CartoonGAN: Transforming Real-Life Images into Cartoon Styles

First Author
Vu Minnh Dung
20224279

Second Author
Nguyen Duy Anh
20224278

Abstract

This paper presents an implementation of CartoonGAN for transforming real-life images into high-quality cartoon-style visuals. By training on a diverse dataset, the model preserves content fidelity while applying vibrant colors and sharp edges, automating the cartoonization process. The results highlight CartoonGAN's effectiveness for creative applications, bridging real-world imagery with artistic styles.

1. Introduction

1.1. Proplem

Transforming real-life images into cartoon-style representations remains a challenging task in image processing. This project leverages Generative Adversarial Networks (GANs) to automatically convert real-world images into high-quality cartoons, preserving key features and providing valuable applications in animation and design.

1.2. Input/Output

This is example of real life image

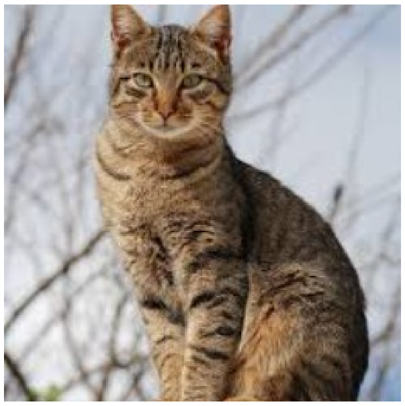

Figure 1. Real life image.

and the output image after applying cartoon style

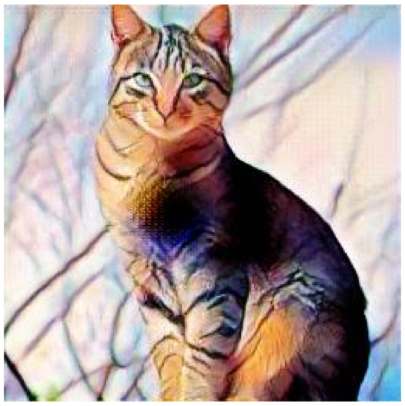

Figure 2. Image with cartoon style.

1.3. Challenge

Crawling high-quality, diverse, and relevant data posed challenges, requiring careful preprocessing to address noise and inconsistencies. Training was equally demanding, involving hyperparameter tuning, avoiding overfitting, and managing large datasets efficiently within computational constraints

2. Related works

2.1. Non-photorealistic rendering (NPR)

Non-Photorealistic Rendering (NPR) [2] algorithms aim to replicate artistic styles, including cartoons, through automatic or semi-automatic methods. Techniques like cel shading simplify 3D shapes into flat shading, creating cartoon-like effects often used in games and animations. However, transforming existing photos or videos into cartoons is more complex.

Flat shading methods employ image filtering or optimization, but these struggle to capture rich artistic styles or high-level abstractions, such as enhancing object boundaries. Segmentation-based approaches improve results but may require user interaction. Specialized methods for portraits use semantic segmentation to detect facial features automatically, yet they are unsuitable for general

2.2. Image synthesis with GANs

Generative Adversarial Networks (GANs) [1] have shown great promise in image synthesis, excelling in tasks like text-to-image translation, image inpainting, and super-resolution. GANs consist of a generator and discriminator trained iteratively, with the adversarial loss encouraging the generator to produce realistic images.

Traditional GAN-based methods for image synthesis require paired datasets, which are impractical for stylization tasks. CycleGAN addresses this limitation by enabling image translation using unpaired data, training two GAN models to map images between two classes. However, CycleGAN's training is slow and struggles with cartoon stylization due to the high-level abstraction and clear edges in cartoon images.

The proposed method overcomes these challenges by using a GAN model specifically designed for photo-to-cartoon translation with unpaired data. It incorporates dedicated loss functions to synthesize high-quality cartoon images efficiently..

