Anime Scene Generator from Real-world Scenario using Generative Adversarial Networks

Final Year Project Report

4th Year Student Names

Le Xuan Huy HE140555

Bui Thi Bich Ngoc HE140996

Under the supervision of

Dr. Phan Duy Hung

Bachelor of Computer Science

Hoa Lac campus - FPT University

17 June 2015

DECLARATION

Project Title: Anime Scene Generator from Real-world Scenario using Generative Adversarial Networks
Authors: Le Xuan Huy, Bui Thi Bich Ngoc
Student ID: HE140555, HE140996
Supervisor: Dr. Phan Duy Hung

We declare that this thesis entitled *Anime Scene Generator from Real-world Scenario using Generative Adversarial Networks* is the result of our own work except the ones cited in the references. The work has not been accepted for any degree and is not concurrently submitted in the candidature of any other degree.

Le Xuan Huy
HE140555

Bui Thi Bich Ngoc
HE140996

Department of Computer Science
Hoa Lac Campus – FPT University

Date: December 1, 2021

ACKNOWLEDGEMENT

We want to express our deep and genuine gratitude to our research instructor, Dr. Phan Duy Hung, for giving us the opportunity to do research and providing invaluable guidance throughout this research. His dynamism, vision, openness, and motivation have profoundly inspired us. He has taught us the methodology to carry out this study and to present the research works as clearly as possible. It was a great privilege and honor to work and study under his guidance. We would also like to thank our teacher for his compassion and immense knowledge.

Besides our instructor, we also want to extend our thanks to our friends at FPT University for their understanding, support, and patience. Our honest appreciation also goes to FPT University and our teachers there, who have created an excellent environment for our improvement during the four years we have studied here.

We also want to send massive regard to our caring, loving, and supportive families. Their assistance and encouragement were a great comfort and relief for us during our hard times.

Finally, we would like to acknowledge all the people who have supported us to complete the project work directly or indirectly. We want to thank the senior and fellow researchers for their inspiration; and all the artists and creators of artworks, without whom this study will never be possible.

ABSTRACT

This thesis presents a unique approach for image cartoonization and style transferring: translating an image or video in real life into an aesthetic, anime-like frame. By paying exceptional attention to the animation painting conduct, we propose to separately distinguish three feature maps from pictures: the surface description that contains smooth color shading characteristic of animation pictures, the construction depiction that emulates flattened global content and clear boundaries in a typical anime frame, and the texture representation that reflects high-frequency surface, forms, and details in animation pictures. All the extracted information will be fed into the Generator with the help of a VGG based discriminator to learn how to cartoonize a real-world photo. The learning objectives of our technique are independently based on each extracted feature map, making our model controllable and adjustable.

Our solution takes unpaired photos and cartoon/anime images for training which can be fine-tuned for different problems and art styles. It is also incredibly lightweight so as to provide quick and easy inference. Experimental results show that our method can generate high-quality cartoon images from real-world photos and outperforms many existing methods.

Keywords: Style Transfer, Image Translation, Generative Adversarial Networks

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Background

From its crack of dawn, humanity had learned to describe natural scenes that they see on rocks and wood. We came from simple, straight-line sketches to realistic, detailed pictures and even cartoon and artistic drawings. Our creations move from caves to wood, then to paper, and later on, to our computers and Cloud. They serve different purposes in our daily lives, but all share one similarity of depicting human life and mind.

In this thesis, we focus on Anime/Cartoon style drawings and how to generate them from the input we have: real-life footage. One primary approach when it comes to this domain transfer problem is using Generative Adversarial Networks [1].

During the last decades, the world of machine learning, especially computer vision and neural style transfer, has been shaken by the dawn of new research - Generative Adversarial Networks (GAN). This whole new idea has led to many valuable pieces of research of all computer vision fields using GAN. A wide range of applications is found by learning to translate data from its original domain to another, such as style transferring, image colorization, image restoration, and super-resolution.

Depending on the quality of the animation, it might take weeks or even months to produce a minute-long cartoon or anime video with today's technology. Research using cartoon images as input might find that their resources are limited to a relatively small number, around hundreds of thousands (or less) of efficient input. Restriction on intake would probably lead to degraded quality of output results. GANs are born to solve these questions.

