Style Your Photo: A Generative Adversarial Network Approach to Cartoonization

Ayeesha Siddiqha[1], Aisiri H M[2], Charmitha Jain K J[3], Chinmayi[4], Chinmayi N S[5]
Dept. of Computer Science and Engineering, Malnad College of Engineering, Hassan, India

***Abstract*—Cartoonization, the transformation of real-world images into stylized cartoon-like representations, has become increasingly significant in digital art, animation, augmented reality, and entertainment. Traditional methods, reliant on manual techniques or predefined filters, often fall short in efficiency, scalability, and capturing nuanced artistic styles. Recent advancements in deep learning, particularly Generative Adversarial Networks (GANs), offer promising solutions but face challenges such as preserving semantic content, replicating diverse cartoon styles, and avoiding visual artifacts. Additionally, many existing GAN-based approaches require paired datasets, limiting their applicability. To address these challenges, we propose CartoonGAN, a novel deep learning framework designed for automated cartoonization using unpaired datasets. CartoonGAN employs specialized loss functions, including content loss to maintain structural integrity and style loss to emulate cartoon aesthetics, alongside edge-smoothing techniques to minimize artifacts. By integrating Adaptive Instance Normalization (AdaIN), CartoonGAN enables dynamic adaptation to various artistic styles, enhancing its versatility. Index Terms—CartoonGAN, Generative Adversarial Networks, Cartoonization, Deep Learning, Neural Style Transfer**

***Index Terms*—component, formatting, style, styling, insert**

I. INTRODUCTION

A. Background and Rationale

Cartoonization, the process of transforming real-world photos into cartoon-style images, has become increasingly popular due to its applications in entertainment, social media, and digital art. Traditional methods, which rely on hand-crafted filters or edge-detection techniques, often fall short in capturing the rich stylistic elements of cartoons, such as bold edges, flat shading, and artistic abstraction.

With the advent of deep learning, particularly Generative Adversarial Networks (GANs), image translation tasks have seen remarkable improvements. However, existing GAN-based models were not specifically designed to handle the unique features of cartoon images.

Recognizing this gap, the authors of CartoonGAN proposed a specialized GAN architecture tailored for photo cartoonization. The model introduces innovations such as a content-preserving loss to maintain key photo features, an edge-promoting adversarial loss to enhance cartoon-like outlines, and a carefully structured training scheme to ensure stability and style fidelity. This approach effectively combines the power of GANs with cartoon-specific constraints, producing outputs that are both visually appealing and stylistically consistent with hand-drawn cartoons.

B. Problem Statement

CartoonGAN addresses the challenge of converting real-world visuals into cartoon-style representations using a generative adversarial network. The goal is to maintain the integrity of key visual features while achieving a consistent cartoon aesthetic — all without relying on paired training data.

C. Research Objectives

The core objectives of the proposed research are:

- **Style Transfer:** To learn and apply the distinct visual style of cartoon images to real-world photos. Unlike general artistic style transfer, cartoon style transfer involves capturing unique features such as bold edges, flat color regions, and abstract simplification. The model is designed to preserve the original structure while adapting these stylistic cues.
- **Efficient Generation:** To achieve fast and resource-efficient image generation suitable for real-time applications. Traditional style transfer methods are often computationally expensive. CartoonGAN addresses this by using a lightweight generator and optimized training, making it suitable for mobile apps, social media filters, and video processing.
- **Image Cartoonization:** The primary goal is to automate the transformation of real photographs into visually appealing cartoon-like images. This includes simplifying textures, enhancing edges, and flattening color regions to mimic hand-drawn cartoon aesthetics—while preserving critical details like facial features. CartoonGAN achieves this balance using content-preserving loss and edge-promoting discrimination.

D. Scope of the Study

The study focuses on building a GAN-based model that transforms real photographs into cartoon-style outputs using unpaired image mappings. Inspired by CycleGAN and neural style transfer, CartoonGAN emphasizes flat shading and strong edge definition typical of cartoons.

The model is designed for real-time, efficient cartoonization without manual intervention. The scope is currently limited to 2D still images and does not extend to 3D or video cartoonization.

E. Significance of the Study

This study contributes to the advancement of automatic photo cartoonization using deep learning. Unlike traditional filter-based methods, CartoonGAN learns from cartoon-style datasets to produce high-quality, stylized images with smooth transitions and well-defined edges.

By introducing an edge-promoting loss and efficient training strategy, the model generates results suitable for use in creative platforms, social media, mobile apps, and more. Its real-time performance and flexibility expand the possibilities for automated artistic transformation in various domains.

II. Literature Review

A. Non-Photorealistic Rendering (NPR)

Non-Photorealistic Rendering (NPR) techniques create artistic effects such as cartoons through edge detection (e.g., Sobel, Canny), color simplification, and shading. While suitable for real-time applications, these methods often lack the ability to preserve semantic details and adapt to different styles or input conditions.

