A Project Report
on
Cartoonify the image

Submitted in partial fulfillment of the requirement for the award of the degree of

GALGOTIAS UNIVERSITY
(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Under The Supervision of
Mrs. Suman Devi
Assistant Professor

Submitted By
PRACHI SINGH
19SCSE1010537

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GALGOTIAS UNIVERSITY, GREATER NOIDA,
INDIA

DECEMBER, 2021

CANDIDATE’S DECLARATION

We hereby certify that the work which is being presented in the project, entitled “Cartoonify the image” in partial fulfillment of the requirements for the award of the B.Tech(CSE) submitted in the School of Computing Science and Engineering of Galgotias University, Greater Noida, is an original work carried out during the period of JULY-2021 to DECEMBER-2021 and Year, under the supervision of Mrs. Suman Devi, Assistant Professor, Department of Computer Science and Engineering, of School of Computing Science and Engineering, Galgotias University, Greater Noida.

The matter presented in the thesis project has not been submitted by me/us for the award of any other degree of this or any other places.

PRACHI SINGH
19SCSE1010537

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Mrs. Suman Devi
Assistant Professor

CERTIFICATE

The Final Project Viva-Voce examination of PRACHI SINGH 19SCSE1010537 has been held on ______________________ and his/her work is recommended for the award of Department Of Computer Science and Engineering.

Signature of Examiner(s) **Signature of Supervisor(s)**

Signature of Project Coordinator **Signature of Dean**

Date: December 2021

Place: Greater Noida

ACKNOWLEDGEMENT

I thank the almighty for giving us the courage and perseverance in completing the main project. This project itself is acknowledgment to all those people who have given us their heartfelt co-operation in making this project a grand success.

I am greatly indebted to project guide Mrs. Suman Devi, Assistant Professor, Computer Science, and engineering, for providing valuable guidance at every stage of this project work. I am profoundly grateful for the unmatched services rendered by him.

Our special thanks to all the faculty of Computer Science and Engineering and peers for their valuable advises at every stage of this work.

ABSTRACT

In this paper, we propose a solution to transforming pho- tos 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 methods do not produce satisfactory results for cartoonization, due to the fact that (1) cartoon styles have unique charac- teristics with high level simplification and abstraction, and(2) 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. In this paper, we propose Car- toonGAN, a generative adversarial network (GAN) frame- work 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: (1) 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 (2) 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.

Table of Contents

List of Figures

1. INTRODUCTION

Cartoons are an artistic form widely used in our daily life. In addition to artistic interests, their applications range from publication in printed media to storytelling for children's education. Like other forms of artworks, many famous cartoon images were created based on real-world scenes.

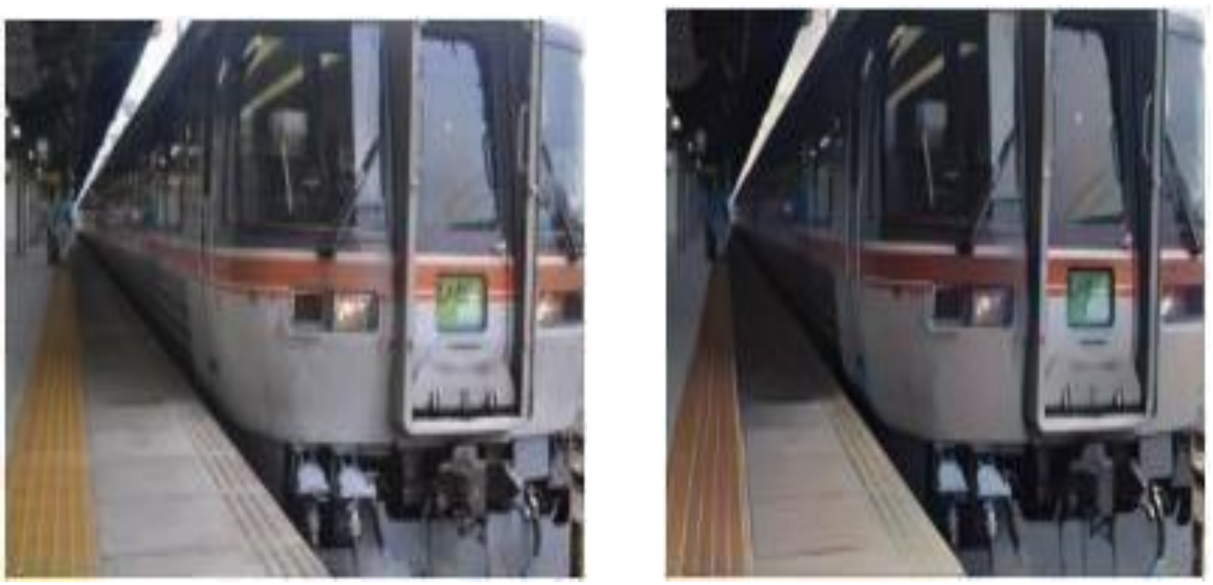


Figure1

Figure1 . shows a real-world scene whose cor-responding cartoon image appeared in the animated film "Your Name". However, manually recreating real- world scenes in cartoon styles is very laborious and involves substantial artistic skills. To obtain high-quality cartoons, artists have to draw every single line and shade each color region of target scenes. Meanwhile, existing image editing software/algorithms with standard features cannot produce satisfactory results for cartoonization. Therefore, specially designed techniques that can automatically transform real- world photos to high-quality cartoon style images are very helpful and for artists, tremendous amount of time can be saved so that they can focus on more creative work. Such tools also provide a useful addition to photo editing soft- ware such as Instagram and Photoshop. Stylizing images in an artistic manner has been widely studied in the domain of non-photorealistic rendering .
Traditional approaches develop dedicated algorithms for specific styles.However, substantial efforts are required to produce fine-grained styles that mimic individual artists. Recently, learning-based style transfer methods in which an image can be stylized based on provided ex- amples, have drawn considerable attention. In

particular, the power of Generative Adversarial Networks (GANs) 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. Although significant success has been achieved with learning based stylization, state-of-the-art methods 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 — clear edges, smooth color shading and relatively simple textures — which is very different from other forms of artworks. In this paper, we propose CartoonGAN, a novel GAN- based approach to photo cartoonization. Our method 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, we do *not* require pairing or correspondence between two sets of images. From the perspective of computer vision algorithms, the goal of cartoon stylization is to map images in the photo manifold into the cartoon mani- fold while keeping the content unchanged. To achieve this goal, we propose to use a dedicated GAN- based architec- ture together with two simple yet effective loss functions. The main contributions of this paper are:

(1) We propose a dedicatedGAN-based approach that effectively learns the mapping from real-world photos to car- toon images using unpaired image sets for training. Our method is able to generate high-quality stylized cartoons, which are substantially better than state-of-the-art methods. When cartoon images from individual artists are used for training, our method is able to reproduce their styles.

(2) We propose two simple yet effective loss functions in GAN-based architecture. In the generative network, to cope with substantial style variation between photos and car- toons, we introduce a semantic loss defined as an $\ell 1$ sparse regularization in the high-level feature maps of the VGG network . In the discriminator network, we propose an edge-promoting adversarial loss for preserving clear edges.

(3) We further introduce an initialization phase to im- prove the convergence of the network to the target manifold. Our method is much more efficient to train than existing methods.

But Machine Learning is constantly evolving thus expanding in almost every field. And research work done by Xinrui Wang and Jinze Yu has enabled us tocartoonize real high-quality images with just a little training.

The process of converting real-life high-quality pictures into practical cartoon scenes is known as **cartoonization**.

Earlier models that proposed the same approach used **black-box models**, the former model achieves great accuracies but downturns the stylization quality causing some bad cases. Like, every cartoon workflow considers different features, these variations pose a relevant effect on black-box models.

To overcome the drawbacks of the former model, more emphasis was given upon human painting behaviors and cartoon images of different styles, and a **white-box model** was developed.
The model decomposes images into three different cartoon representations, which further counsel the network optimization to generate cartoonized images.
Surface Representation: It helps to extract smooth surfaces of the image that contains a weighted low-frequency component where the color composition and surface texture are retained along with edges, textures, and details.
Structure Representation: It helps to derive global structural information and sparse color blocks, once done we implement adaptive coloring algorithms like the Felzenswalb algorithm to develop structural representation that can help us to generate sparse visual effects for celluloid styled cartoon process.
Textured Representation: It helps us to retain painted details and edges. The three-dimensional image is converted to single-channel intensity map that helps to retain pixel intensity compromising color and luminance, it follows the approach of manual artist that first draw a line sketch with contours and then apply colors to it. The extracted outputs are fed to a Generative Neural Networks (GAN) framework, which helps to optimize our problem making the solution more flexible and diversified.