Nowadays, GAN has shown its prominence with effective and practical solutions in domain transfer and image generation problems. Some recent research has shown impressive results in many computer vision fields, such as generating random comic characters, translating selfies or pictures to anime style, sketches colorization, super-resolution, etc. A fun yet awesome image generation project called AnimeGAN uses the

DCGAN model trained on a dataset of 143,000 anime character faces to generate new anime faces [37]. However, we can still see some unclean results, partially caused by outliers at the input process. CartoonGAN proposes image translation with unpaired training data, significantly reducing the effort needed for data preprocessing [3]. The project features a simple patch-level discriminator, edge-promoting adversarial loss, and L1 sparse regularization of high-level feature maps in the VGG network for content loss. Nonetheless, this black box model is the enemy of generalization. CycleGAN, one of the first and most inspirational research, introduced us to the Cycle-consistent Adversarial Networks with cycle-consistent loss and full-cycle transform [4]. Their methods are extended and improved in many later studies. A big problem with the Cycle-consistent Adversarial Networks is that they require a considerable amount of input data. Comixify works with videos and tries to convert them into comics [5]. They extract keyframes from the input videos, translate them into comics, and also intend to add speech recognition in future works.

This thesis mainly focuses on translating real-life footage into anime/cartoon style. Moreover, it involves new algorithms and manually collected datasets in order to improve generated animation. Our product is a lightweight and uncomplicated model that can perform style transferring quicker and easier than many others.

1.2 Objectives and Contributions

The primary target of this thesis is to translate images and videos in real life into amine/cartoon style. Many researchers have tried to do this before, but we are looking for a more accurate and artistic result by implementing new methods and evaluators.

This thesis uses a features extraction method of the real-life input which separates images into three different representations. We also use FID as the evaluation metric to calculate and compare our work's performance to SOTA's models.

1.3 Outline

Chapter 1: A general introduction about Anime/Cartoon Style Transfer using GANs and the scope of this thesis.

Chapter 2: List of literature studies that are related to this thesis's fields of research and an overview of all the technologies used in this study for implementation.

Chapter 3: Details implementation of the technologies introduced in the previous chapter. We will analyze and extract features from real-life images and videos and then convert them into anime/cartoon style.

Chapter 4: Display of results from the training model and evaluation metrics. We also include some mathematical and visual comparisons of our model to SOTA models of the same topic.

Chapter 5: Conclusion about the results of testing our models and possible future works that could be done for further improvisation.

CHAPTER 2

RELATED WORKS

2.1 Image Smoothing and Segmentation

2.1.1 Image Smoothing

Image smoothing is an image enhancement process, which is usually applied as one module of image preprocessing in various projects. Smoothing is often used to reduce noise in images and give us a more accurate intensity surface.

Commonly used methods are the classic filtering-based and the optimization-based.

There are many types of smoothing filters, such as box filters (simple averaging), Gaussian filters (center pixels weighted more), edge-preserved filters, bilateral filters, etc. Box filter averages pixel to smooth image and helps erase noises by making variance of noise in the average to become smaller than the variance of the pixel noise. Nevertheless, a significant drawback of box filtering is the reduction of fine image details after smoothing. Tomasi and Manduchi introduced the concept of bilateral filtering applied on color images for edge-preserving smoothing [6], and He et al. use a guided filter - modified bilateral filtering with better performance on edges [7]. The main point is, we need to make sure that the filter smoothes enough to clean up the noise, but not so much as to remove important image gradients.

Optimization approaches have tried to utilize quadratic cost functions or solve a linear system, which is pretty the same as implicit filters. Farbman et al. discard the old ways by utilizing a weighted least squares optimization framework to tighten up the edge-preserving operator [8]. Min et al. perform global smoother for edge-preserving image smoothing based on weighted least squares [9]. However, the explicit filters are often simpler and lighter than the optimization-based ones. Thus, we will adjust a trainable

guided filter for this thesis - an explicit image filter [10] to extract the top smooth layer from our input.

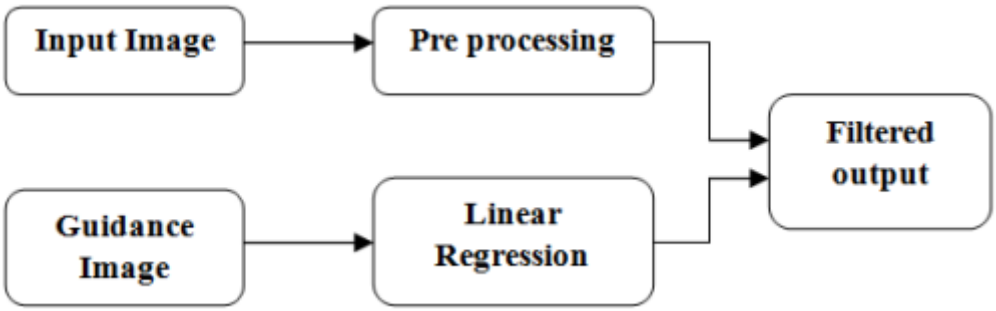


Fig. 1. Guided Image Filter

Original image | Guided Filter | Fast Guided Filter

Fig. 2. He et al. guided filter with window radius $r = 4$, regularization parameter $\varepsilon = 0.2^2$, subsampling ratio $s = 4$.

