[\log(1 - D_i(G_i(p_t)))].\end{aligned} \tag{5}$$

Then the overall edge-promoting adversarial loss is the sum of adversarial losses for all cartoon styles, defined as:

$$\mathcal{L}_{adv}(G, D) = \sum_{i=1}^{n} \mathcal{L}_{adv}(G_i, D_i) \tag{6}$$

3.2.3 Style loss $\mathcal{L}_{sty}(G, D_{aux})$

To enable G generate multiple styles with significant difference, the style loss is further introduced. This loss makes use of an auxiliary classifier D_{aux}. Given an input cartoon image c which can be real or generated, D_{aux} outputs the possibility that c belongs to a style. The possible styles considered by D_{aux} include n cartoon styles (denoted as $s = \{i\}_{i=1}^n$) and one extra style of edge-smoothing cartoons (denoted as s_e). D_{aux} is trained with the data $\{S_{data}(\mathcal{C}^i), S_{data}(\mathcal{E}^i)\}_{i=1}^n$. Then the style loss for G_i is defined as

$$\begin{aligned}\mathcal{L}_{sty}(G_i, D_{aux}) = &\mathbb{E}_{c^i_j \sim S_{data}(\mathcal{C}^i), s}[-\log D_{aux}(s|c^i_j)] \\ &+ \mathbb{E}_{e^i_k \sim S_{data}(\mathcal{E}^i), s_e}[-\log D_{aux}(s_e|e^i_k)] \\ &+ \mathbb{E}_{p_t \sim S_{data}(\mathcal{P}), s}[-\log D_{aux}(s|G_i(p_t))]\end{aligned} \tag{7}$$

Finally, the overall style loss is the sum over each G_i:

$$\mathcal{L}_{sty}(G, D_{aux}) = \sum_{i=1}^{n} \mathcal{L}_{sty}(G_i, D_{aux}) \tag{8}$$

Fig. 4. For an original photo (left), the image (right) is the result after the initialization phase. See Section 3.3 for details.

3.3 Training with an initialization phase

Since the GAN model is highly nonlinear, with random initialization, the optimization can be easily trapped at sub-optimal local minimum. To help improve its convergence, we propose a new initialization phase. Note that the target of the generator G is to reconstruct the input photo in n cartoon styles while keeping the semantic content. We start the adversarial learning framework with a generator which only reconstructs the content of input images, i.e., the generator first simulates the distribution of real photos $S_{data}(p)$. For this purpose, in the initialization phase, we pre-train the generator G with only the hierarchical semantic content loss $\mathcal{L}_{con}(G)$.

Figure 4 shows an example of the reconstructed image after 10 epochs of this initialization training phase, which already produces reasonable reconstruction. Our experimental results show that this simple initialization phase helps Multi-style CartoonGAN fast converge to a good configuration, without premature convergence. Similar observation is made in [2] which uses the content image to initialize the result image to improve style transfer quality.

4 Theoretical Analysis

Property 1. *As G and D are given enough capacity (i.e., no limit for parameters), the optimal discriminator $D^* = \{D^*_i\}_{i=1}^n$, which solves the min-max problem in Eq.(1), can correctly classify the edge-smoothed images in $S_{data}(\mathcal{E}^i)$, $i = 1, 2, \cdots, n$, to be fake.*

Proof. The optimal discriminator $D^* = \{D^*_i\}_{i=1}^n$ that solves the min-max problem in Eq.(1) must maximize each adversarial loss $\mathcal{L}_{adv}(G_i, D_i)$ defined in Eq.(5). Let $\mathcal{X}$ be an embedding space which contains $\mathcal{C}^i$, $\mathcal{E}^i$ and $G_i(\mathcal{P})$. We rewrite the loss $\mathcal{L}_{adv}(G_i, D_i)$ as

$$\begin{aligned}\mathcal{L}_{adv}(G_i, D_i) = &\int_{x \in \mathcal{C}^i} p_{c^i}(x) \log D_i(x) \mathrm{d}x \\ &+ \int_{x \in G_i(\mathcal{P})} p_{g_i}(x) \log(1 - D_i(x)) \mathrm{d}x \\ &+ \int_{x \in \mathcal{E}^i} p_{e^i}(x) \log(1 - D_i(x)) \mathrm{d}x.\end{aligned} \tag{9}$$

where $p_{c^i}(x)$, $p_{g_i}(x)$ and $p_{e^i}(x)$ are distributions of data $S_{data}(\mathcal{C}^i)$, $G_i(S_{data}(\mathcal{P}))$ and $S_{data}(\mathcal{E}^i)$ in $\mathcal{X}$, respectively. Note that $p_{c^i}(x)$, $p_{g_i}(x)$ and $p_{e^i}(x)$ are zero in the subspaces $\mathcal{X} \setminus \mathcal{C}^i$, $\mathcal{X} \setminus G_i(\mathcal{P})$ and $\mathcal{X} \setminus \mathcal{E}^i$, respectively. Let $p_i(x) = \frac{1}{2}(p_{g_i}(x) + p_{e^i}(x))$. We have

$$\begin{aligned} &\int_{x\in\mathcal{X}} p_i(x)\mathrm{d}x = \\ &\frac{1}{2}\int_{x\in G_i(\mathcal{P})} p_{g_i}(x)\mathrm{d}x + \frac{1}{2}\int_{x\in\mathcal{E}^i} p_{e^i}(x)\mathrm{d}x = 1 \end{aligned} \tag{10}$$

So $p_i(x)$ is also a distribution in $\mathcal{X}$. We have

$$\begin{aligned} &\mathcal{L}_{adv}(G_i, D_i) = \\ &\int_{x\in\mathcal{C}^i} p_{c^i}(x)\log D_i(x)\mathrm{d}x + \int_{x\in\mathcal{X}} 2p_i(x)\log(1-D_i(x))\mathrm{d}x \\ &= \int_{x\in\mathcal{X}} \left(2p_i(x)\log(1-D_i(x)) + p_{c^i}(x)\log D_i(x)\right)\mathrm{d}x. \end{aligned} \tag{11}$$

By comparing the form (11) with the standard adversarial loss, i.e., Eq.(3) in [25], and using the result in Proposition 1 in [25], we have that for fixed G_i, the optimal discriminator D_i^* has the form

$$D_i^*(x) = \frac{p_{c^i}(x)}{p_{c^i}(x) + 2p_i(x)}, \tag{12}$$

Note that during the training, no sample x will come from $\mathcal{X} \setminus (\mathcal{C}^i \cup G_i(\mathcal{P}) \cup \mathcal{E}^i)$ and then $p_{c^i}(x) + 2p_i(x) \neq 0$. Furthermore, we have $\mathcal{C}^i \cap \mathcal{E}^j = \emptyset$, $\forall i, j$, because cartoon images have clear edges. Therefore, for $x \in \mathcal{E}^i$ we have $p_{c^i}(x) = 0$ and thus $D_i^*(x) = 0$. So the optimal discriminator can classify the edge-smoothed cartoon images to be fake. □