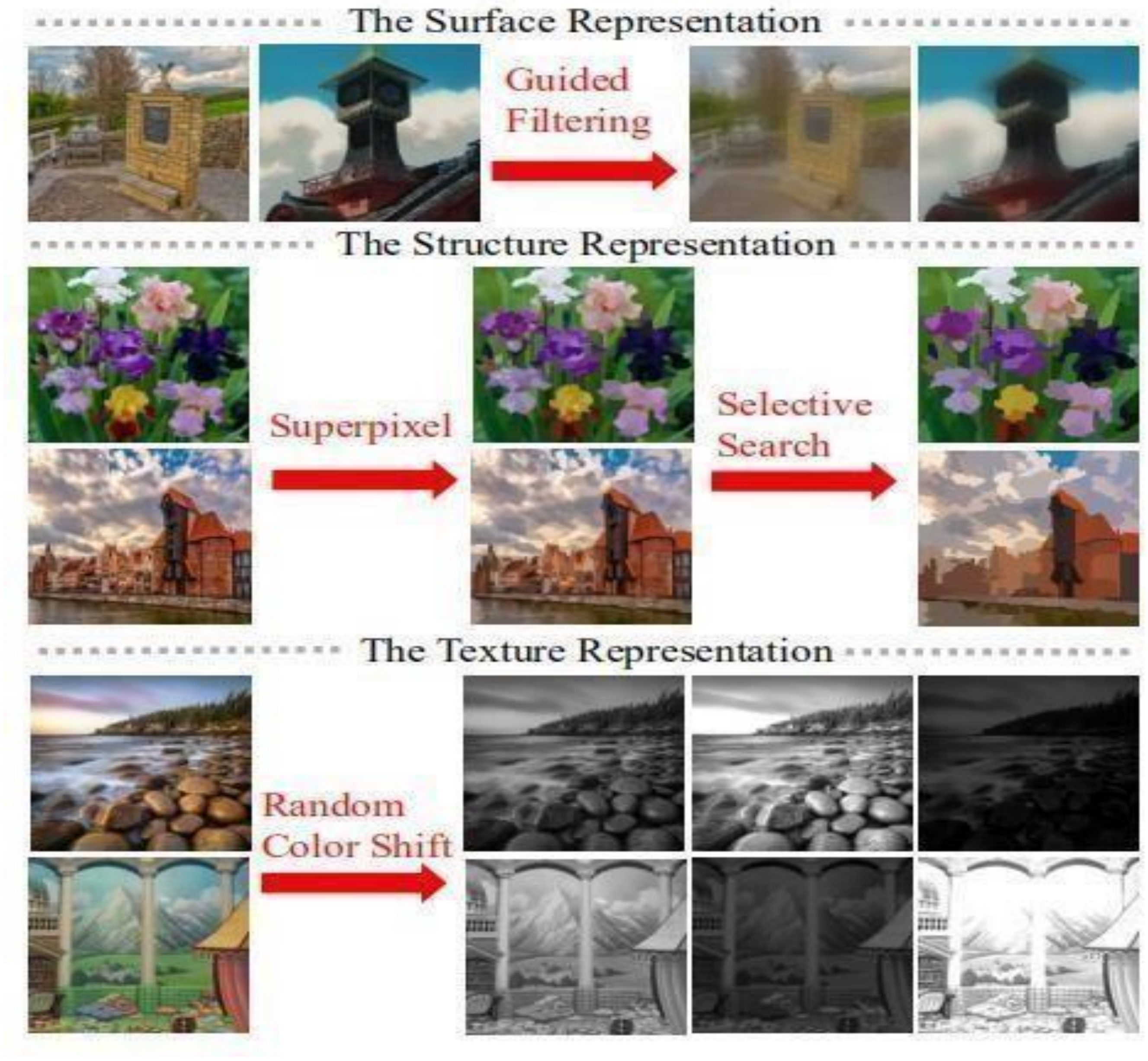

The Surface Representation
Guided Filtering
The Structure Representation
Superpixel
Selective Search
The Texture Representation
Random Color Shift

Figure2

2. Proposed Approach

PREPROCESSING

Along with the proposed three-step approach, preprocessing is an important part of our model. It helps to smoothen the image, filter the features, converting it to sketches, and translating the output from a domain to another. After implementing these related work we can be sure that the output generated by our model will give us the best output that retains the highest quality features.

- **Super-pixel and Structure Extraction:** This method is used to divide the image into regions and defining a predicate for measuring the boundary between two regions. Based on the predicate segmentation, an algorithm is developed whose decision is based on a greedy technique but still helps to satisfy global properties. After identification of contours, we implement ***Gradient Ascent*** to initialize the image with rough clusters and iteratively amend the clusters until convergence. Advancing our process, to develop a cartoon-like segmentation method we use **the *Felzenszwalb algorithm*** that helps us to seize global content information and produce practically usable results for celluloid style cartoon workflows.

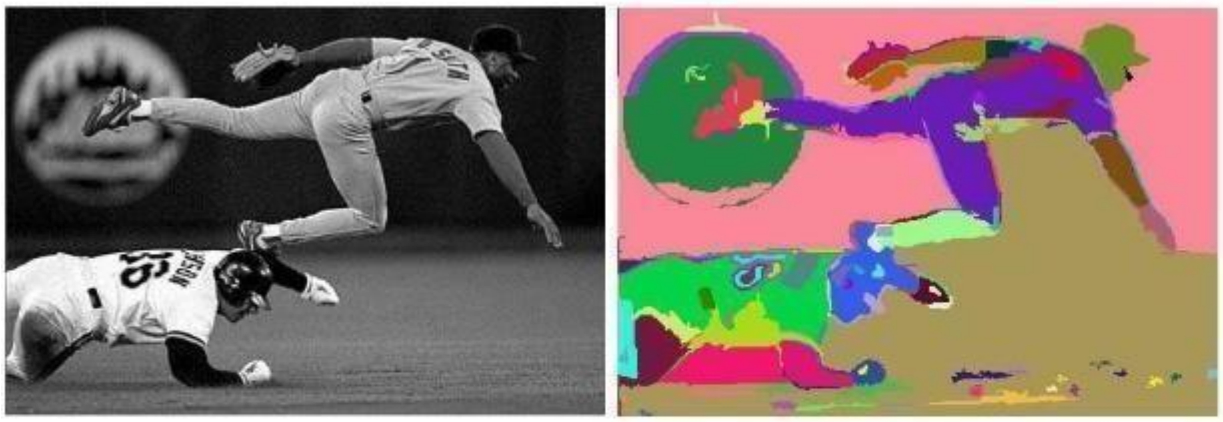


Figure3

- **Image Smoothening**: To extract smooth and cartoon resembling surfaces from images, ***Guided filters*** are used. A guided filter is an advanced version of ***Bilateral filters*** with better near the edge behavior. The goal is simply *removing/significantly decreasing* the noise and obtaining useful image structures. The filtering output of the guided filter is an optimal linear transform of an input image. Following the approach of Bilateral filters it retains smoothing property and in addition, is free from gradient reversal artifacts.

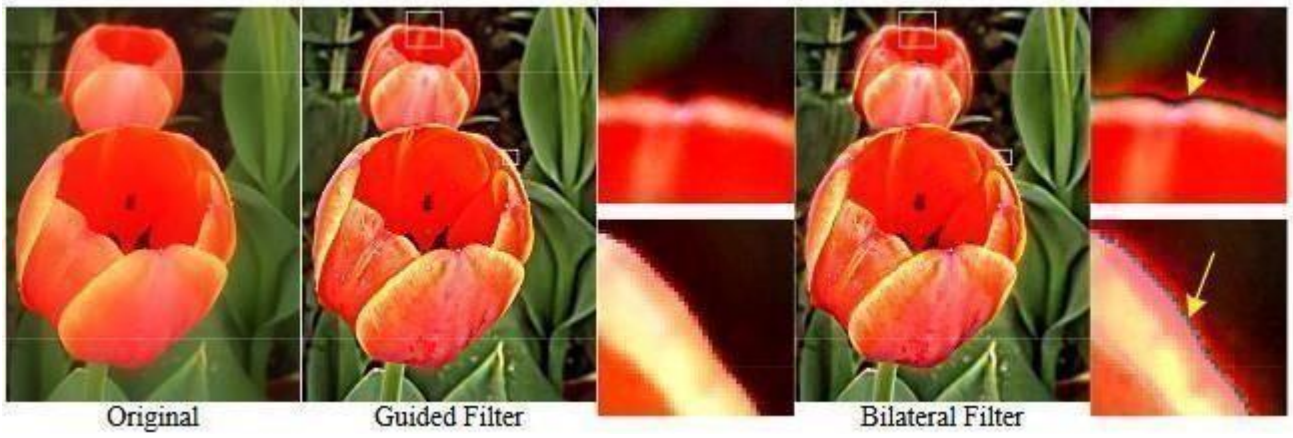


Figure 4

- **Non-photorealisticRendering:** It helps to convert images into artistic styles such as sketching, painting, and water-coloring. To expand its functionality we use it with ***Neural Style Transfer Methods*** that helps to sum up the style of one image and another. The combined piece of code helps to mark semantic edges while segregating image details. But in the "White box cartoonization" method a single image is utilized and learns cartoonist features from a set of animated visuals allowing our model to produce high-quality output on diverse cases.

