Image to Cartoon using CartoonGAN

Vu Minh Hien
HUST-ETE16
hien.vm224269@sis.hust.edu.vn

Le Hoang Anh
HUST-ETE16
anh.lh224297@sis.hust.edu.vn

January 2025

Abstract

In this project, we propose a solution to transforming photos of real-world scenes into cartoon style images, which is valuable and challenging in computer vision and computer graphics. Our solution belongs to learning based methods, which have recently become popular to stylize images in artistic forms such as painting. However, existing meth ods do not produce satisfactory results for cartoonization, due to the fact that cartoon styles have unique charac teristics with high level simplification and abstraction, and cartoon images tend to have clear edges, smooth color shading and relatively simple textures, which exhibit signif icant challenges for texture-descriptor-based loss functions used in existing methods.

So, we propose CartoonGAN, a generative adversarial network (GAN) framework for cartoon stylization. Our method takes unpaired photos and cartoon images for training, which is easy to use. Two novel losses suitable for cartoonization are pro posed: a semantic content loss, which is formulated as a sparse regularization in the high-level feature maps of the VGG network to cope with substantial style variation between photos and cartoons, and an edge-promoting adversarial loss for preserving clear edges. We further in troduce an initialization phase, to improve the convergence of the network to the target manifold. Our method is also much more efficient to train than existing methods. Exper imental results show that our method is able to generate high-quality cartoon images from real-world photos (i.e., following specific artists' styles and with clear edges and smooth shading) and outperforms state-of-the-art methods.

1 Introduction

Cartoons are widely used in daily life, from media publishing to children's education, and are often derived from real-world scenes. However, manually converting photos into cartoons is labor-intensive and requires significant artistic skills. Traditional image editing software is often insufficient for creating high-quality cartoonized images. Therefore, automatic techniques for transforming real-world photos into high-quality cartoon images are valuable, saving artists time and allowing them to focus on more creative tasks.

This paper introduces CartoonGAN, a novel GAN-based approach for photo cartoonization. CartoonGAN uses unpaired photo and cartoon datasets for training, simplifying the data collection process. It employs a specialized GAN architecture with two effective loss functions: semantic loss, which ensures content consistency while handling style variations between photos and cartoons, and edge-promoting adversarial loss, which preserves clear, sharp edges typical of cartoon images.

Additionally, CartoonGAN includes an initialization phase to improve convergence and training efficiency. This approach produces high-quality cartoon images that outperform existing methods and is capable of replicating the styles of specific artists. Overall, CartoonGAN provides an efficient and powerful tool for automatic cartoonization of photos, enabling high-quality results with minimal manual effort.

2 Related Work

2.1 Non-photorealistic rendering (NPR)

Many NPR algorithms have been developed, either automatically or semi-automatically, to mimic specific artistic styles including cartoons. Some works render 3D shapes in simple shading, which creates cartoon-like effect. Such techniques called cel shading can save substantial amount of time for artists and have been used in the creation of games as well as cartoon videos and movies. However, turning existing photos or videos into cartoons 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 or formulations in optimization problems. However, it is difficult to capture rich artistic styles using simple mathematical formulas. In particular, applying filtering or optimization uniformly to the entire image does not give the high-level abstraction that an artist would normally do, such as making object boundaries clear. To improve the results, alternative methods rely on segmentation of images/videos, although at the cost of requiring some user interaction. Dedicated methods have also been developed for portraits, where semantic segmentation can be derived automatically by detecting facial components. However, such methods cannot cope with general images.

2.2 Stylization with neural networks

Convolutional Neural Networks (CNNs) have received considerable attention for solving many computer vision problems. Instead of developing specific NPR algorithms which require substantial effort for each style, style transfer has been actively researched. Unlike traditional style transfer methods which require paired style/non-style images, recent studies show that the VGG network 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. first proposed a neural style transfer (NST) method based on CNNs that transfers the style from the style image to the content image. They use the feature maps of a pre-trained VGG network to represent the content and optimize the result image, such that it retains the content from the content image while matching the texture information of the style image, where the texture is described using the global Gram matrix. 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 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. 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. 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. In comparison, our method learns a cartoon style using two sets of images (i.e., real-world photos and cartoon images).