Property 2. *The optimal generator G_i^* can generate desired cartoon images in $\mathcal{C}^i$.*

Proof. By Property 1, when the discriminator D_i is optimal, $D_i^*(x) = 0$, if $x \in \mathcal{E}^i$. At this case, $p_i(x) = \frac{1}{2}p_{g_i}(x)$. Then by substituting the optimal discriminator in Eq.(12) into the adversarial loss in Eq.(11), we have

$$\begin{aligned} &\mathcal{L}_{adv}(G_i, D_i^*) \\ &= \int_{x\in\mathcal{X}} \left(2p_i(x)\log(1-D_i^*(x)) + p_{c^i}(x)\log D_i^*(x)\right)\mathrm{d}x \\ &= \int_{x\in\mathcal{X}} \left(p_{g_i}(x)\log(1-D_i^*) + p_{c^i}(x)\log D_i^*(x)\right)\mathrm{d}x, \end{aligned} \tag{13}$$

The last equation is exactly the form of Kullback-Leibler divergence, which reaches the minimum if and only if $p_{g_i} = p_{c^i}$. Then the optimal generator G_i^* generates desired cartoon images in $\mathcal{C}^i$. □

5 Experiments

We implemented the proposed MS-CartoonGAN in PyTorch. All experiments were performed on a PC with an NVIDIA Titan Xp GPU. To compare MS-CartoonGAN with state of the art, we collected the training and test data as presented in Section 5.1. In Section 5.2, we present the comparison between MS-CartoonGAN and representative stylization works including both single-style and multi-style methods. Thanks to the shared single encoder that exploits the common characteristics of different cartoon styles, MS-CartoonGAN can not only generate high-quality cartoon images in different styles, but also be easily extended to quickly learn and output new styles. We study this extension in Section 5.3. We also present an ablation study to analyze the effectiveness of each component in our MS-CartoonGAN model in Section 5.4.

All images in this section are provided in high resolution for zoom-in examination.

5.1 Data

The training data contains real-world photos and cartoon images, and the test data only includes real-world photos. All the training images are resized and cropped to 256×256.

Photos. 6,153 photos are collected from Flickr, in which 5,402 photos are for training and others for testing.

Cartoon images. Different artists have different painting styles when creating cartoon images of real-world scenes. To obtain multiple sets of cartoon images in which each set has the same style, we use the key frames of cartoon films drawn and directed by the same artist as the training data. In our experiments, we use three styles in Section 5.2:

- Shinkai style: we use 4,573 cartoon images from several short cartoon videos directed by Makoto Shinkai. This style depicts fine details, and has gentle and delicate strokes in light color, such as blue and green.
- Spirited style: we use 3,617 cartoon images from the cartoon films "Spirited Away" directed by Miyazaki Hayao. Hayao always draws his fairy tale world in a romantic way, so this style has very high color contrast by using lines to outline the abstract content and filling different regions with various colors.
- Longholiday style: we use 2,786 cartoon images from the French cartoon animations "Les Grandes Grandes Vacances" directed by Delphine Maury and Olivier Vinuesa. This style uses strong solid edges and deep shadows akin to oil painting.

Given the MS-CartoonGAN already trained by the above three styles, in Section 5.3, we use the fourth style:

- Voice style: 4,390 cartoon images from the cartoon film "A Silent Voice" directed by Naoko Yamada

to illustrate the easy style extension in MS-CartoonGAN by only training (using the new fourth style) a newly added decoder/sub-discriminator with the fixed encoder.

5.2 Comparisons to the state of the art

5.2.1 Comparison to single-style methods

We compare MS-CartoonGAN with recently proposed single-style CNN-based stylization methods, namely NST [2], CycleGAN [3], UNIT [4], CartoonGAN [8], and GDWCT [7]. Since they are all single-style methods, we train each of them multiple times to get different style results. In contrast, we only need to train our model once. Note that the original NST takes one style image I_s and one content image I_c as input, and transfers the style from I_s to I_c. For fair comparison, we apply two adaptations of NST. In the first adaptation, we manually choose a style image which has