3. Proposed methods - CartoonGAN

3.1. CartoonGAN architecture

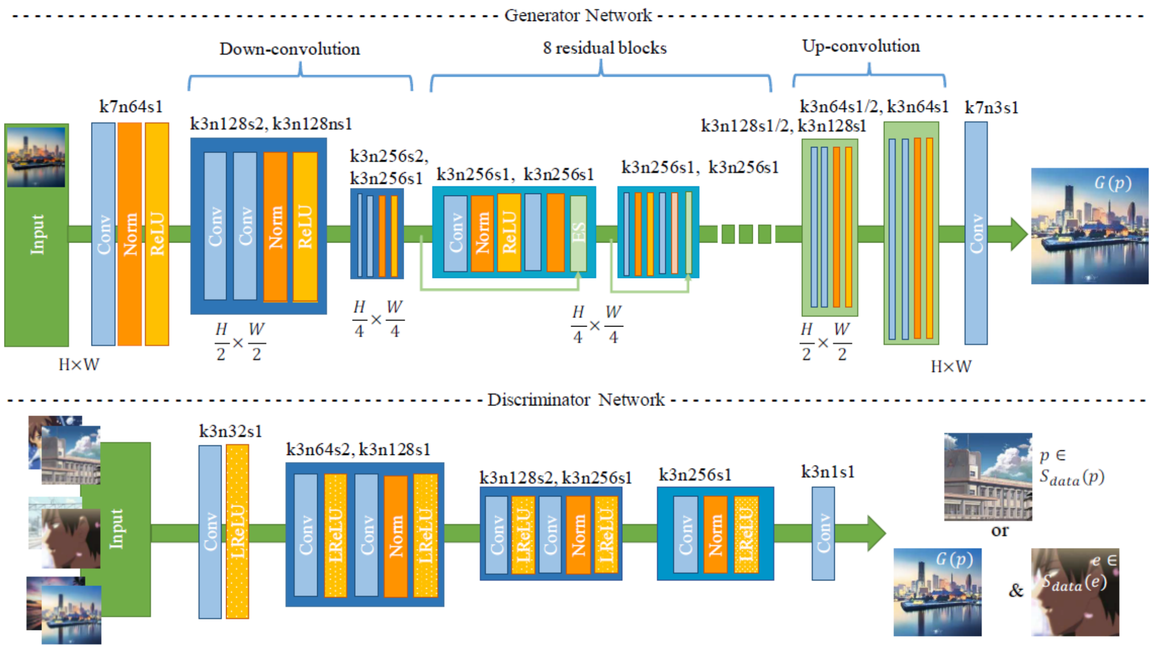

Figure 3. Architecture of the generator and discriminator networks in the proposed CartoonGAN.

Refer to figure 3, the generator network G transforms input images into cartoon-style images. It begins with a convolutional layer, followed by two down-convolution blocks that compress and encode the images to extract local features. Eight identical residual blocks then construct content and manifold features, based on the layout from [15]. Finally, two up-convolution blocks and a 7×7 convolutional layer reconstruct the output in cartoon style.

The discriminator network D evaluates whether an image is a real cartoon. To reduce complexity, a patch-level discriminator with fewer parameters is used. It focuses on local features, making it shallower than typical full-image discriminators. D starts with flat layers, followed by two strided convolutional blocks to downsample and encode local features. A feature construction block and a 3×3 convolutional layer then produce the classification output. Leaky ReLU (LReLU) with a = 0.2 is applied after each normalization layer.

3.2. Loss function

The loss function L(G,D) comprises two components: adversarial loss Ladv(G,D), , and content loss Lcon(G,D), . A weight w balances these losses, with larger w retaining more content details. For experiments, w is set to 0,00005, striking a balance between stylization and content preservation.

$$L(G, D) = Ladv(G, D) + wLcon(G, D), \qquad (1)$$

3.3. Adversarial loss Ladv(G,D)

CartoonGAN's adversarial loss trains the generator (G) to create cartoon-like images and the discriminator (D) to classify real cartoon images, edge-smoothed cartoon images, and generated images. Since cartoon edges are key but sparse, the discriminator also learns from edge-removed versions of cartoon images created through edge detection, dilation, and smoothing. This edge-promoting loss guides the generator to produce images with clear cartoon characteristics.S

$$\begin{aligned} L_{\{adv\}(G,D)} = {} & E_{\{c_i \sim S_{\{data\}(c)}\}}[\backslash log\, D(c_i)] \\ & + E_{\{e_j \sim S_{\{data\}(e)}\}}\left[\backslash log\left(1 - D(e_j)\right)\right] \\ & + E_{\{p_k \sim S_{\{data\}(p)}\}}[\backslash log(1 - D(p_k))] \end{aligned}$$

3.4. Content loss Lcon(G,D)

In addition to transformation between correct manifolds, one more important goal in cartoon stylization is to ensure the resulting cartoon images retain semantic content of the input photos. In CartoonGAN, we adopt the high-level feature maps in the VGG network pre-trained by, which has been demonstrated to have good object preservation ability. Accordingly, we define the content loss as:

$$L_{\{con\}(G,D)} = E_{\{p_i \sim S_{\{data\}(p)}\}}\left[|\, VGG_{l(G(p_i))} - VGG_{l(p_i)|_1}\right]$$

3.5. Initialization phase

The GAN model's nonlinearity and random initialization can lead to suboptimal convergence. To address this, a new initialization phase is proposed. Initially, the generator network G is trained only with the semantic content loss Lcon(G,D), focusing on reconstructing the input photo's content in a cartoon style. After 20 epochs, the generator produces a reasonable reconstruction. This initialization phase aids CartoonGAN in converging quickly to a good solution without premature convergence. Similar methods have been shown to improve style transfer quality in other studies.

4. Experiments

4.1. Data

The training data includes 4,000 real-world photos from cocodataset.org , with 80% used for training and the rest for testing. The images are resized and cropped to 256×256. For cartoon images, image are taken from many anime japanese film in safebooru.org. In the experiments, 4,000 cartoon images from are used to train their respective style models.

4.2. Implement

To make the GAN better learn to produce clear edges in the cartoon image, the model is trained with an edge-smoothed version on every cartoon image, too. In the paper, the edges are first detected by canny-edge, then the edges are dilated and smoothed with gaussian smoothing. In my implementation, I do the canny edges, the dilation and the gaussian blur with openCV and I make the white backbground transparent and paste the edges back on the original image with Pillow.