2.3 Image synthesis with GANs

An alternative, promising approach to image synthesis is to use Generative Adversarial Networks (GANs), which produce state-of-the-art results in many applications such as text to image translation, image inpainting, image super-resolution, etc. The key idea of a GAN model is to train two networks (i.e., a generator and a discriminator) iteratively, whereby the adversarial loss provided by the discriminator pushes the generated images towards the target manifold.

Several works 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 was recently proposed, which is a framework able to perform image translation with unpaired training data. To achieve this goal, it trains two sets of GAN models at the same time, mapping from class A to class B and from class B to class A, respectively. The loss is formulated based on the combined mapping that maps images to the same class. However, simultaneously training two GAN models often converges slowly, resulting in a time-consuming training process. This method also produces poor results for cartoon stylization due to the characteristics (i.e., high-level abstraction and clear edges) of cartoon images. As a comparison, our method utilizes a GAN model to learn the mapping between photo and cartoon manifolds using unpaired training data. Thanks to our dedicated loss functions, our method is able to synthesize high-quality cartoon images, and can be trained much more efficiently.

2.4 Network architectures

Many works show that although deep neural networks can potentially improve the ability to represent complex functions, they can also be difficult to train because of the notorious vanishing gradient problem. The recently introduced concept of residual blocks is a powerful choice to simplify the training process. It designs an "identity shortcut connection" which relieves the vanishing gradient issue while training. Models based on residual blocks have shown impressive performance in generative networks.

Another common way to ease the training of deep CNNs is batch normalization, which is designed to counteract the internal covariate shift and reduce the oscillations when approaching the minimum point. In addition, Leaky ReLU (LReLU)is a widely used activation function in deep CNNs for efficient gradient propagation which increases the performance of networks by allowing a small, non-zero gradient when the unit is not active. We integrate these techniques in our cartoonization deep architecture.

3 CartoonGAN

A GAN framework consists of two CNNs. One is the generator G which is trained to produce output that fools the discriminator. The other is the discriminator D which classifies whether the image is from the real target manifold or synthetic.

We formulate the process of learning to transform real-world photos into cartoon images as a mapping function which maps the photo manifold P to the cartoon manifold C. The mapping function is learned using training data $S_{\text{data}}(p) = \{p_i \mid i = 1, \ldots, N\} \subset P$ and $S_{\text{data}}(c) = \{c_i \mid i = 1, \ldots, M\} \subset C$, where N and M are the numbers of photo and cartoon images in the training set, respectively.

Like other GAN frameworks, a discriminator function D is trained for pushing G to reach its goal by distinguishing images in the cartoon manifold from other images and providing the adversarial loss for G. Let L be the loss function, G^* and D^* be the weights of the networks. Our objective is to solve the min-max problem:

$$(G^*, D^*) = \arg\min_G \max_D L(G, D)$$

We present the detail of our network architecture in Section 3.1 and propose two loss functions for G and D in Section 3.2. To further improve the network convergence, we propose an initialization phase and incorporate it into CartoonGAN, which is summarized in Section 3.3.

3.1 CartoonGAN architecture

Refer to Figure 2. In CartoonGAN, the generator net workGisusedtomapinputimagestothecartoon-manifold. Cartoon stylization is produced once the model is trained. G 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, eight residual blocks with identical layout are used to construct the content and manifold feature. We employ the residual block layout proposed in [15]. Finally, the output cartoon style images are reconstructed by

two up-convolution blocks which con tain fractionally strided convolutional layer with stride 1/2 and a final convolutional layer with 7×7 kernels.
Complementary to the generator network, the discriminator network D is used to judge whether the input image is a real cartoon image. Since judging whether an image is cartoon or not is a less demanding task, instead of a regular full-image discriminator, we use a simple patch-level discriminator with fewer parameters in D. Different from object classification, cartoon style discrimination relies on local features of the image. Accordingly, the network D 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) [23] with $=$ 0.2 is used after each normalization layer.