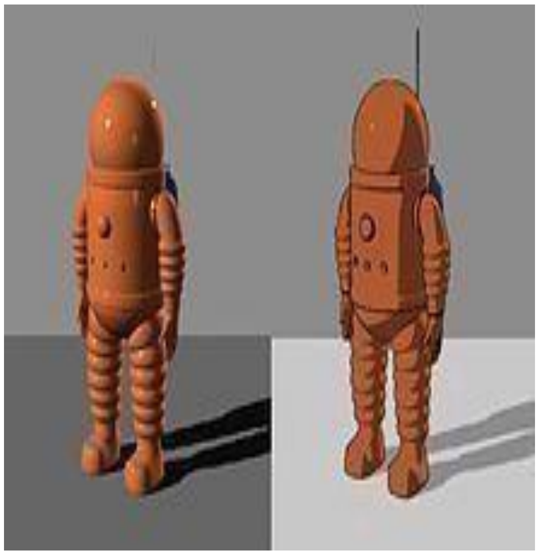


Figure5

- Generative Adversarial Network-It is an image synthesizer that helps to generate new data using joint probability. To generate new images it uses Generator and Discriminator. The generator makes images and Discriminator checks images to be real or fake and then sends feedback to the generator thus asking him to generate better data. The more both networks are trained, the better images we get.

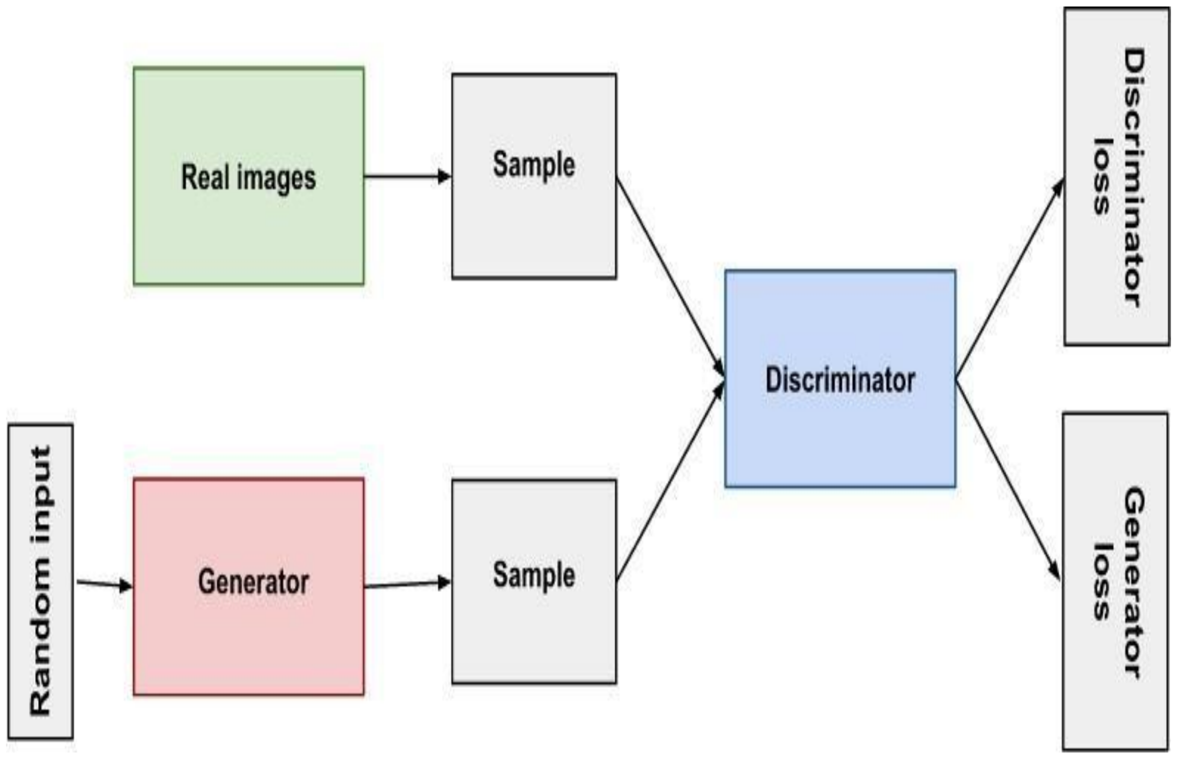


Figure6

- **Image-to-Image Translation**: The drawback with GAN is, it only works for given training data, but paired training data isn’t always available. To overcome the drawback we employ cycleGAN where the goal is to translate an image from a source domain X to a target domain Y even in absence of paired training data.

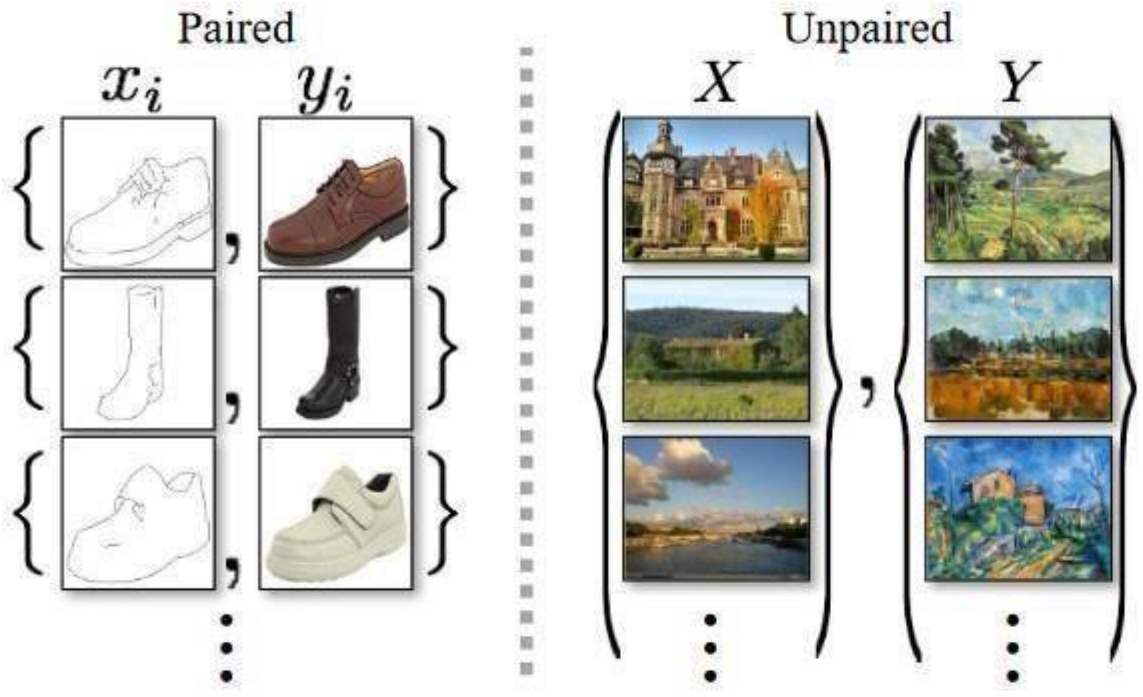


Figure7

To continue with the process, we segregate image features which enforces the network to learn different features with separate objectives, making the process more robust.

2. LITERATURE SURVEY Related Work

Non-photorealistic rendering (NPR)

Many NPR algorithms have been developed, either au- tomatically or semi-automatically, to mimic specific artis- tic styles including cartoons . Some works render 3D shapes in simple shading, which creates cartoon-like ef- fect . 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 im- ages with flat shading, mimicking cartoon styles. Such methods use either image filtering or formulations in optimization problems . However, it is difficult to cap- ture rich artistic styles using simple mathematical formulas. In particular, applying filtering or optimization uniformly to the entirely image does not give the high-level abstrac- tion that an artist would normally do, such as making object boundaries clear. To improve the results, alternative meth- ods rely on segmentation of images/videos , although at the costof

requiring someuser interaction. Dedicated meth- ods have also been developed for portraits , where semantic segmentation can be derived automatically by de- tecting facial components. However, such methods cannot cope with general images.

Stylization with neural networks

Convolutional Neural Networks (CNNs) have received considerable attention for solving many computer vision problems. Instead of developing specific NPR al- gorithms which require substantial effort for each style, style transfer has been actively researched. Unlike tradi- tional 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 re- sult 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 vari- ety of artistic styles automatically. However, it requires the content and style images to be reasonably similar. Further- more, when images contain multiple objects, it may transfer styles to semantically different regions. The results for car- toon style transfer are more problematic, as they often fail to reproduce clear edges or smooth shading.
Li and Wand obtained style transfer by local match- ing of CNN feature map sand 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 se- mantically 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 regard- ing 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 in painting, image super-resolution , etc. The key idea of a GAN model is to train two networks (i.e., a generator and a dis- criminator) iteratively, whereby the adversarial loss provided by the discriminator pushes the generated images to- wards 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 per- form 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 is based on the combined mapping that the map 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-levelabstrac- tion and clear edges) of cartoon images. As a comparison, our method utilizes a GAN model to learn the mapping be- tween 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 re- cently introduced concept of residual blocks is a pow- erful 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 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 gra- dient when the unit is not active. We integrate these tech- niques in our cartoonization deep architecture.