2.1.2 Image Segmentation

Image segmentation aims to separate images into different regions. Here we want to use segmentation to establish boundaries between image regions in order to extract its structure representation.

Introduced by Ren and Malik in 2003, superpixels were created to group pixels similar in color and other low-level properties. Various well-known segmentation and grouping methods exploit the power of superpixel algorithms to perform in a faster and more memory-efficient manner.

Many segmentation methods utilize graph-based algorithms, such as the work of Moore et al. called superpixel lattices [11], or SLIC & comparison from Achanta et al. [12], and a variant of SLIC combines with affinity propagation clustering called SLICAP to improve performance on boundary-based and region-based criteria. More recent approach includes Yang et al. superpixel segmentation with fully convolutional networks [13], which attempt to incorporate the method using deep neural nets.

Fig. 3. Outdoor & indoor scene segmentation results produced by Felzenszwalb algorithm with $\sigma = 0.8$, $k = 300$

This thesis would rely on the image's graph-based representation, which develops an efficient Felzenszwalb segmentation algorithm [14] to extract our image's structure representation. Details on this algorithm will be discussed later in our thesis.

2.2 Generative Adversarial Networks (GANs)

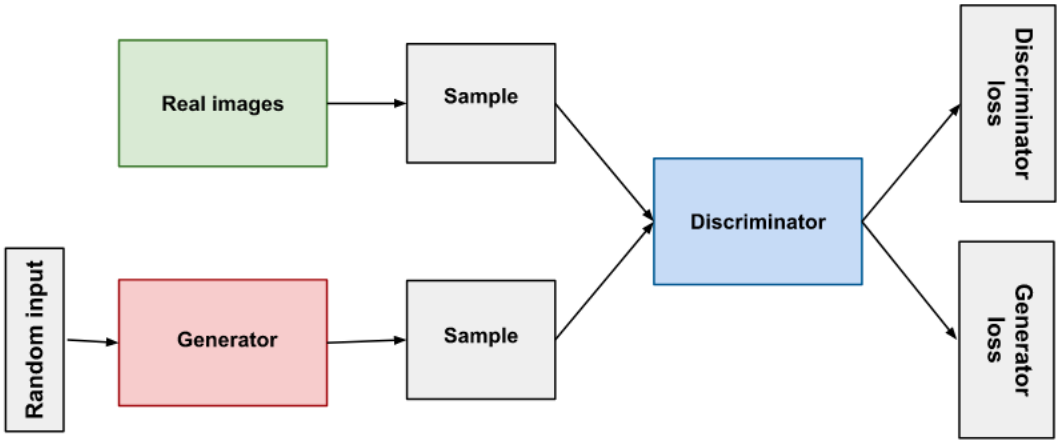


Fig. 4. Basic of GAN's architecture

Generative Adversarial Networks (GANs) architecture was first introduced by Goodfellow et al. in 2014 in a paper of the same name [1]. The initial idea of their work is to build a model that could estimate the generation process using adversarial networks.

In recent years, GANs have quickly become a state-of-the-art data generation method that has achieved impressive results. The solution lies in the adversarial training between the generator and the discriminator. In **Fig. 4**, we have a simple representation of a GAN's architecture, where both the generator and the discriminator are neural networks. They communicate by the propagation and back-propagation algorithms to update weights and losses in order to learn.

A GAN structure contains two separately trained networks: the generator (*G*), which learns to create fake data while continuously receiving feedback from the discriminator; the discriminator (*D*) is basically a binary classifier trying to separate real and generated data. The authors first define a prior on input noise variables $p_z(z)$ to learn the generator's distribution p_g over data *x*. A mapping to data space $G(z;\ \theta_g)$ is introduced, where *G* is a differentiable function represented by a multilayer perceptron with parameters θ_g. About the discriminator, there is also a second multilayer perceptron $D(x;\ \theta_d)$ that

outputs a single scalar representing the probability that *x* came from the data rather than p_g.