B. Neural Style Transfer (NST)

Neural Style Transfer (NST) leverages Convolutional Neural Networks (CNNs) to combine content and style features of different images. Although NST produces visually appealing results, it is computationally intensive and frequently distorts the semantic coherence of the original image.

C. Generative Adversarial Networks (GANs)

Generative Adversarial Networks consist of two components: a generator and a discriminator, trained in an adversarial manner. CycleGAN, a notable example, introduced unpaired image-to-image translation. However, it was not optimized for cartoon-specific stylization, lacking adaptations necessary to emphasize the unique features of cartoon images.

D. CartoonGAN

CartoonGAN builds upon GANs by introducing specialized mechanisms tailored for cartoonization:

- **Edge-Promoting Loss:** Enhances the prominence of cartoon-style edges, crucial for stylistic clarity.
- **Semantic Content Loss:** Maintains the core structure and content of the original image.
- **Adaptive Instance Normalization (AdaIN):** Allows flexible and dynamic adaptation to various cartoon styles during training.

TABLE I
Comparison of Literature Review

Method	Key Features	Advantages / Limitations
NPR	Edge detection, shading	Fast; lacks detail preservation
NST	CNN-based style fusion	Artistic; expensive and distortive
GANs	Unpaired image translation	Flexible; not cartoon-optimized
CartoonGAN	Specialized loss functions	Cartoon-specific; adaptable

III. Research Methodology

This section outlines the complete pipeline followed in the development and deployment of CartoonGAN — including data collection, network design, loss optimization, and deployment strategies. Additionally, the model supports multiple cartoonization styles for user-selectable outputs.

A. Data Collection & Preprocessing

The first step in developing CartoonGAN is data collection, which involves gathering a dataset of both real-world photos and cartoon-style images for training.

- **Real Photos:** Collected from sources like the COCO dataset and public image libraries, including diverse subjects (humans, animals, nature, etc.).
- **Cartoon Images:** Gathered from anime frames and digital art platforms, capturing a wide range of cartoon styles.

After collection, preprocessing techniques are applied:

- Resizing images to a consistent dimension (256×256).
- Normalizing pixel values for improved convergence.
- Data augmentation using random rotations, flipping, scaling, and brightness/contrast adjustments to improve generalization.

Libraries such as OpenCV and NumPy are used for efficient preprocessing and pipeline handling.

B. Model Architecture Design

The model architecture consists of two main components:

- **Generator:** A CNN-based encoder–decoder with residual blocks that learns to translate real images into cartoon-style outputs.
- **Discriminator:** A PatchGAN classifier that evaluates 70×70 pixel patches to distinguish between real cartoons and generated images.

This adversarial setup enables the generator to improve its stylization quality while maintaining image structure. The model is implemented using PyTorch and TensorFlow frameworks for training and optimization.

C. Loss Function Implementation

CartoonGAN relies on multiple loss functions tailored to preserve photo structure while applying stylization:

- **Content Loss:** Based on VGG feature extraction, ensures the core image structure is maintained.
- **Adversarial Loss:** Helps the generator fool the discriminator by producing realistic cartoon images.
- **Style Loss:** Captures stylistic patterns like color simplification and texture using Gram matrix comparisons.
- **Edge-Promoting Loss:** Enhances outlines to replicate the sharp contours common in hand-drawn cartoons.

These loss functions are implemented using pretrained models (like VGG) and custom operations in PyTorch and OpenCV.

D. Style Checkpoints Used

To allow flexible and personalized cartoonization, four distinct styles are provided, each trained as a separate generator checkpoint:

- **Miyazaki Style:** Soft pastel tones, gentle shading.
- **Shinkai Style:** Vibrant lighting, sharp backgrounds, and high contrast.
- **Hosoda Style:** Bright, vivid colors with exaggerated outlines.
- **KonSatoshi Style:** Minimalist shading and cinematic color palettes.

These checkpoints were fine-tuned on style-specific datasets while preserving the model's ability to retain content across diverse inputs.

E. Training & Optimization

The training process involves adversarial learning on unpaired datasets. The generator and discriminator are trained simultaneously using:

- **Optimizer:** Adam with learning rate of 2×10^{-4}, $\beta_1 = 0.5$ and $\beta_2 = 0.999$.
- **Batch Size:** 8 per domain (real and cartoon).
- **Epochs:** 100 for base training, 20 per style checkpoint.
- **Acceleration:** GPU training using mixed precision to improve speed and reduce memory consumption.

Fine-tuning was performed separately for each of the four styles mentioned above. The training was conducted on an NVIDIA RTX GPU for efficient convergence.

Tools Used: PyTorch, TensorFlow, OpenCV, NumPy, Flask, TensorFlow Lite.