CartoonGAN

We design the generator and discrimina- tor networks to suit the particularity of cartoon images; see Figure 2 for an overview. We formulate the process oflearning to transform real- world photos into cartoon images as a mapping function which maps the photo manifold P to the cartoon mani- fold C.
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 mani- fold or ing data Sdata(p) = {pi |i = 1 ...N } P ⊂ and Sdata(c) = {ci |i = 1 ...M } C, wh⊂ere 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 pro- viding the adversarial loss for G. Let L be the loss function,

G* and D* be the weights of the network. 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 Sec- tion 3.2. To further improve the network convergence, we propose an initialization phase and incorporate it into Car- toonGAN, which is summarized in Section 3.3.

CartoonGAN architecture

Refer to Figure 2. In CartoonGAN, the generator net- work G is used to map input images to the cartoon 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 . Finally, the output cartoon style images are reconstructed by two up-convolution blocks which
contain fractionally strided convolutional layer with stride 1/2 and a final convolutional layer with 7 × 7 kernels.

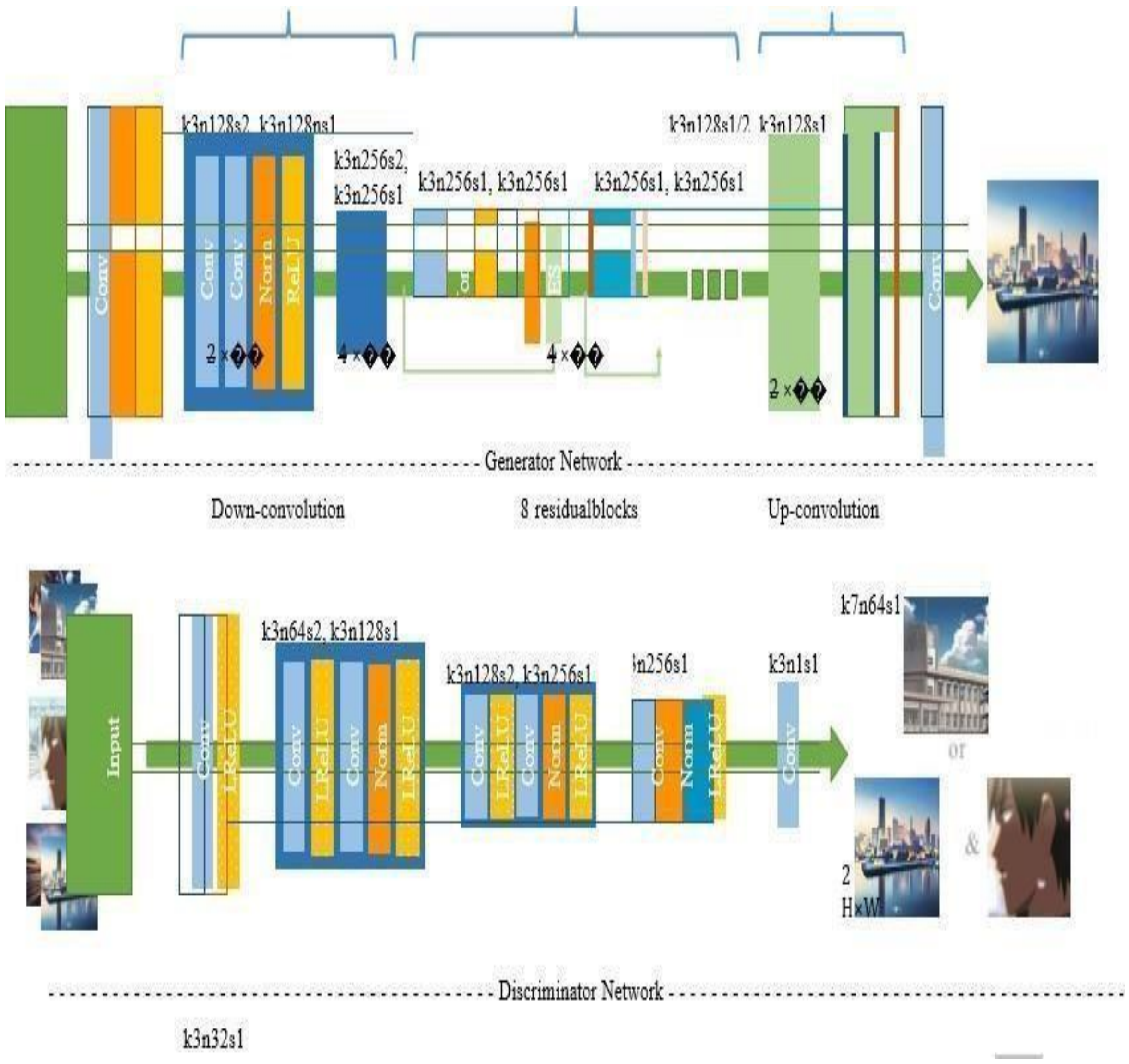


Figure 8. Architecture of the generator and discriminator networks in the proposed CartoonGAN, in which k is the kernel size, n is the number of feature maps and s is the stride in each convolutional layer, 'norm' indicates a normalization layer and 'ES' indicates elementwise sum.

Complementary to the generator network, the discrimi- nator 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 reg- ular 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 re- duce 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 classifica- tion response. Leaky ReLU (LReLU) with α = 0.2 is used after each normalization layer.

Loss function

The loss function L(G,D) in Eq.(1) consists of two parts: (1) the adversarial loss Ladv (G,D) (Section 3.2.1), which drives the generator network to achieve the de- sired manifold transformation, and (2) the content loss Lcon(G,D) (Section 3.2.2), which preserves the image content during cartoon stylization. We use a simple addi- tive form for the loss function:

$$L(G,D)= Ladv (G,D) + \omega Lcon(G,D),$$

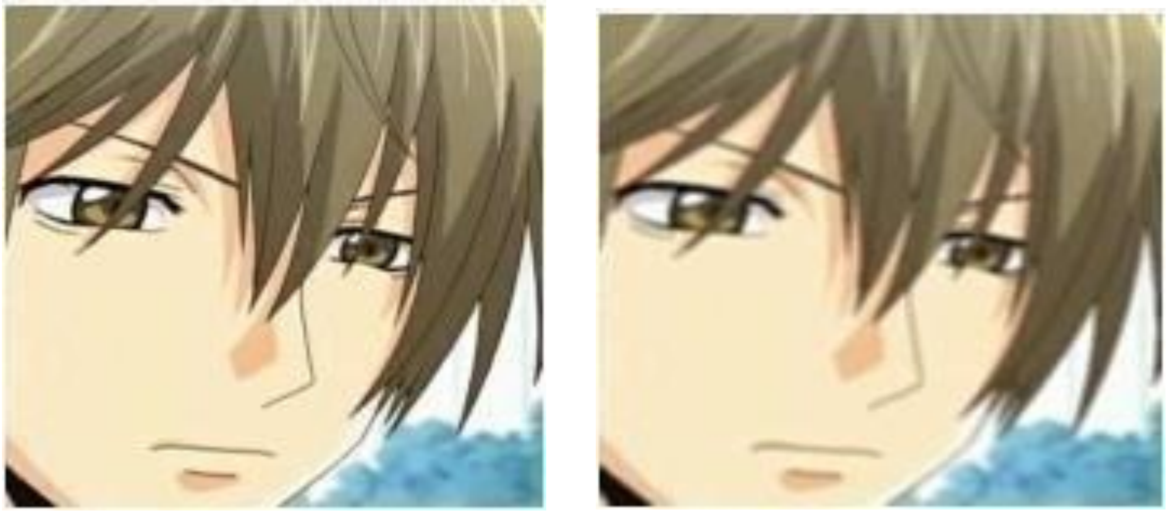
where ω is the weight to balance the two given losses. Larger ω leads to more content information from the in- put photos to be retained, and therefore, results in stylized images with more detailed textures. In all our experiments, we set ω = 10 which achieves a good balance of style and content preservation.