Figure 1: The CartoonGAN architecture includes convolutional layers with kernel size k, feature maps n, stride s, normalization (norm), and elementwise sum (ES).

3.2 Loss fuction

The loss function $L(G, D)$ in Eq. (1) consists of two components: (1) the adversarial loss $L_{adv}(G, D)$ (Section 3.2.1), which guides the generator to perform the desired manifold transformation, and (2) the content loss $L_{con}(G, D)$ (Section 3.2.2), which ensures the preservation of content during cartoon stylization. We use a simple additive form for the loss function:

$$L(G, D) = L_{adv}(G, D) + \omega L_{con}(G, D),$$

where ω is a weight used to balance the two losses. A larger ω retains more content information from the input photos, resulting in stylized images with more detailed textures. In our experiments, we set $\omega = 10$, which strikes a good balance between style and content preservation.

3.2.1 Adversarial loss $L_{adv}(G, D)$

The adversarial loss is applied to both networks G and D, which influences the cartoon transformation process in the generator network G. Its value indicates how much the output image of the generator G resembles a cartoon image. In previous GAN frameworks, the discriminator D is tasked with determining whether the input image is synthesized by the generator or from the real target manifold. However, we observe that simply training the discriminator D to differentiate between generated and true cartoon images is insufficient for transforming photos into cartoons. This is because presenting clear edges is an important characteristic of cartoon images, but the proportion of these edges is usually very small in the entire image. Therefore, an output image that lacks clearly reproduced edges but has correct shading is likely to confuse the discriminator trained with a standard loss.

To address this issue, we automatically generate a set of images $S_{data}(e) = \{e_i \mid i = 1 \dots M\} \subset E$ by removing clear edges from the training cartoon images $S_{data}(c) \subset C$, where C is the cartoon manifold and E is the manifold of cartoon-like images without clear edges. Specifically, for each image $c_i \in S_{data}(c)$, we apply the following three steps: (1) detect edge pixels using a standard Canny edge detector [**?**], (2) dilate the edge regions, and (3) apply Gaussian smoothing in the dilated edge regions.

Figure 3 shows an example of a cartoon image and its modified version with edges smoothed out. Recall that for each photo p_k in the photo manifold P, the generator G outputs a generated image $G(p_k)$. In CartoonGAN, the goal of training the discriminator D is to maximize the probability of assigning the correct label to $G(p_k)$, the cartoon images without clear edges (i.e., $e_j \in S_{data}(e)$), and the real cartoon images (i.e., $c_i \in S_{data}(c)$), such that the generator G is correctly guided to transform the input into the appropriate manifold. Therefore, we define the edge-promoting adversarial loss as:

$$L_{adv}(G, D) = \mathbb{E}_{c_i \sim S_{data}(c)}[\log D(c_i)] + \mathbb{E}_{e_j \sim S_{data}(e)}[\log(1 - D(e_j))] + \mathbb{E}_{p_k \sim S_{data}(p)}[\log(1 - D(G(p_k)))].$$

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

Figure 2: By removing clear edges in a cartoon image $c_i \in S_{\text{data}}(c)$, we generate a corresponding image $e_i \in S_{\text{data}}(e)$.

3.2.2 Content loss $L_{\text{con}}(G, D)$

In cartoon stylization, besides transforming to the correct manifolds, it is crucial to preserve the semantic content of the input photos in the resulting cartoon images. In CartoonGAN, we use high-level feature maps from the VGG network [**?**], pre-trained by [**?**], which has proven to be effective for object preservation. We define the content loss as follows:

$$L_{\text{con}}(G, D) = \mathbb{E}_{p_i \sim S_{\text{data}}(p)} \left[\| \text{VGG}l(G(p_i)) - \text{VGG}l(p_i) \|_1 \right]$$