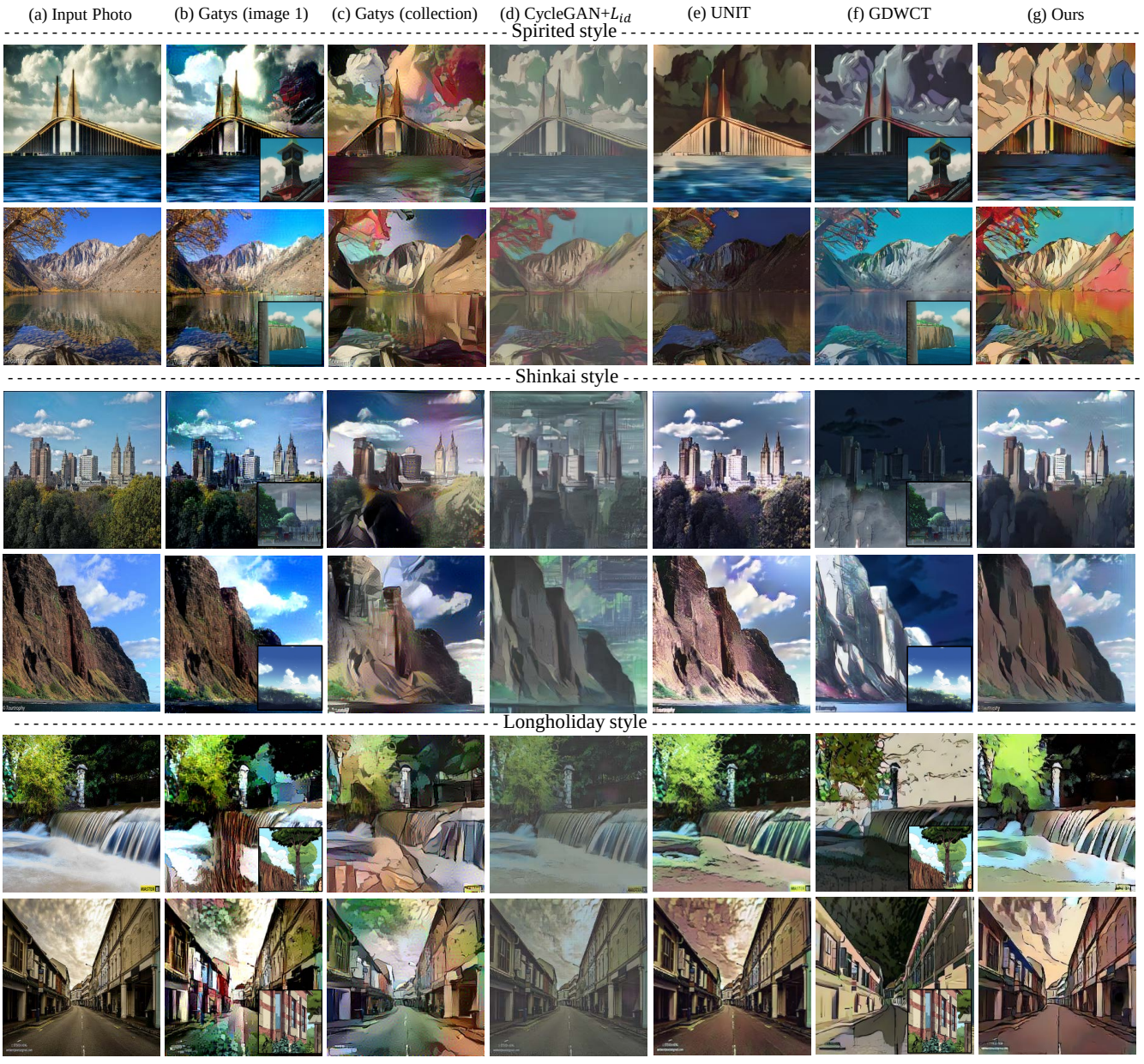


Fig. 5. Comparison with state-of-the-art single-style methods, i.e., NST [2], CycleGAN [3] with identity loss L_{id}, UNIT [4], and GDWCT [7], for Spirited (top), Shinkai (middle) and Longholiday (bottom) styles. Gatys (image 1) and Gatys (collection) are two adaptations of NST where a cartoon image with close content to the input photo (shown in the corner of the images) and all the cartoon images are used for training, respectively.

close content to the input photo. In the second adaptation, we extend NST to take all the cartoon images for training, similar to the comparison in [3]. For CycleGAN, we test two versions: with and without identity loss L_{id}. We find that with L_{id}, the results of CycleGAN can better preserve content; thus we choose this version for comparison.

Qualitative results compared with NST, CycleGAN, UNIT and GDWCT are presented in Figure 5, which clearly demonstrate that they cannot deal with cartoon styles well (Figures 5b-5f). In comparison, by reproducing the necessary clear edges and smooth shading while retaining the content of the input photo, our MS-CartoonGAN model produces high-quality results in each style (Figure 5g).

More specifically, NST [2] using only a single style image may not be able to fully learn the style, especially for areas in the target image whose content is different from the style image (Figure 5b). When NST is extended to take more training data, rich styles can be better learned. However, the stylized images tend to have local regions stylized differently, causing inconsistency artifacts (Figure 5c).

The results of CycleGAN do not capture cartoon styles well. Although the identity loss is useful to better preserve content and the results tend to reproduce some key textures of cartoon images, they appear to ignore the semantics of cartoon images and far from satisfactory (Figure 5d).

The results of UNIT (Figure 5e) are better than NST and CycleGAN. It is owing to the shared-latent space assumption and the framework in UNIT. The assumption is useful to infer the joint distribution from the marginal distributions in the photo domain and cartoon domain respectively. But the results of UNIT are clearly worse than ours, which suggests the power of edge-promoting adversarial loss and VGG feature maps to extract hierarchical content information in our method. To make details more clearly, Figure 6

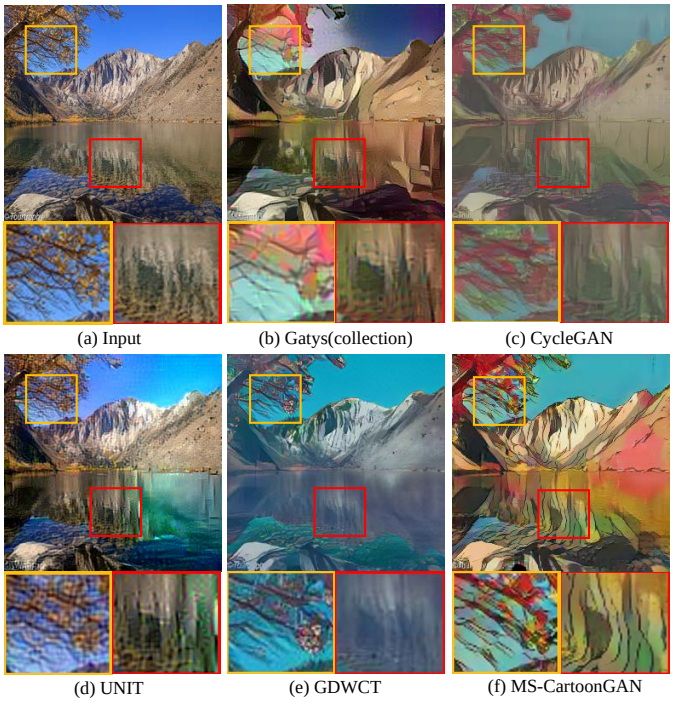


Fig. 6. Details of edge generation. (a) Input image, (b) NST [2] using all the images in the training set as the style image, (c) CycleGAN [3] with the identity loss, (d) UNIT [4], (e) GDWCT [7], (f) MS-CartoonGAN.

shows close-up views of an example in Figure 5. Obviously, our MS-CartoonGAN generates essential edges which are very important characteristics for cartoon styles.

When style images have obvious textures, GDWCT can well capture style features and transfer them to content images. However, cartoon images are highly simplified with little texture information. So the cartoonization results of GDWCT are not as good as ours (Figure 5f). Furthermore, GDWCT focuses on local regions, resulting in missing important global structure information of input images.

Since MS-CartoonGAN is extended from the single-style CartoonGAN, we compare CartoonGAN separately. Referring to Figure 7, it is clearly observed that our method learns high level simplification and flat shading better while preserving key content information. Smooth and solid edges are also clearly exhibited in our results. These improvements are due to our design of the shared framework. Since the encoder is shared by all styles, it captures the common aspects of cartoon styles and exploits more training data, so has better capability to capture the content and style characteristics of different cartoon styles. Benefiting from the power of auxiliary classifier, our encoder-decoder framework can enhance differences between styles and have more stable results. It is also remarkable that results of different styles have their own meticulous strokes as real cartoon images. For example, Shinkai style draws details carefully, and Spirited style and Longholiday style tend to drop unimportant details. We achieve this by our hierarchical content loss which can learn content from different abstract levels.