Adversarial loss Ladv (G,D)

The adversarial loss is applied to both networks G and D, which affects the cartoon transformation process in the gen- erator network G. Its value indicates to what extent the out- put image of the generator G automatically generate a set of images Sdata(e) = {ei |i = 1 ...M } ⊂ E by remov- ing clear edges in Sdata(c), where C and E are the cartoon manifold and the manifold of cartoon-like images without clear edges, respectively. In more detail, for each image ci ∈ Sdata(c), we apply the following three steps: (1) detect edge pixels using a standard Canny edge detector [2], (2) di- late 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 pk in the photo manifold P, the generator G out- puts a generated image G(pk). In CartoonGAN, the goal of training the discriminator D is to maximize the probability of assigning the correct label to looks like a cartoon image. In previous GAN frameworks , the task of the dis- criminator D is to figure out whether the input image is syn- thesized 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 suf- ficient for transforming photos to cartoons. This is because the presentation of clear edges is an important characteris- tic of cartoon images, but the proportion of these edges is usually very small in the whole image. Therefore, an out- put image without clearly reproduced edges but with correct shading is likely to confuse the discriminator trained with a

standard loss. To circumvent thisoblem, from the training cartoon images Sdata(c) ⊂ C, we G(pk), the cartoon images without clear edges (i.e., ej ∈ Sdata(e)) and the real car- toon images (i.e., ci ∈ Sdata(c)), such that the generator G can be guided correctly by transforming the input to the correct manifold. Therefore, we define the edge-promoting adversarial loss as:

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



(a) A cartoon image ci b) The edge-smoothed version ei

. Figure9

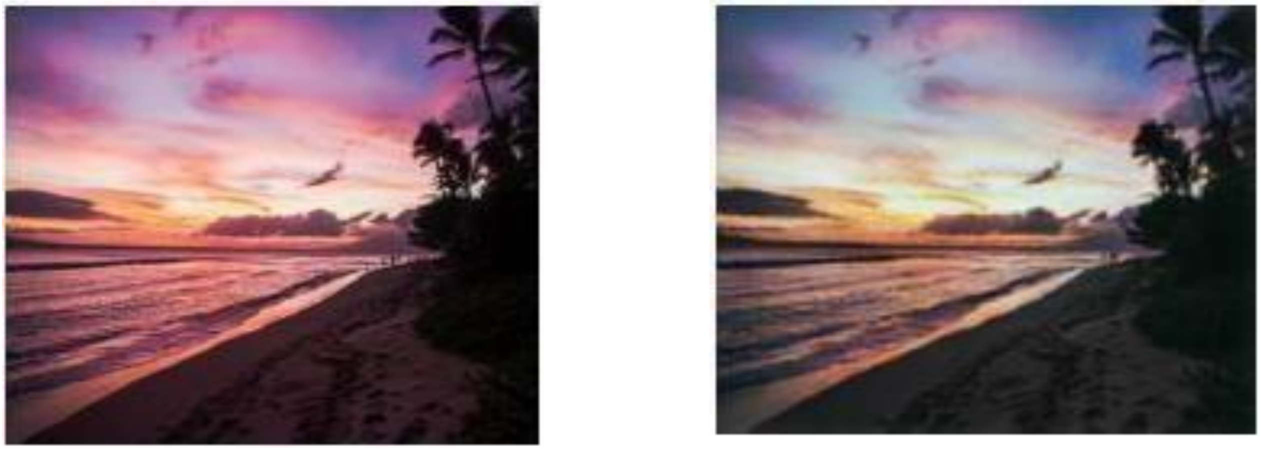
Figure 9. By removing clear edges in a cartoon image c ∈ Sdata(c), we generate a corresponding image ei ∈ Sdata(e).

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 fea- ture maps in the VGG network [30] pre-trained by [27], which has been demonstrated to have good object preser- vation ability. Accordingly, we define the content loss as:

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

where l refers to the feature maps of a specific VGG layer.Unlike other image generation methods , we de- fine our semantic content loss using the ℓ1 sparse regular- ization of VGG feature maps between the input photo and the generated cartoon image. This is due to the fact that car- toon 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 concen- trate on local regions where the representation and regional characteristics change dramatically. ℓ1 sparse regulariza- tion is able to cope with such changes much better than the standard ℓ2 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.



(a) Original photo (b) Image after initialization

Figure 10. For an original photo (a), the image (b) is the result after the initialization phase. See the main text for details.

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 tar- get of the generator network G is to reconstruct the input photo in a cartoon style while keeping the semantic content. We start the adversarial learning framework with a generatorwhich only reconstructs the content of input images. For this purpose, in the initialization phase, we pre- train the generator network G with only the semantic content loss loss Lcon(G,D). Figure 4 shows an example of the recon- structed image after 10 epochs of this initialization training phase, which already produces reasonable reconstruction. Our experimental results show that this simple initialization phase helps CartoonGAN fast converge to a good configu- ration, without premature convergence. Similar observation is made in which uses the content image to initialize the result image to improve style transfer quality.

4. Experiments

We implemented our CartoonGAN in Torch and Lua language. The trained models in our experiments are available Able to facilitate evaluation of future methods. All experi- ments were performed on an NVIDIA Titan Xp GPU CartoonGAN is able to produce high-quality cartoon stylization using the data of individual artists for

training, which are easily obtained from cartoon videos, since our method does not require paired images. Different artists have their unique cartoon styles, which can be effectively learned by CartoonGAN. Some results of different artistic styles generated by CartoonGAN are shown in Figure 5.
To compare CartoonGAN with state of the art, we col- lected the training and test data as presented in Section 4.1. In Section 4.2, we present the comparisonbetween the pro- posed method and representative stylization methods. In Section 4.3, we present a further ablation experiment to an- alyze the effectiveness of each component in our Cartoon- GAN model.

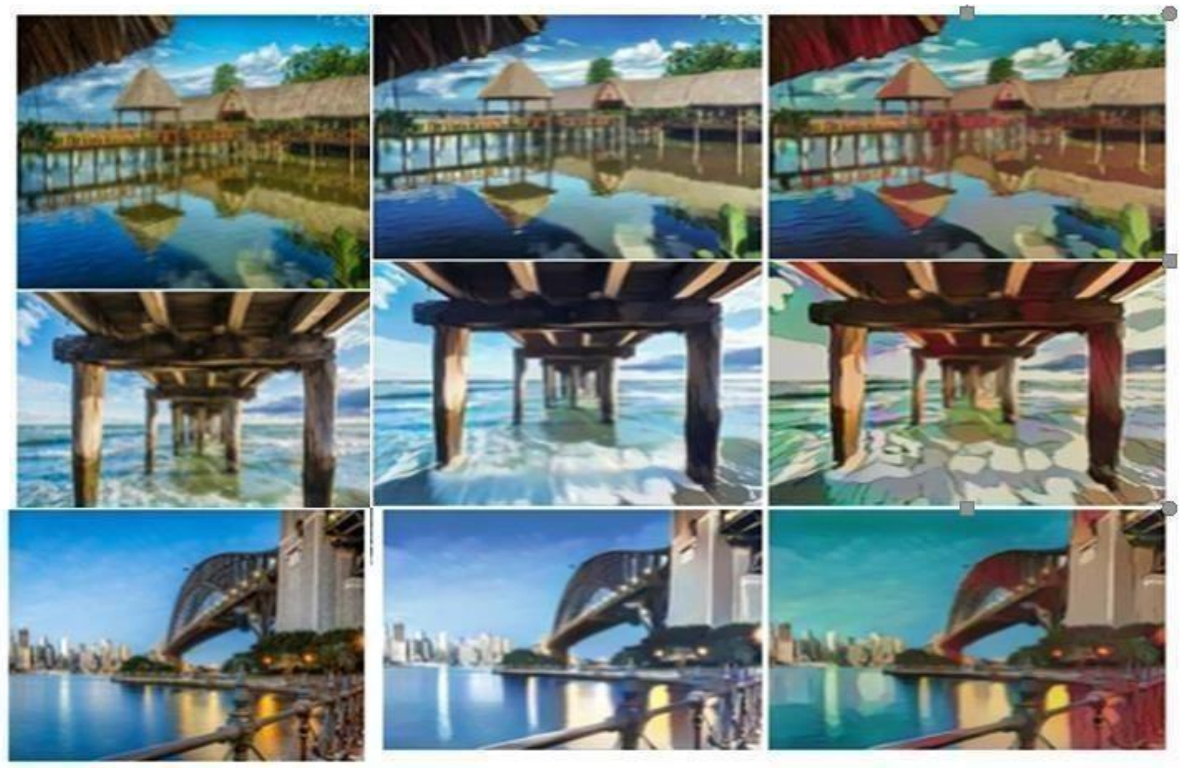


Figure 11Some results of different artistic styles generated by Car- toonGAN. (a) Input real-world photos. (b) Makoto Shinkai style. (c) Miyazaki Hayao style.

Data

The training data contains real-world photos and cartoon images, and the test data only includes real-world photos. Allthetrainingimagesareresizedandcroppedto 256×256.

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

Cartoon images.