where l refers to the feature maps of a specific VGG layer. Unlike other image generation techniques [**?**, **?**], we define our semantic content loss using ℓ_1-sparse regularization on the VGG feature maps between the input photo and the generated cartoon image. This choice is motivated by the distinct characteristics of cartoon images (i.e., clear edges and smooth shading) compared to photos. Even with an appropriate VGG layer that aims to capture image content, feature maps can be influenced by significant style differences. These differences are often concentrated in local regions where representations and regional characteristics vary dramatically. ℓ_1-sparse regularization handles these changes better than the standard ℓ_2-norm. As shown later, this is essential for reproducing the cartoon style. We use the feature maps from the 'conv4 4' layer to compute the semantic content loss.

3.3 Initialization phase

Due to the highly nonlinear nature of the GAN model, random initialization often leads to optimization being stuck in suboptimal local minima. To address this issue and improve convergence, we introduce a new initialization phase. The goal of the generator network G is to transform the input photo into a cartoon style while preserving its semantic content. In the initialization phase, we begin the adversarial learning process with a generator that only reconstructs the content of the input image. During this phase, the generator network G is pre-trained using only the semantic content loss $L_{\text{con}}(G, D)$. Figure 4 illustrates an example of the reconstructed image after 10 epochs of this initialization phase, which

already produces reasonable results. Our experiments indicate that this simple initialization phase enables CartoonGAN to converge quickly to an optimal configuration, avoiding premature convergence. A similar approach is seen in [**?**], where the content image is used to initialize the result image to improve style transfer quality.

4 Experiments

We implemented CartoonGAN using Torch [**?**] and the Lua programming language. The models trained during our experiments are available to facilitate future method evaluations. All experiments were conducted on an NVIDIA Titan Xp GPU. CartoonGAN is capable of producing high-quality cartoon stylizations by training on data from individual artists, which can be easily collected from cartoon videos, as our method does not require paired images. Different artists have unique cartoon styles, and CartoonGAN effectively learns these styles. Some results showcasing the various artistic styles generated by CartoonGAN are displayed in Figure 4.

Figure 3: For an original photo (a), the image (b) is the result after the initialization phase.

To compare CartoonGAN with state-of-the-art methods, we gathered the training and test data as described in Section 4.1. In Section 4.2, we provide a comparison of our proposed method with representative stylization techniques. Additionally, in Section 4.3, we present an ablation study to evaluate the impact of each component in the CartoonGAN model.

4.1 Data

The training dataset consists of real-world photos and cartoon images, while the test dataset includes only real-world photos. All images are resized and cropped to a resolution of 256×256 pixels. For the photo dataset, we collected 6,153 images from Flickr, of which 5,402 are used for training and the rest for testing. Regarding cartoon images, different artists exhibit unique styles when depicting real-world scenes in cartoon form. To ensure a consistent style in the cartoon images, we use key frames from animated films created by the same artist as the training data. In our experiments, 4,573 and 4,212 cartoon images from various short animated films are used to train the Makoto Shinkai and Mamoru Hosoda style models, while 3,617 and 2,302 images from the films "Spirited Away" and "Paprika" are used for the Miyazaki Hayao and "Paprika" style models.

4.2 Comparison with state of the art

We first compare CartoonGAN with two recently proposed methods in CNN-based stylization, namely NST and CycleGAN. 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 a fair comparison, we apply two adaptations of NST. In the first adaptation, we manually choose a style image that has content similar to the input photo. In the second adaptation, we extend NST to take all the cartoon images for training, similar to the comparative experiment in [38]. We also compare two versions of CycleGAN, i.e., without and with the identity loss L_{identity}. The incorporation of this loss tends to produce stylized images with better content preservation. 200 epochs were trained for both CycleGAN and our CartoonGAN.

Qualitative results are presented in Figure 4, which clearly demonstrate that NST and CycleGAN cannot handle cartoon styles well (Figures 6b-6e). In comparison, by reproducing the necessary clear edges and smooth shading while retaining the content of the input photo, our CartoonGAN model produces high-quality results (Figure 6f). More specifically, NST using only a style image may not fully learn the style, especially for areas in the target image whose content differs from the style image (Figure 4b). 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 4c).