5.2.2 Comparison to multi-style methods

To demonstrate the superiority of our multi-task framework, we take state-of-the-art multi-domain methods MUNIT [5] and ComboGAN [6], as our baseline models for comparison. For MUNIT, during training, real-world images and cartoon images of each style are treated as target domains. During testing, to guide the network to generate cartoon images of a specific style, the style code extracted from a representative cartoon image of the target style is used to provide guidance. For ComboGAN, we follow the same way as MUNIT to organize traing data, and compare two versions, i.e., without or with the identity loss. These methods and our multi-style CartoonGAN generate results of multiple styles using only one model. Results of different styles are displayed in different rows in Figure 8 for better comparison.

For multi-task translation, a good method is expected to produce images that not only look plausible for given styles, but also clearly present the unique characteristics of individual styles, making them visually different from other styles. Both baseline methods decouple the generator into an encoder and a decoder to learn content and style, but cannot deal with our problem of mapping the real-world photo manifold to multiple cartoon manifolds well. As shown in Figure 8, although MUNIT and ComboGAN match the color to specific styles well and keep the semantics of input images, they cannot reconstruct content in specific cartoon styles precisely, making image content hard to recognize. Moreover, every cartoon style has its own way to form edges and shading, which is difficult to learn in a common multi-task model. Our method benefits from the architecture with a shared encoder and individual decoders/sub-discriminators, and synthesizes output images with high quality details. As a result, we can generate multi-style results with clearer edges and smoother shading.

5.2.3 Quantitative evaluations

Training Time. Our method takes less training time when training multi-style cartoonization than other methods. For single-style cartoonization, CycleGAN with L_{id}, UNIT and GDWCT take 0.548s, 0.566s and 1.011s respectively, in each iteration. For multi-style cartoonization, using these single-style methods requires training multiple models and the training time increases linearly with the number of styles, which is not efficient. ComboGAN designs a strategy to save training time but the strategy is found ineffective in our photo cartoonization task, largely because it only performs translation between two arbitrary domains each time. ComboGAN costs 0.213s in each iteration for one style and MUNIT costs 0.368s. In comparison, our MS-CartoonGAN takes advantage of the shared encoder and uses VGG feature maps rather than a cycle architecture, reducing the training time efficiently. Our method takes 0.248s for each iteration when the number of styles n = 3, only 0.082s on average for each style. With the basic pair of encoder and decoder, adding one decoder for a specific cartoon style only takes 0.069s (see Section 5.3 for more details). Note that, CycleGAN, UNIT, MUNIT and ComboGAN all need roughly 10^6 iterations, but GDWCT and our method only need iterations at the order of 10^5. So our method can learn multiple translations more efficiently and can be easily extended to more styles.

User Study. It is generally challenging to evaluate the visual quality of images, in particular for image stylization. Thanks to the activities of "anime pilgrimage" in the world, we can get real-world scenes corresponding to the scenes in a cartoon movie. We randomly select 70 such

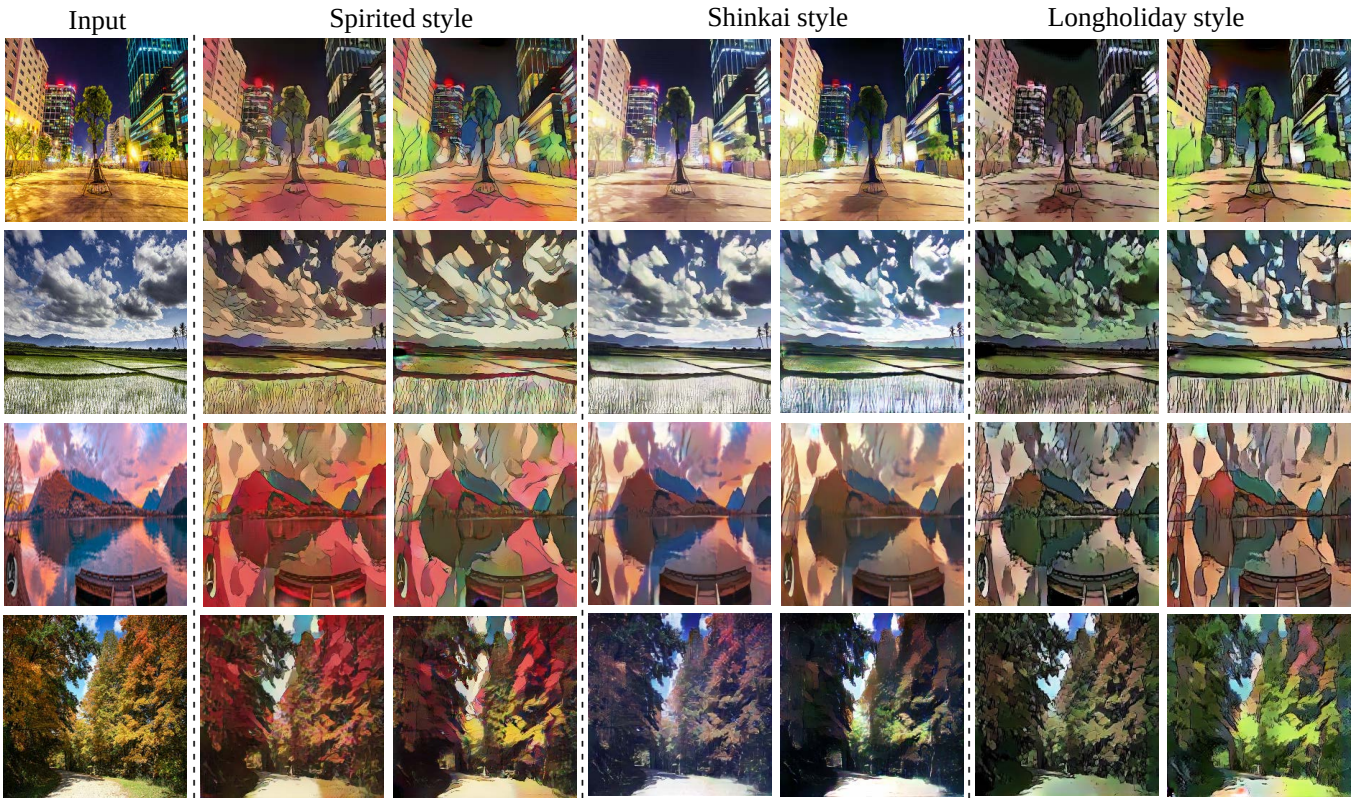


Fig. 7. Comparison between CartoonGAN and MS-CartoonGAN. For each style, the left column is generated by single-style CartoonGAN and the right one is generated by MS-CartoonGAN.