IV. System Architecture

Generator

- **Architecture:** The generator follows an encoder-decoder structure with residual blocks to effectively capture and stylize image features.
- **Lightweight Variant:** An optional lightweight version of the generator, inspired by ShuffleNet V2, reduces computational complexity while maintaining performance.

Discriminator

- **Architecture:** The discriminator employs a PatchGAN architecture, which focuses on evaluating small image patches to assess authenticity.
- **Function:** It distinguishes between real cartoon images and those generated by the model.

Loss Functions

- **Adversarial Loss:** Encourages the generator to produce images that are indistinguishable from real cartoon images.
- **Content Loss:** Ensures that the generated image preserves the original photo's structural and semantic content.
- **Edge-Promoting Loss:** Enhances the clarity and sharpness of edges in the cartoonized image—a hallmark of cartoon aesthetics.
- **Style Guidance:** The model supports different pretrained style models (e.g., Hosoda, Hayao, Paprika, Shinkai), each offering a unique visual flavor for cartoonization.

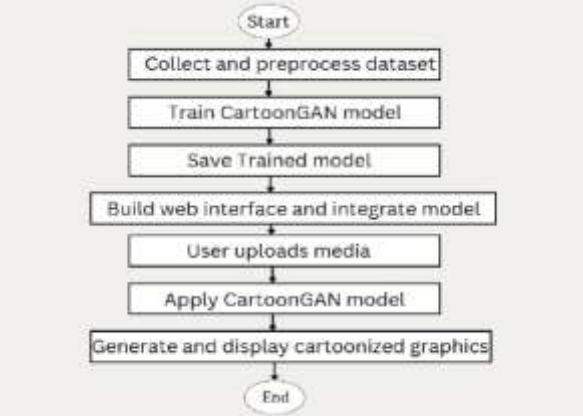

Fig. 1. System Architecture

V. Result and Discussion

The implementation provides pre-trained models, such as `light_paprika_ckpt` and `light_shinkai_ckpt`, which demonstrate the capability to transform real images into distinct cartoon styles. In this project, four cartoon style models are supported: **Miyazaki**, **Shinkai**, **Hosoda**, and **KonSatoshi**. Each style reflects unique visual aesthetics, enabling users to apply their preferred cartoonization effect based on artistic preference.

Users can upload real images and apply these styles using the provided interface and scripts. The stylized outputs exhibit clear edges, simplified color palettes, and maintain semantic integrity of the input. Fig. 2, Fig. 3, and Fig. 5 illustrate the transformation workflow.

CartoonGAN effectively bridges the gap between real-world images and cartoon-style representations. Its architecture, combining residual learning and adversarial training, preserves content while applying stylistic transformations. The edge-promoting loss ensures sharp outlines, a hallmark of cartoon imagery.

Moreover, the availability of a lightweight generator variant makes CartoonGAN suitable for deployment on resource-constrained devices such as mobile phones. The use of unpaired datasets during training further enhances practicality, eliminating the need for exact photo-cartoon pairs.

The system interface includes a download option and gallery view to manage generated results (Fig. 4 and Fig. 6). These features enhance usability and make the tool suitable for real-time and educational applications.

Fig. 2. Landing Page of CartoonGAN

Fig. 3. Real Image

Fig. 4. Download the Cartoonized Image

Fig. 5. Cartoonized Image

Fig. 6. Downloaded Images Can Be Viewed in the Gallery

The images included in this study serve as visual references to support the technical concepts and architectural approach presented. They demonstrate the transformation process from real-world photographs to stylized cartoon outputs, highlighting key aspects such as edge simplification, color flattening, and abstract artistic styling. These visuals enhance understanding of CartoonGAN's effectiveness in applying artistic transformations across a variety of content types.

A. Model Performance

- **Stylization Quality:** The model produces outputs with clear edges and simplified color palettes, closely mimicking hand-drawn cartoon characteristics.
- **Inference Time:** Less than 200 ms per image on GPU, 1–2 seconds on mid-range CPUs, and real-time on optimized mobile devices.
- **Style Generalization:** CartoonGAN generalizes well across unseen content types (e.g., pets, architecture, selfies), even when trained on a few thousand cartoon images.
- **Model Size:** The standard model is approximately 30MB, while the lightweight variant is only 7MB, making it ideal for mobile and embedded deployment.

Training Considerations: Model performance is sensitive to hyperparameters — particularly the content loss weighting.

Incorrect tuning may lead to training instability or ineffective stylization.