Different artists have different styles when creating cartoon images of real-world scenes. To ob- tain a set of cartoon images with 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, 4,573 and 4,212 cartoon images from several short cartoon videos are used for training the Makoto Shinkai and Mamoru Hosoda

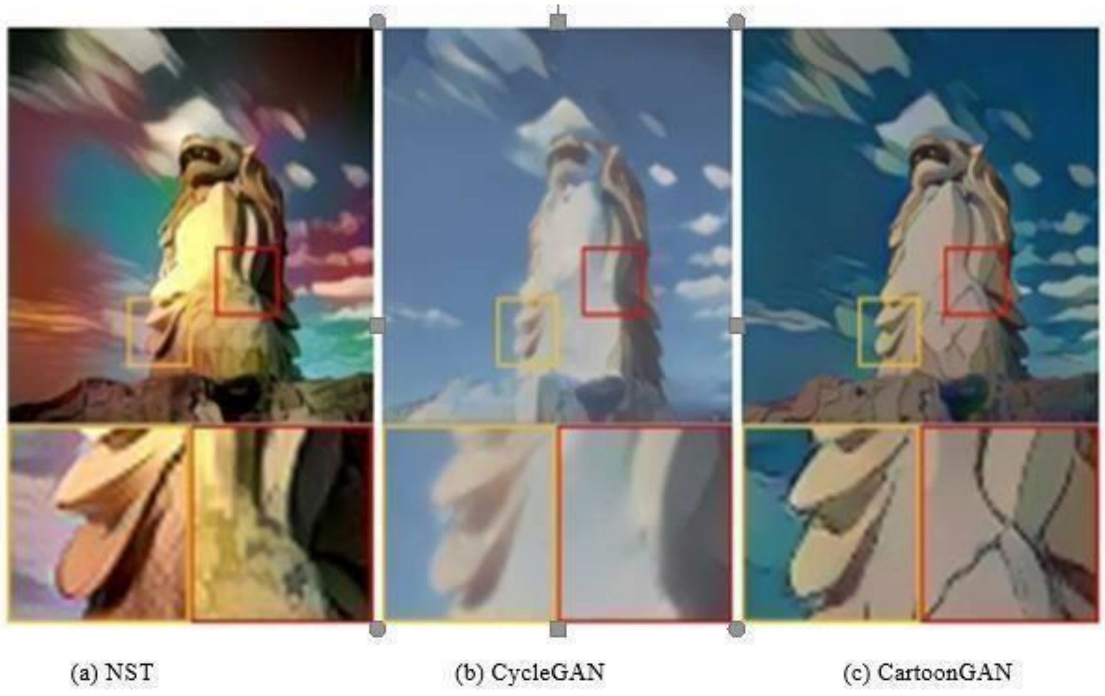


Figure 12. Details of edge generation. (a) The result of NST [6] using all the images in the training set as the style image. (b) CycleGAN with the identity loss. (c) Our result.

Comparison with state of the art

We first compare Cartoon GAN with two recently pro- posed methods in CNN-based stylization, namely NST and Cycle GAN . Note that the original NST takes one style image Is and one content image Ic as input, and trans- fers the style from Is to Ic. For fair comparison, we apply two adaptations of NST. In the first adaptation, we manually choose a style image which has close content 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. We also compare two versions of Cycle- GAN, i.e., without and with the identity loss Lidentity . The incorporation of this loss tends to produce stylized images with better content preservation. 200 epochs were trained for both Cycle GAN and our Cartoon GAN.

Qualitative results. , which clearly demonstrate that NST and Cycle GAN cannot deal with cartoon styles well. In comparison, by reproducing the necessary clear edges and smooth shading while retaining the content of the input photo, our Cartoon- GAN model produces hight- quality results .

More specifically, NST using only a 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 13b). When NST is extended to take more training data, rich styles can be better learned. However, the styl- ized images tend to have local regions stylized differently, causing inconsistency artifacts (Figure 13c).

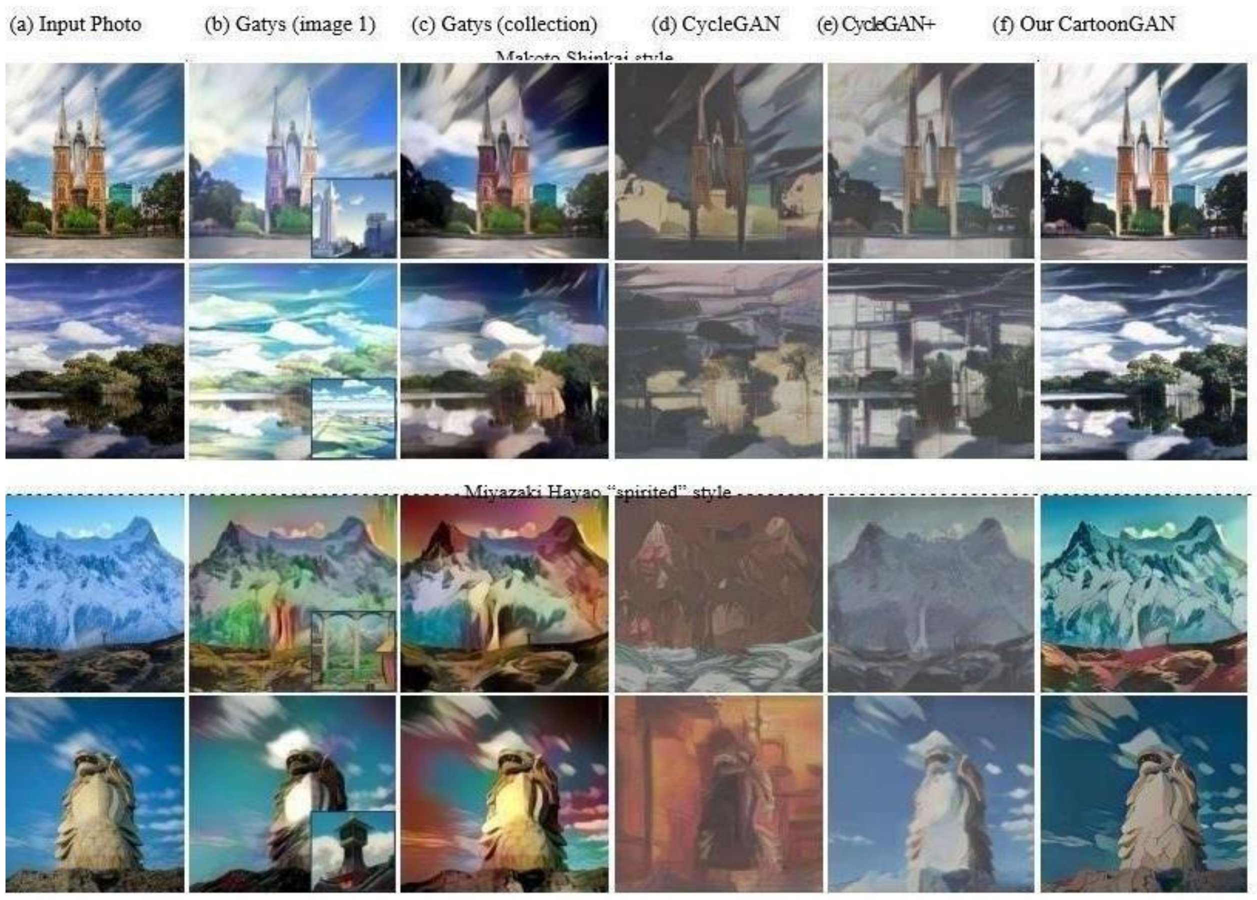


Figure13

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 well (Figure 6d). The identity loss is useful to avoid this prob- lem, but the stylization results are still far from satisfactory (Figure 6e). In comparison, Cartoon GAN produces high- quality cartoonization which well follows individual artist's style. Figure 7 shows close-up views of an example in Figure 6, demonstrating that our

Cartoon GAN generates thees- sential edges which are very important for the cartoon style.
Our CartoonGAN has the same property of not requiring paired images for training as CycleGAN. However, Car- toonGAN takes much less training time. For each epoch, CycleGAN and CycleGAN with Lidentity take 2291.77s and 3020.31s, respectively, whereas CartoonGAN only takes 1517.69s, about half compared with CycleGAN + Lidentity . This is because CycleGAN needs to train two GAN models for bidirectional mappings, which seriously 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 architec- ture to restrain the content, CartoonGAN can learn cartoon stylization moreefficiently.
We also compare our method with CNNMRF and Deep Analogy , with Paprika and Mamoru Hosoda

Roles of components in loss function

We perform the ablation experiment to study the role of each part in CartoonGAN. Figure 9 shows the examples of ablations of our full loss function, in which all the results are trained by Makoto Shinkai style's data. The follow- ing results show that each component plays an important role in CartoonGAN. First, the initialization phase helps the generator G quickly converge to a reasonable manifold. As shown in Fig. 9b, without initialization, although some key features are shown, the styles are far from expectation. Second, even with a suitable VGG layer, large and often local- ized differences in feature maps of input and cartoon style images are still needed due to massive style differences. Us- ing the $\ell 1$ sparse regularization (instead of $\ell 2$) of high-level VGG feature maps helps cope with substantial style differ- ences between cartoon images and photos. Last, the elabo- rately designed edge loss guides the generator G to produce clear edges in results, leading to better cartoon style images.

The majority of photo editing websites offer the so-called Cartoon Effect. The main advantages of online photo to cartoon effect apps are simplicity and quickness. You'll have to upload a photo from your computer or from the web, find Cartoon Effect in the tool set or choose between styles or variants of this funny photo effect (like in case of www.picturetopeople.org, Kuso Cartoon) and press the button Apply (or Go). The image processing varies from several seconds up to 1-2 minutes.