Fig. 8. Multi-style translation comparison with state-of-the-art multi-style methods, i.e., MUNIT [5], ComboGAN [6] without or with identity loss for Spirited (top), Shinkai (middle) and Longholiday (bottom) styles.

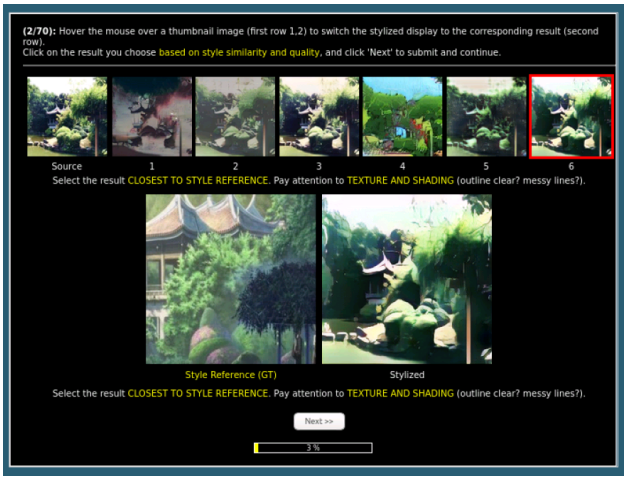


Fig. 9. A sample question and user interface setup in our user study.

real-world scenes and translate them into different cartoon styles (i.e., Spirited style, Shinkai style or Longholiday style) using three multi-style methods, namely ComboGAN, MUNIT and MS-CartoonGAN. We also compare CycleGAN, GDWCT and single-style CartoonGAN as representatives of single-style methods. Then we conduct a user study to let participants compare cartoonized results of these six methods, using the cartoon scenes from cartoon movies as a reference. Each participant needs to answer 70 questions. In each question, we ask the participant to compare the generated results with the corresponding style movie scene and choose the closest one from 6 choices. A sample question is shown in Figure 9. 22 participants took part in the user study and the percentage of each method chosen as the best is shown in Table 1. As indicated from the table, MUNIT and ComboGAN are rarely selected because they cannot handle the wide range of scenes flexibly. Although CycleGAN receives more preference, the proportion chosen is still far lower than ours, since our method can produce clear edges, smooth color shading and relatively simple texture which is obviously shown in Figures 5 and 6. Without image content analysis, GDWCT tends to migrate texture into inappropriate regions. The performance of single-style CartoonGAN is close to our MS-CartoonGAN. However, MS-CartoonGAN is more stable and obtains good results in all tasks, while single-style CartoonGAN does not perform well in Longholiday style. Overall, our multi-style method produces good cartoonization results and has better capa-

TABLE 1
User study result. The percentage of each method chosen as the best.

Methods	Spirited Style	Shinkai Style	Paprika style
CycleGAN [3]	15.91%	14.00 %	7.20%
GDWCT [7]	10.68%	2.15%	3.03%
MUNIT [5]	0.91%	1.08%	1.14%
ComboGAN [6]	0.91%	7.06%	1.14%
CartoonGAN [8]	31.82%	37.08%	28.79%
MS-CartoonGAN	39.77%	38.64%	58.71%

TABLE 2
Fréchet Inception Distance results of different methods.

Methods	Spirited style	Shinkai style	Longholiday style	Average
CycleGAN [3]	124.49	122.11	174.18	140.26
GDWCT [7]	154.33	132.90	121.66	136.29
MUNIT [5]	170.29	145.79	168.71	161.59
ComboGAN [6]	228.08	170.70	207.33	202.03
CartoonGAN [8]	151.29	129.52	155.03	145.28
MS-CartoonGAN	131.53	125.22	119.05	128.60

bility to handle diverse cartoon styles.

Fréchet Inception Distance (FID). FID is proposed to evaluate the similarity between distributions of two drawing sets [39], e.g., real and generated cartoon images. The lower the score, the closer the two distributions are. We use FID to evaluate the stablility of our and baseline results for different styles. In our experiment, 761 samples are translated into each of three styles, leading to 2,283 generated images in total. As summarized in Table 2, compared with single-style methods, our model has a consistent and stable performance for three styles. CycleGAN has lower scores in two styles but much higher scores in one style and average than MS-CartoonGAN. The scores of GDWCT, MUNIT, ComboGAN and CartoonGAN are all lower than MS-CartoonGAN. These results are possibly due to that our shared-encoder multi-decoder framework is stable and better in generating diverse multi-style images than other models.

5.3 Extension to new style

Compared to other multi-style methods, MS-CartoonGAN has a distinguishing characteristic that it can include more styles by only partially training additional decoder/discriminator with fixed encoder. Here we demonstrate the ability of our shared encoder for generating images of a new cartoon style. We denote the MS-CartoonGAN model trained by three styles in Section 5.2 as MSC_3. Now we want to include one more Voice style into it.

Figure 10b shows a result of the MS-CartoonGAN model, which is trained from MSC_3: we fixed the encoder in MSC_3 and only trained the newly added decoder/discriminator using the training data in the Voice style. The additional training converges using only 50 epochs. The result shows good simplification and abstraction, and has clear edges, demonstrating the ability of the shared encoder to extract common characteristics of cartoon styles. To expand style differences or better distinguish similar styles, we can also modify the auxiliary style classifier and adjust it to the increased number of categories, as shown in Figure 10c. Both ways can lead to desirable results. Note that we do not need the initialization phase anymore when a pre-trained encoder is used. These features make MS-CartoonGAN desirable in the application of rapid and adaptive deployment on the mobile terminals.

Fig. 10. Results of extending our network to include a new cartoon style. (a) the input photo. (b) the generated cartoon image in Voice style by only adding a pair of decoder and discriminator without updating the auxiliary classifier. (c) the generated cartoon image in Voice style by adding a pair of decoder and discriminator, and adjusting the auxiliary classifier. (d) an example in Voice style.