The stylization results of CycleGAN do not capture the cartoon styles well. Without the identity loss, the output images do not preserve the content of the input photos effectively (Figure 4d). The identity loss helps alleviate this problem, but the stylization results are still far from satisfactory (Figure 4e). In comparison, CartoonGAN produces high-quality cartoonization that closely follows individual artists' styles.

Figure 4: Comparison of CartoonGAN with NST and CycleGAN for Makoto Shinkai (top) and Miyazaki Hayao (bottom) styles. Gatys (image 1) and Gatys (collection) are two adaptations of NST where a cartoon image with close content to the input photo and all the cartoon images are used for training, respectively.

Our CartoonGAN has the same property of not requiring paired images for training as CycleGAN. However, CartoonGAN requires much less training time. For each epoch, CycleGAN and CycleGAN with L_{identity} take 2291.77s and 3020.31s, respectively, whereas CartoonGAN only takes 1517.69s, about half compared to CycleGAN with L_{identity}. This is because CycleGAN needs to train two GAN models for bidirectional mappings, which significantly slows down the training process. For image cartoonization, mapping back from cartoons to photos is not necessary. By using the VGG feature maps rather than a cycle architecture to constrain the content, CartoonGAN learns cartoon stylization

more efficiently.

We also compare our method with CNNMRF and Deep Analogy, with Paprika and Mamoru Hosoda styles in Figure 5. Since both methods expect a single style image, two strategies are used to choose the style image from the training set: a manually selected cartoon image most similar in content with the input photo (image1) and a randomly picked cartoon image (image2). These methods fail to reproduce the characteristics of cartoon styles and produce results with clear artifacts, whereas our method produces high-quality stylization.

Figure 5: Cartoonization with 'Paprika' and Mamoru Hosoda styles, compared with CNNMRF and Deep Analogy.

4.3 Roles of components in loss function

We perform an ablation experiment to investigate the contribution of each component in CartoonGAN. Figure 6 presents examples of ablations of our complete loss function, with all results trained using Makoto Shinkai's style data. The following observations show that each component is essential for the performance of CartoonGAN. First, the initialization phase allows the generator G to rapidly converge to a reasonable manifold. As illustrated in Figure 6b, without initialization, although some key features are present, the styles fall short of expectations. Second, even with an appropriate VGG layer, significant and often localized differences in feature maps between the input and cartoon style images persist due to the vast style differences. Using l1 sparse regularization (instead of l2) on high-level VGG feature maps helps manage these substantial style discrepancies between cartoon images and photos. Lastly, the carefully crafted edge loss directs the generator G to produce clear edges in the results, resulting in better cartoon-style images.

Figure 6: Cartoonization with 'Paprika' and Mamoru Hosoda styles, compared with CNNMRF and Deep Analogy.

5 Conclusion and Future Work

In this project, we introduce CartoonGAN, a Generative Adversarial Network designed to transform real-world photos into high-quality cartoon images. The goal is to replicate the distinctive features of cartoon images, for which we propose (1) a new edge-promoting adversarial loss that enhances clear edges, and (2) an $\ell 1$ sparse regularization of high-level VGG feature maps for content loss, which ensures enough flexibility to generate smooth shading. Additionally, we present a simple yet effective initialization phase that aids in improving convergence. Experimental results show that CartoonGAN successfully learns a model to convert real-world photos into cartoon-style images with excellent quality and efficiency, significantly outperforming current state-of-the-art stylization methods.

In future work, due to the importance of portrait images, we aim to explore how to leverage local facial features to improve cartoon stylization for human faces. While our loss functions are tailored to address the specific challenges of cartoon stylization, the underlying concepts can also be applied to other image synthesis tasks, which we plan to investigate further. Furthermore, we intend to introduce sequential constraints into the training process to extend our method to video handling.

6 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-toimage 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.