Their target is to train *D* so as to maximize the probability of correctly classified examples from training data and the *G*. Simultaneously, *G* is trained to minimize $\log\Big(1 - D\big(G(z)\big)\Big)$.

In conclusion, GANs is basically a min-max game between the *G* and *D* with value function $V(D,G)$, where $\mathbb{E}_{x\sim p_{data}(x)}[\log D(x)]$ means the expected value of $\log D(x)$ given *x* distributed as $p_{data}(x)$:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}\Big[\log\Big(1 - D\big(G(z)\big)\Big)\Big]$$

This is what we call adversarial training. We will go into more details in the *G* and *D* architecture in our methodology explanation later. GANs' results are impressive and extend to various fields of image processing and multiple types of art styles. Here we have some examples of character creation using GANs by Jin et al. in **Fig. 5**. You can see that the generated characters' faces look pretty authentic, like those from an art studio [15]. Although there are still limits on human expression and viewpoint, GANs are still a great choice to implement our method for image-to-image translation.

Fig. 5. Character creation using GANs

Following the success of Goodfellow's GAN model, many extensions have arisen and left their marks in the image processing field. Deep Convolutional Generative Adversarial Network, DCGAN, attempts to combine the ideas of convolutional nets with unsupervised learning [16]. It introduces the constraints on the model to develop a high-quality generator, which then becomes the solid base for even more GANs' extensions and applications. Conditional GANs, or cGANs, take extra information in addition to the image as input to train the generator and the discriminator [17]. Progressive GANs

introduce a new method to train GANs model that involves progressively increasing the model depth during the training process [18]. We choose to use GANs for our study because it is indeed powerful for image generation projects, especially domain and style translation tasks.

2.3 Non-Photorealistic Rendering (NPR)

Non-Photorealistic Rendering (NPR) is a computer graphic area that focuses on various styles of digital art. NPR algorithms, especially Neural Style Transfer, have been developed to generate/translate images with different artistic styles, such as painting, drawing, animation, and architecture illustration, etc. Some common 2D artistic image rendering techniques include extracting outlines and silhouettes to get sufficient shape information; pen-and-ink painting with strokes, tone, texture, and outlines to emphasize essential features. **Fig. 6** describes a non-photorealistic rendering from a landscape picture to some oil-painting style.

Fig. 6. Original picture and non-photorealistic representation of a lake

A variety of methods have been developed to create images with flat shading, mimicking cartoon styles. One method used is image filtering [19], but applying filtering uniformly to the entire image has side effects, such as reducing abstraction, losing details, or making object boundaries clear. For improved results, semantic segmentation of images can be applied to specific types, such as portraits. Unfortunately, this method does not perform well on different kinds of images, which means poor generalization.

Neural Style Transfer (NST) is a well-known method to perform non-photorealistic rendering. Its task is to change the style of an image in this domain to the style of an

image in another domain, so that the origin image can adopt the style of the other. NST methods synthesize images with artistic style by combining the content of one image and the style of another image. Recent studies on NST show that the VGG network trained for object recognition has the ability to extract semantic features of objects [20]. **Fig. 7** describes the architecture of a VGG-16 convolutional neural network, which is 16 layers deep.