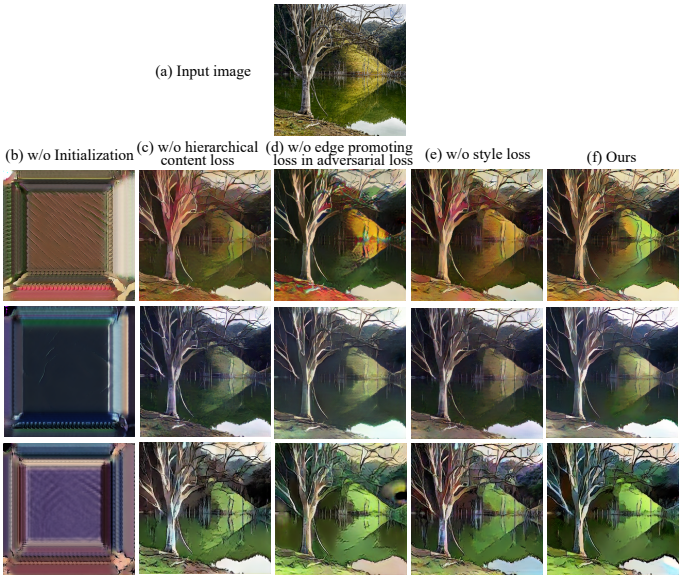


Fig. 11. Results of removing/changing components in the loss function of MS-CartoonGAN: (a) input photo, (b) without initialization process, (c) using the original content loss as CartoonGAN [24] instead of our hierarchical content loss, (d) removing edge promoting loss in the adversarial loss, i.e. using vanilla adversarial loss, (e) our MS-CartoonGAN. Note that except for the input photo, each row corresponds to one style.

5.4 Ablation study

5.4.1 *Study on initialization and loss*

First, we perform ablation experiments to study the role of initialization and loss terms in MS-CartoonGAN. Figure 11 shows the results of ablation analysis of our full loss function. The following observations show that each component plays an important role in MS-CartoonGAN. (1) The initialization phase helps the generator G quickly converge to reasonable cartoon manifolds of different styles. As shown in Figure 11b, without initialization, the generated results lose content features totally. (2) As shown in Figure 11c, using the original content loss in CartoonGAN [8], the generated results cannot retain semantic content and recover flat shading in different abstract levels. So our hierarchical content loss is more suitable in the multi-style learning task. (3) The elaborately designed edge loss guides the generator G to produce clear edges in results, leading to better cartoon style images. (4) The auxiliary classifier together with the style loss helps our network generate significant differences between styles, as demonstrated in Figures 11e and 11f. The FID scores in our ablation study are summarized in Table 3, showing the effectiveness of our architecture design.

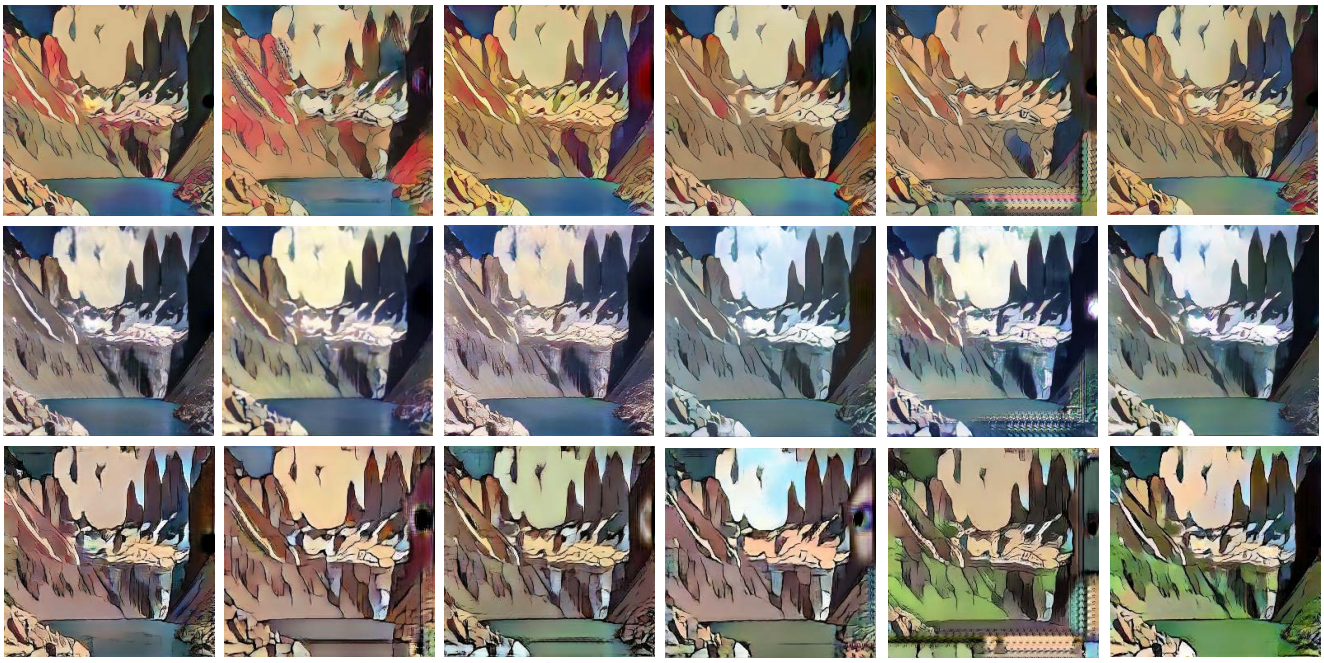


Fig. 12. Results of different architecture designs for decoders. The first column shows results using basic architecture in CartoonGAN [8] for comparison. Based on the basic architecture, (a) adds a 64-channel convolution block after the first up-convolution block, (b) adds a 64-channel convolution block after the second up-convolution block, (c) adds a 32-channel convolution block after the second up-convolution block, (d) adds a 16-channel convolution block after the second up-convolution block, (e) adds a 32-channel and a 16-channel convolution blocks after the second up-convolution block, i.e., our final architecture.

TABLE 3
Fréchet Inception Distance results for ablation study.

Methods	Spirited style	Shinkai style	Longholiday style	Average
w/o hierarchical content loss	158.86	133.24	136.31	142.80
w/o edge loss in adversarial loss	158.29	129.39	124.41	137.36
w/o style loss	157.55	146.23	136.75	146.84
Ours	131.53	125.22	119.05	125.26

5.4.2 Study on additional layer

Note that in MS-CartoonGAN, in each decoder R_i, before the final convolutional layer with 7×7 kernel, we add an additional layer which consists of two more flat-convolution blocks with 32 and 16 channels respectively. As a comparison, the single-style CartoonGAN [8] does not have this additional layer. We further perform ablation experiments to study the role of this additional layer and to explore the best shared network design.