However, as all quick online solutions these apps have drawbacks. A lot of photo online photo editing tools are rather humdrum because they are deprived of enhancement features. In these apps cartoonization is limited to 1-click operation.

Besides, sometimes colors may become blurred and it leads to an unsatisfactory result. Such apps as www.convert to cartoon.com, Photo.to, Any Making and others belong to this group. At the same time there are online photo editors with more advanced tools. They have a variety of adjustment options. For example, Be Funky helps you modify sketch brightness, contrast, smoothness and other details.

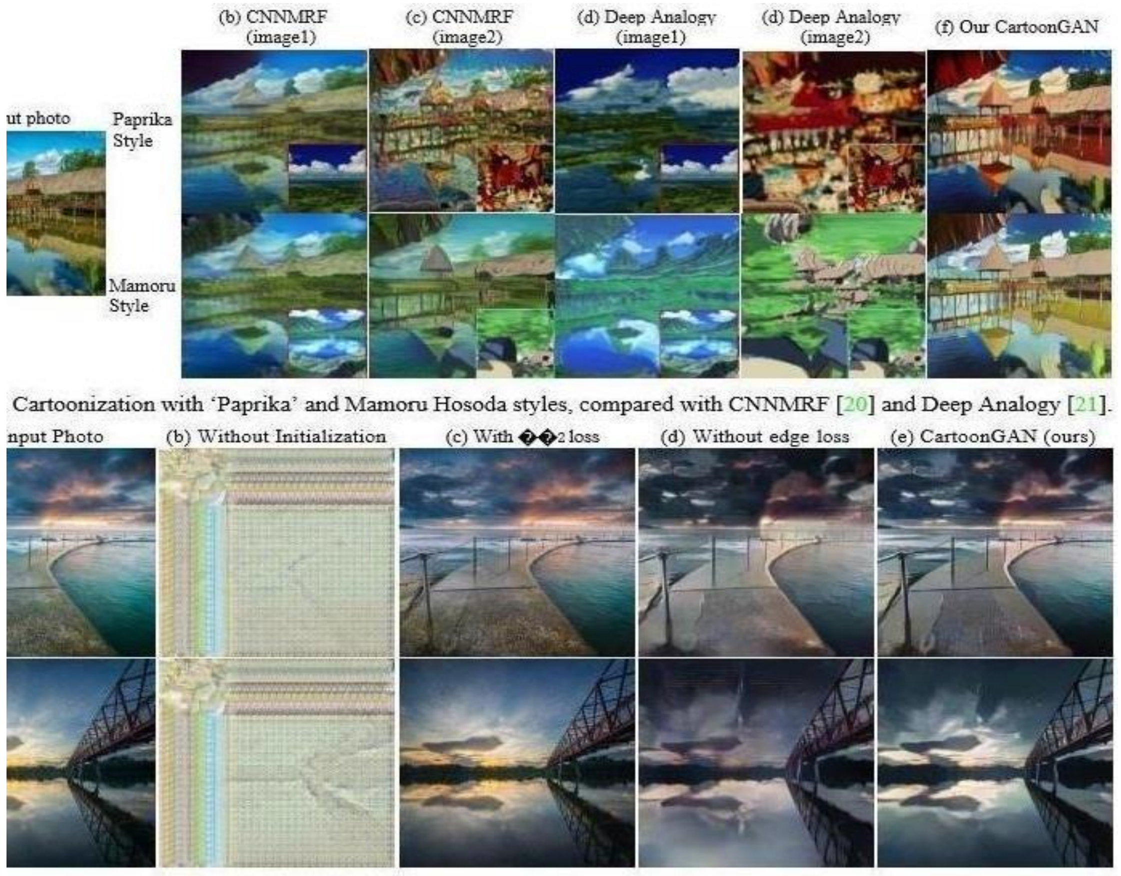

(b) CNNMRF (image1)
(c) CNNMRF (image2)
(d) Deep Analogy (image1)
(d) Deep Analogy (image2)
(f) Our CartoonGAN
ut photo
Paprika Style
Mamoru Style
Cartoonization with 'Paprika' and Mamoru Hosoda styles, compared with CNNMRF [20] and Deep Analogy [21].
nput Photo
(b) Without Initialization
(c) With ��2 loss
(d) Without edge loss
(e) CartoonGAN (ours)

Figure14

3. The Workflow

- The input is first passed through Surface representation where Structural and Textural features are removed, once we imitate cartoon painting style and smooth surfaces, the output is passed through guided filters in order to retain smooth edges. A discriminator *Ds* is proposed to verify whether result and paired cartoon images have similar surfaces, and regulate the generator *G* to learn the information stored in the extracted surface representation.

- The structural features are then passed through Structural representation that clear boundaries in the cellular style framework and then we implement Felzenszwalb algorithm to segment the areas. The algorithm assists us in coloring each segment with an average pixel value. To impose a spatial constraint on global content between outputs and provided paired cartoons we use pre-trained *VGGNetwork*.
- As discussed earlier the variation of luminance and color information are non-trivial issues to the model, therefore, we choose a random color shift algorithm to convert three-channel input to single-dimension outputs that cling to high-quality features. A discriminator Dt is then proposed to verify textual features from the result and paired cartoon image, and regulates generator G to learn the information stored in extracted texture representation.