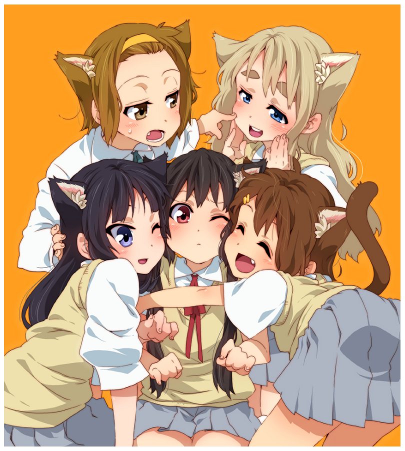

Figure 4. cartoon image.

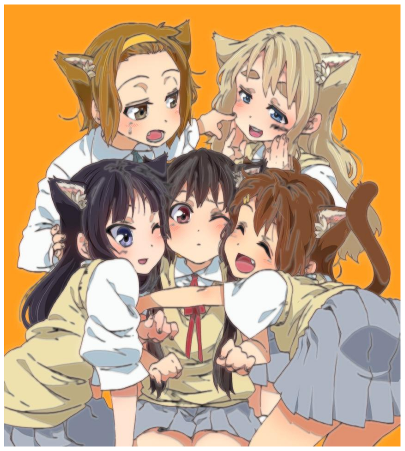

Figure 5. smoothed image.

An ablation experiment was conducted to examine the role of each component in CartoonGAN. Results, demonstrate the importance of each part. First, the initialization phase helps the generator quickly converge to a reasonable manifold, as seen in, where styles without initialization are far from expected. Second, the VGG feature maps help address significant style differences between cartoon images and photos. Finally, the edge loss guides the generator to produce clear edges, enhancing the cartoon style of the output images.

The Generator aims to create images that look as close to real images as possible, making it difficult for the Discriminator to distinguish between real and generated images. The Generator receives real images and learns to produce outputs that are more realistic, minimizing the Discriminator's ability to detect them. The Generator's loss functions push it to reduce the gap between the generated images and the real images, leading to improved image quality over time.

The Discriminator's role is to distinguish between real and fake images. It receives three types of inputs: real images, generated images from the Generator, and smoothed images from the Generator. The Discriminator calculates losses based on these inputs, trying to maximize its ability to correctly classify real from fake. As the Generator improves, the Discriminator adjusts its decision boundaries, continuously learning to better differentiate between real and fake images.

The Generator creates realistic images, while the Discriminator tries to distinguish real from fake. They support each other by improving their performance through a feedback loop—when the Generator generates better images, the Discriminator becomes better at identifying fake ones. Training both together ensures the Generator produces high-quality images, and the Discriminator accurately detects them.

4.3. Result

The generator loss loss decreases rapidly in the initial epochs, reaching around 5 around 20–30 epochs

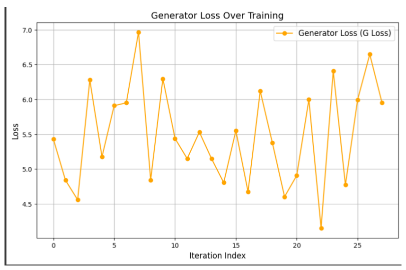

Figure 6. Generator loss

In Fig 6, the loss maintain around 5.5 from epoch 92 to 101, A decreasing generator loss in CartoonGAN generally indicates that the generator is improving, producing outputs closer to the desired cartoon style, and better fooling the discriminator

While the images exhibit a strong cartoon style, there are still areas that could be improved for even better results.

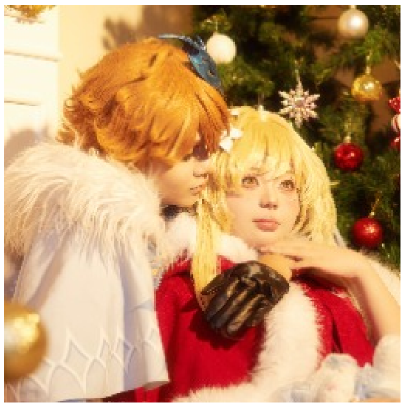

Figure 7. Input image.

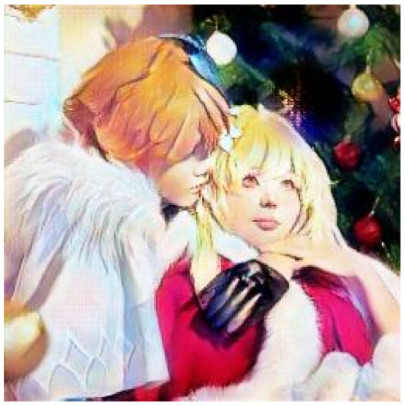

Figure 8. Output image.

References

[1] 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 27*, pages 2672–2680, 2014. 2

[2] P. L. Rosin and J. Collomosse. *Image and Video-Based Artistic Stylisation*. Springer, 2013. 1