First, we fix the residual block allocation of the encoder and decoders, so that they are almost evenly distributed. We adjust the remaining architecture of decoders to show the advantages of using additional convolutions before the output. Results of using different decoder designs are shown in Figure 12. Based on the architecture of single-style CartoonGAN [8], we add convolution blocks either after the first up-convolution block or after the second up-convolution block, and compare different designs. Since edge features (which are important for cartoon images) are small in the whole image to generate, adding convolution blocks to later blocks of convolutions can help synthesize details according to our needs. The number of parameters of each design is shown in Table 4. We can find that results with two additional convolution blocks using 32 channels and 16 channels respectively (shown in Figure 12e) achieve better results. On the other hand, as shown in Table 4, the design of shared encoder reduces the number of parameters effectively and the design of additional layers does not need too many extra parameters to train compared with the encoder in single-style CartoonGAN. So, this design is chosen as our decoder design.

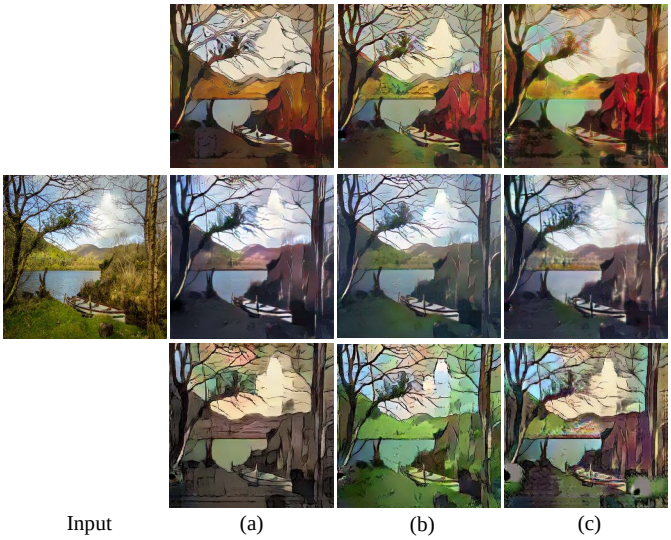


Fig. 13. Results of different allocations of shared residual blocks (with a total of 9 residual blocks). (a) uses 4 residual blocks in the shared encoder and 5 residual blocks in each decoder, (b) uses 5 residual blocks in the shared encoder and 4 residual blocks in each decoder, (c) uses 6 residual blocks in the shared encoder and 3 residual blocks in each decoder. Each row corresponds to one cartoon style. We adopt (b) for our final architecture to balance the quality of results and model consumption.

TABLE 4
The total numbers of parameters of basic architecture and each design with additional layers. The shared encoder and basic decoder without additional layers are also shown for comparison.

Design	Parameters
shared encoder	7.01M
basic decoder	5.28M
64-channel after first up-convolution	5.35M
64-channel after second up-convolution	5.35M
32-channel after second up-convolution	5.30M
16-channel after second up-convolution	5.28M
32-16-channel after second up-convolution	5.31M

5.4.3 Study on shared residual blocks

Next, we explore the best shared network design by comparing different allocations of residual blocks to the shared encoder and decoders. We fix the total number of residual blocks in the encoder and a decoder to 9. Allocating more residual blocks to the shared encoder can help save time and space during training. While allocating more residual blocks to decoders can help improve multi-style translation quality. Results of using different numbers of residual blocks in the encoder and decoders are shown in Figure 13, showing that (1) results in Figure 13c are worst and (2) results in Figure 13b are slightly better than those in Figure 13a. We finally use 5 residual blocks in the shared encoder and 4 residual blocks in decoders in our implementation, to reach a balance between the quality of results and the resource consumption of our model.

6 Conclusion

In this paper, we propose MS-CartoonGAN, a generative adversarial network to transform real-world photos to cartoon images of multiple styles. Our method achieves faithful cartoonization using three effective loss terms: (1) a novel edge-promoting adversarial loss for clear edges, and (2) a hierarchical content loss with an ℓ_1 sparse regularization of high-level feature maps in the VGG network, which provides sufficient flexibility for reproducing smooth shading, and (3) a style loss defined from an auxiliary style classifier, which enables our model generate significant difference between styles.

MS-CartoonGAN is dedicated to handling the task of style transfer for multiple cartoon styles at the same time. In particular, MS-CartoonGAN improves upon single-style CartoonGAN [8] in the following aspects. (1) Training our model once can realize style transfer of multiple cartoon styles, while single-style CartoonGAN needs to be trained multiple times. (2) The design of a shared encoder and multiple decoders for generator helps reduce the complexity of framework. (3) Since the encoder is shared by multiple styles, it is trained with more data and learns to extract content information and common characteristics of cartoon images better, which makes our model produce better results than single-style CartoonGAN. (4) An auxiliary classifier is introduced to help the generator produce significant difference between styles. Experimental results and a user study show that MS-CartoonGAN can generate high-quality and expressive cartoon images, outperforming state-of-the-art methods.

Acknowledgement

This work was supported by the Natural Science Foundation of China (61725204).