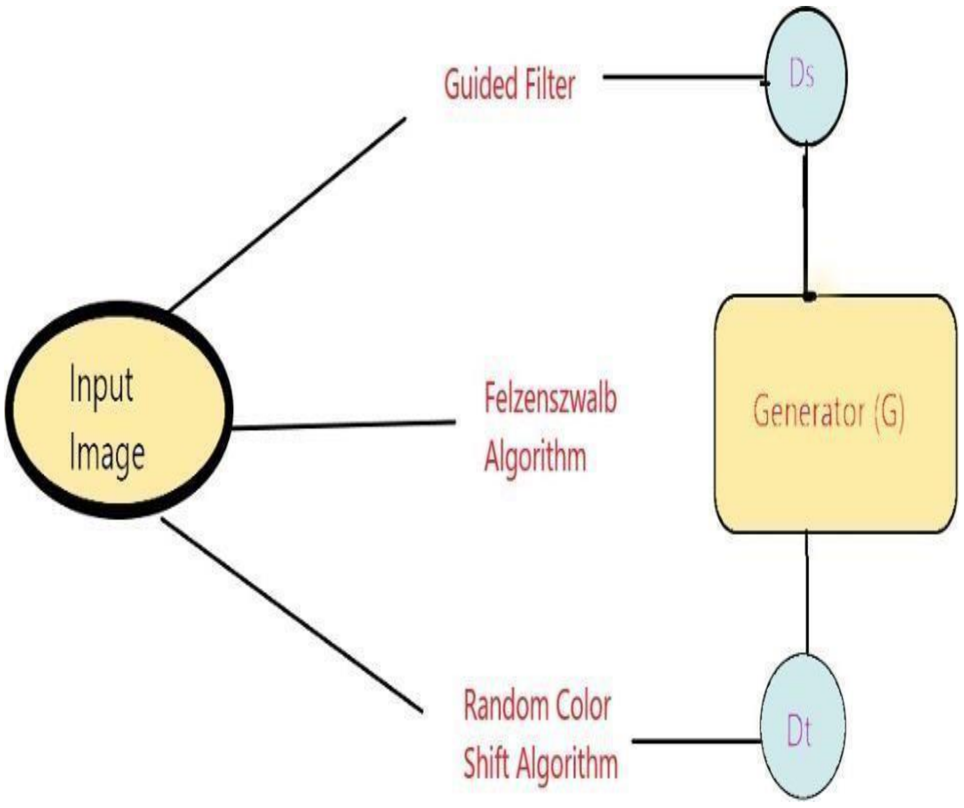


Figure15

4.SOURCE CODE

```
import cv2 #for image processing
import easygui #to open the filebox
import numpy as np #to store image
import imageio #to read image stored at particular path import
sys
import matplotlib.pyplot as plt
import os import tkinter as tk
from tkinter import filedialog
from tkinter import * from PIL
import ImageTk, Image
""" fileopenbox opens the box to choose file
and help us store file path as string """
```

```
def upload():
    ImagePath=easygui.fileopenbox()
    cartoonify(ImagePath) #read the
    image
    originalmage = cv2.imread(ImagePath)
    originalmage = cv2.cvtColor(originalmage, cv2.COLOR_BGR2RGB)
#print(image) # image is stored in form of numbers
# confirm that image is chosen if originalmage is None:
    print("Can not find any image. Choose appropriate file")
    sys.exit()
    ReSized1 = cv2.resize(originalmage, (960, 540))
#plt.imshow(ReSized1, cmap='gray') #converting
an image to grayscale
grayScaleImage = cv2.cvtColor(originalmage, cv2.COLOR_BGR2GRAY)
ReSized2 = cv2.resize(grayScaleImage, (960, 540))
#plt.imshow(ReSized2, cmap='gray') #applying median
blur to smoothen an image smoothGrayScale =
cv2.medianBlur(grayScaleImage, 5)
ReSized3 = cv2.resize(smoothGrayScale, (960, 540))
#plt.imshow(ReSized3, cmap='gray')
#retrieving the edges for cartoon effect #by using
thresholding technique getEdge =
cv2.adaptiveThreshold(smoothGrayScale, 255,
cv2.ADAPTIVE_THRESH_MEAN_C,
cv2.THRESH_BINARY, 9, 9)
ReSized4 = cv2.resize(getEdge, (960, 540))
#plt.imshow(ReSized4, cmap='gray')
#applying bilateral filter to remove noise #and
keep edge sharp as required
colorImage = cv2.bilateralFilter(originalmage, 9, 300, 300)
ReSized5 = cv2.resize(colorImage, (960, 540))
#plt.imshow(ReSized5, cmap='gray')
#masking edged image with our "BEAUTIFY" image
cartoonImage = cv2.bitwise_and(colorImage, colorImage, mask=getEdge)
ReSized6 = cv2.resize(cartoonImage, (960, 540))
#plt.imshow(ReSized6, cmap='gray')
# Plotting the whole transition images=[ReSized1, ReSized2, ReSized3,
ReSized4, ReSized5, ReSized6]
fig, axes = plt.subplots(3,2, figsize=(8,8), subplot_kw={'xticks':[], 'yticks':[]},
gridspec_kw=dict(hspace=0.1, wspace=0.1)) for
i, ax in enumerate(axes.flat):
    ax.imshow(images[i], cmap='gray')
//save button code plt.show() def save(ReSized6,
ImagePath): #saving an image using imwrite()
newName="cartoonified_Image" path1 =
os.path.dirname(ImagePath)
extension=os.path.splitext(ImagePath)[1] path =
os.path.join(path1, newName+extension)
    cv2.imwrite(path, cv2.cvtColor(ReSized6, cv2.COLOR_RGB2BGR))
    I = "Image saved by name " + newName +" at "+ path
    tk.messagebox.showinfo(title=None, message=I)
    top=tk.Tk()
top.geometry('400x400')
top.title('Cartoonify Your Image !')
top.configure(background='white')
label=Label(top,background='#CDCDCD', font=('calibri',20,'bold'))
upload=Button(top,text="Cartoonify                                          an
Image",command=upload,padx=10,pady=5)
upload.configure(background='#364156',
foreground='white',font=('calibri',10,'bold'))
upload.pack(side=TOP,pady=50) save1=Button(top,text="Save       cartoon
     image",command=lambda:
save(ImagePath,                                    ReSized6),padx=30,pady=5)
save1.configure(background='#364156',
foreground='white',font=('calibri',10,'bold'))
save1.pack(side=TOP,pady=50) top.mainloop()
```

RESULT

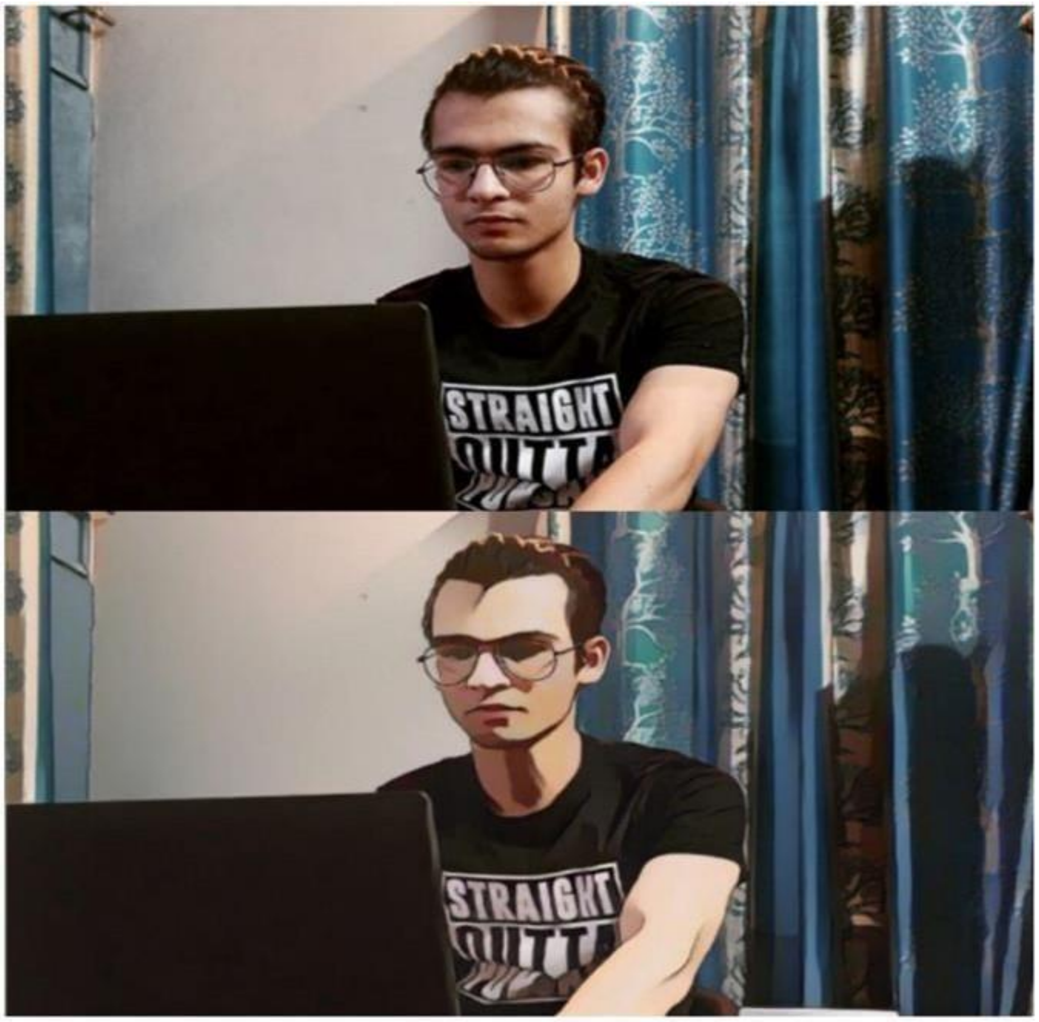


Figure16

5. Conclusion and Future Work

In this paper we proposed CartoonGAN, a Generative Adversarial Network to transform real-world photos to high-quality cartoon style images. Aiming at recreating faithful characteristics of cartoon images, we propose (1) a novel edge-promoting adversarial loss for clear edges, and

(2) an ℓ1 sparse regularization of high-level feature maps in the VGG network for content loss, which provides suf- ficient flexibility for reproducing smooth shading. We also propose a simple yet efficient initialization phase to help improve convergence. The experiments show that Cartoon- GAN is able to learn a model that transforms photos of real- world scenes to cartoon style images with high quality and high efficiency, significantly outperforming the state-of-the- art stylization methods.

In the future work, due to the importance of portrait, we would like to investigate how to exploit local facial features to improve cartoon stylization for human faces. Although we design our loss functions to tackle specific nature of cartoon stylization, similar ideas are useful for other image synthesis tasks, which we will investigate further. We also plan to add sequential constraints to the training process to extend our method to handling videos.

6.References

[1] J. Bruna, P. Sprechmann, and Y. LeCun. Super-resolution with deep convolutional sufficient statistics. In *International Conference on Learning Representations (ICLR)*, 2016.

[2] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (6):679–698, 1986.

[3] Y. Chen, Y.-K. Lai, and Y.-J. Liu. Transforming photos to comics using convolutional neural networks. In *Interna- tional Conference on Image Processing*, 2017.

[4] R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A MATLAB-like environment for machine learning. In *NIPS Workshop on BigLearn*, 2011.

[5] V. Dumoulin, I. Belghazi, B. Poole, A. Lamb, M. Arjovsky, O. Mastropietro, and A. Courville. Adversarially learned in- ference. In *International Conference on Learning Represen- tations (ICLR)*, 2017.

[6] L. Gatys, A. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2414–2423, 2016.

[7] L. A. Gatys, A. S. Ecker, and M. Bethge. Texture synthesis and the controlled generation of natural stimuli using convo- lutional neural networks. *arXiv preprint arXiv:1505.07376*, 12, 2015.

[8] 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 Pat- tern Recognition (CVPR)*, 2017.

[9] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D.Warde-Farley, S.Ozair, A.Courville, andY.Bengio. Gen- erative adversarial nets. In *Advances in Neural Information Processing Systems 27*, pages 2672–2680. 2014.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learn- ing for image recognition. In *IEEE Conference on Com- puter Vision and Pattern Recognition (CVPR)*, pages 770– 778, 2016.

[11] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin. Image analogies. In *ACM SIGGRAPH*, pages 327– 340, 1998.