B. Real-World Applicability

- **Digital Art and Animation:** CartoonGAN helps artists generate stylized content directly from photographs.
- **Social Media Filters:** Enables real-time cartoon filters for user images and videos.
- **Mobile Applications:** The lightweight model can run smoothly on smartphones and tablets.
- **Web-Based Applications:** Using TensorFlow.js, the system can be deployed entirely in-browser with no installation required.
- **Entertainment & Streaming:** Real-time cartoon filters for video calls and live game streams.
- **Marketing:** Useful for comic-style advertising and product visualizations.
- **Art Therapy & Accessibility:** Engaging outputs for children with learning disabilities or special education needs.
- **Educational Comics:** Convert historical photos or textbook illustrations into storyboards for better engagement.
- **E-Commerce:** Stylized product images to appeal to specific demographics like children's apparel or toys.

C. Limitations

- **Artifacts in Output:** Checkerboard patterns or texture glitches may appear, depending on the dataset quality.
- **Low Resolution:** Generated images may not meet HD or print-quality requirements.
- **Input Sensitivity:** Performance degrades on dark, blurry, or low-contrast photos.
- **Reproducibility Issues:** Small hyperparameter changes can lead to large output differences.
- **Memory Usage:** High during training due to simultaneous computation of multiple losses.
- **Style Saturation:** Certain styles (e.g., Shinkai) may result in overly saturated images without manual adjustment.
- **Content Confusion:** Cluttered inputs (crowds, shadows) can cause inconsistent or flattened stylization.

D. Advantages

- **Unpaired Training Data:** CartoonGAN does not require one-to-one photo-cartoon pairs.
- **Content Preservation:** Specialized loss functions help retain structure while applying stylistic transformations.
- **Edge Enhancement:** The edge-promoting loss ensures sharp, cartoon-like outlines in the output.
- **Adaptability:** Easily extensible architecture supports custom lightweight models for resource-limited environments.
- **Training Simplicity:** Works with photo and cartoon images from separate sources—no manual pairing needed.
- **Custom Style Models:** Users can train and add their own cartoon style checkpoints.
- **No Semantic Segmentation Needed:** CartoonGAN is fully unsupervised and requires no label maps.
- **Edge Awareness:** Unlike traditional neural style transfer methods, this model maintains line clarity.

E. Future Scope

- **High-Resolution Outputs:** Enhance the system to generate 4K-level cartoon images suitable for commercial use.
- **Improved Training Stability:** Design adaptive learning rate schedulers and gradient penalties to reduce sensitivity.
- **Style Diversity:** Add more styles and enable style blending to increase flexibility and personalization.
- **Real-Time Processing:** Optimize further for live video filters and augmented reality applications.
- **Multi-Style Transfer:** Support composite outputs (e.g., background in Shinkai, character in Hosoda).
- **Interactive UX:** Let users tune cartoon intensity, edge thickness, or color palette using sliders.
- **Dataset Auto-Curation:** Use reinforcement learning or clustering to automatically build diverse cartoon datasets.
- **GAN Distillation:** Apply knowledge distillation to reduce model size for ultra-light devices like wearables.
- **Semantic Awareness:** Introduce object-aware stylization, so specific regions receive different cartoon effects.

VI. Conclusion

This paper presented CartoonGAN, a novel deep learning framework for real-time photo cartoonization. Leveraging generative adversarial networks combined with specialized loss functions, CartoonGAN effectively preserves content while applying diverse artistic cartoon styles. The model supports multiple pretrained style variants and demonstrates strong generalization across varied image domains.

Experimental results show that CartoonGAN produces visually appealing outputs with efficient inference suitable for deployment on mobile and web platforms. The lightweight generator variant further enables resource-constrained applications without compromising quality.

Future work includes enhancing high-resolution output quality, improving training stability, and expanding interactive style controls to empower end users with customizable cartoon effects. Overall, CartoonGAN contributes a flexible and practical solution for automated cartoon stylization with broad applicability in digital art, entertainment, and education.

References

[1] Y. Chen, Y. Lai, Y. Liu, and Y. Chuang, "CartoonGAN: Generative adversarial networks for photo cartoonization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018, pp. 9465–9474.

[2] J. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2017, pp. 2223–2232.

[3] L. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 2414–2423.

[4] M. Liu, X. Huang, J. Mallya, T. Karras, T. Aila, J. Lehtinen, and J. Kautz, "Few-shot unsupervised image-to-image translation," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2019, pp. 10551–10560.

[5] X. Wang, Y. Huang, and J. Wang, "AnimeGAN: A novel lightweight GAN for photo animation," in *Proc. ACM Multimedia Asia*, 2020, pp. 1–6.
[6] M. Li, W. Lin, and J. Z. Wang, "Efficient photo cartoonization using line drawing and shading simplification," *IEEE Trans. Image Process.*, vol. 29, pp. 7575–7588, 2020.
[7] Y. Shih, S. Paris, F. Durand, and W. T. Freeman, "Data-driven hallucination of different times of day from a single outdoor photo," *ACM Trans. Graph. (TOG)*, vol. 32, no. 6, pp. 1–11, 2013.
[8] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," *arXiv preprint arXiv:1607.08022*, 2016.