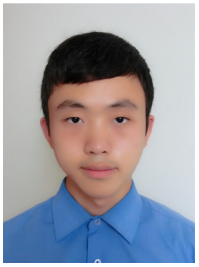
References

[1] P. L. Rosin and J. Collomosse, *Image and Video-Based Artistic Stylisation*. Springer, 2013.
[2] L. Gatys, A. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2414–2423.
[3] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *International Conference on Computer Vision*, 2017.
[4] M. Liu, T. Breuel, and J. Kautz, "Unsupervised image-to-image translation networks," in *Advances in Neural Information Processing Systems*, 2017, pp. 700–708.
[5] X. Huang, M. Liu, S. J. Belongie, and J. Kautz, "Multimodal unsupervised image-to-image translation," in *European Conference on Computer Vision (ECCV)*, 2018, pp. 179–196.
[6] A. Anoosheh, E. Agustsson, R. Timofte, and L. V. Gool, "Combogan: Unrestrained scalability for image domain translation," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 783–790.
[7] W. Cho, S. Choi, D. K. Park, I. Shin, and J. Choo, "Image-to-image translation via group-wise deep whitening-and-coloring transformation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 639–10 647.
[8] Y. Chen, Y. Lai, and Y. Liu, "CartoonGAN: Generative adversarial networks for photo cartoonization," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 9465–9474.
[9] T. Saito and T. Takahashi, "Comprehensible rendering of 3-D shapes," in *ACM SIGGRAPH*, vol. 24, no. 4, 1990, pp. 197–206.
[10] H. Winnemöller, S. C. Olsen, and B. Gooch, "Real-time video abstraction," *ACM Transactions on Graphics*, vol. 25, no. 3, pp. 1221–1226, 2006.
[11] L. Xu, C. Lu, Y. Xu, and J. Jia, "Image smoothing via L0 gradient minimization," *ACM Transactions on Graphics*, vol. 30, no. 6, p. 174, 2011.
[12] J. Wang, Y. Xu, H.-Y. Shum, and M. F. Cohen, "Video tooning," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 574–583, 2004.
[13] M. Yang, S. Lin, P. Luo, L. Lin, and H. Chao, "Semantics-driven portrait cartoon stylization," in *International Conference on Image Processing*, 2010.
[14] P. L. Rosin and Y.-K. Lai, "Non-photorealistic rendering of portraits," in *Workshop on Computational Aesthetics*, 2015, pp. 159–170.
[15] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin, "Image analogies," in *ACM SIGGRAPH*, 1998, pp. 327–340.
[16] S.-S. Huang, G.-X. Zhang, Y.-K. Lai, J. Kopf, D. Cohen-Or, and S.-M. Hu, "Parametric meta-filter modeling from a single example pair," *The Visual Computer*, vol. 30, no. 6-8, pp. 673–684, 2014.
[17] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
[18] J. Bruna, P. Sprechmann, and Y. LeCun, "Super-resolution with deep convolutional sufficient statistics," in *International Conference on Learning Representations (ICLR)*, 2016.
[19] L. A. Gatys, A. S. Ecker, and M. Bethge, "Texture synthesis and the controlled generation of natural stimuli using convolutional neural networks," *arXiv preprint arXiv:1505.07376*, 2015.
[20] L. A. Gatys, A. S. Ecker, M. Bethge, A. Hertzmann, and E. Shechtman, "Controlling perceptual factors in neural style transfer," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
[21] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.
[22] C. Li and M. Wand, "Combining Markov random fields and convolutional neural networks for image synthesis," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2479–2486.
[23] J. Liao, Y. Yao, L. Yuan, G. Hua, and S. B. Kang, "Visual attribute transfer through deep image analogy," *ACM Transactions on Graphics*, vol. 36, no. 4, p. 120, 2017.

[24] Y. Chen, Y.-K. Lai, and Y.-J. Liu, "Transforming photos to comics using convolutional neural networks," in *International Conference on Image Processing*, 2017, pp. 2010–2014.
[25] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2014, pp. 2672–2680.
[26] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum, "Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling," in *Advances in Neural Information Processing Systems*, 2016, pp. 82–90.
[27] V. Dumoulin, I. Belghazi, B. Poole, A. Lamb, M. Arjovsky, O. Mastropietro, and A. Courville, "Adversarially learned inference," in *International Conference on Learning Representations (ICLR)*, 2017.
[28] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
[29] L. Karacan, Z. Akata, A. Erdem, and E. Erdem, "Learning to generate images of outdoor scenes from attributes and semantic layouts," *arXiv preprint arXiv:1612.00215*, 2016.
[30] Y. Choi, M. Choi, M. Kim, J. Ha, S. Kim, and J. Choo, "Stargan: Unified generative adversarial networks for multi-domain image-to-image translation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 8789–8797.
[31] J. Zhang, Y. Shu, S. Xu, G. Cao, F. Zhong, M. Liu, and X. Qin, "Sparsely grouped multi-task generative adversarial networks for facial attribute manipulation," in *ACM Multimedia Conference on Multimedia Conference (MM)*, 2018, pp. 392–401.
[32] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019, pp. 4401–4410.
[33] X. Yang, D. Xie, and X. Wang, "Crossing-domain generative adversarial networks for unsupervised multi-domain image-to-image translation," in *Proceedings of the 26th ACM international conference on Multimedia*, 2018, pp. 374–382.
[34] R. Gomez, Y. Liu, M. De Nadai, D. Karatzas, B. Lepri, and N. Sebe, "Retrieval guided unsupervised multi-domain image to image translation," in *Proceedings of the 28th ACM International Conference on Multimedia*, 2020, pp. 3164–3172.
[35] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *European Conference on Computer Vision*, 2016, pp. 694–711.
[36] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *International Conference on Machine Learning*, vol. 30, no. 1, 2013.
[37] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.
[38] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 6, pp. 679–698, 1986.
[39] 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," *Advances in neural information processing systems*, vol. 30, pp. 6626–6637, 2017.

Yezhi Shu is a Ph.D student with Department of Computer Science and Technology, Tsinghua University. She received her B.Eng. degree from Shandong University, China, in 2019. Her research interests include computer vision, deep learning algorithms and applications.

Ran Yi is a Ph.D student with Department of Computer Science and Technology, Tsinghua University. She received her B.Eng. degree from Tsinghua University, China, in 2016. Her research interests include machine learning, computer vision and computer graphics.

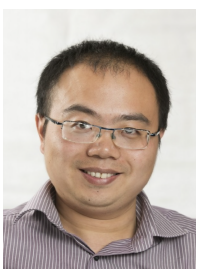


Mengfei Xia is a Ph.D student with Department of Computer Science and Technology, Tsinghua University. He received his B.S. degree from Tsinghua University, China, in 2020. His research interests include mathematic models in deep learning, computational geometry and computer vision.

Zipeng Ye is a Ph.D student with Department of Computer Science and Technology, Tsinghua University. He received his B.Eng. degree from Tsinghua University, China, in 2017. His research interests include computational geometry and computer vision.

Wang Zhao is a Ph.D student with Department of Computer Science and Technology, Tsinghua University. He received his B.Eng. degree from Tsinghua University, China, in 2019. His research interest focuses on 3D computer vision.

Yang Chen is a researcher with Tencent, China. She received her B.Eng degree from Beijing University of Posts and Telecommunications, in 2015, and her M.Eng. degree from Tsinghua University, China, in 2018. Her research interests include deep learning, image stylization and computer graphics.



Yu-Kun Lai is a Professor at School of Computer Science and Informatics, Cardiff University, UK. He received his B.S and PhD degrees in Computer Science from Tsinghua University, in 2003 and 2008 respectively. His research interests include computer graphics, geometric processing, computer vision and image processing. He is on the editorial boards of *Computer Graphics Forum* and *The Visual Computer*. For more information, visit https://users.cs.cf.ac.uk/Yukun.Lai/

Yong-Jin Liu is a Professor with the Department of Computer Science and Technology, Tsinghua University, China. He received the B.Eng. degree from Tianjin University, China, in 1998, and the PhD degree from the Hong Kong University of Science and Technology, Hong Kong, China, in 2004. His research interests include computational geometry, computer graphics and computer vision. He is a senior member of the IEEE and a member of ACM.