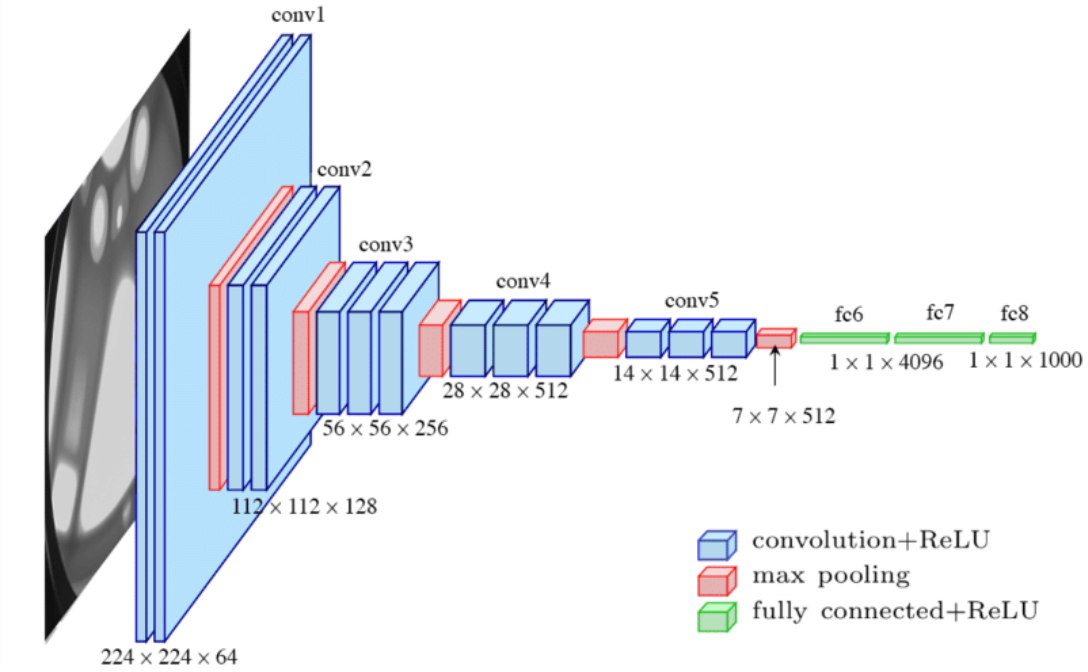


Fig. 7. Architecture of a VGG-16 network

There are already many impressive implementations of neural style transfer in animation translating. Gatys et al. first proposed an NST approach based on CNNs that transfers the style from the style image to the content image [21]. Chen et al. introduce CartoonGAN, which trained on unpaired data with several impactful losses for general photo cartoonization [3]. Their work uses a simple patch-level discriminator with fewer parameters in *D*, features edge-promoting adversarial loss, and L1 sparse regularization of high-level feature maps in the VGG network for content loss. He et al. propose three constraints based on three drawing techniques focusing on voids, brush strokes, and ink wash tone to translate images to Chinese ink wash paintings and call it ChipGAN [22].

Meanwhile, CariGANs [23] and WarpGAN [24] use different approaches to perform geometric exaggeration and appearance stylization to generate caricatures.

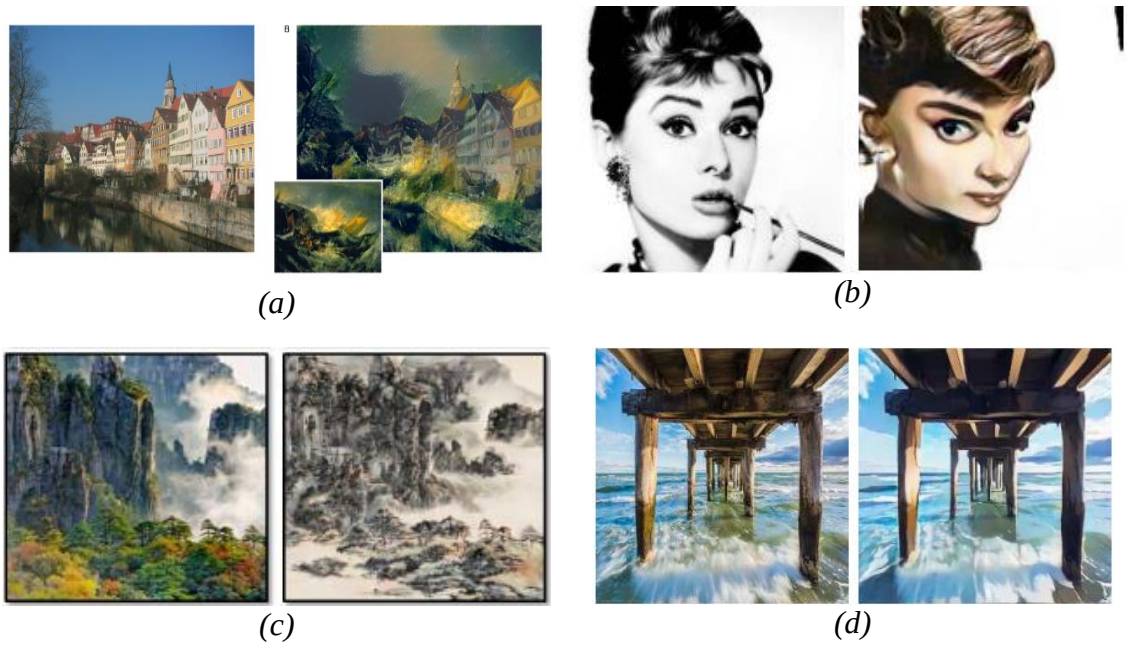


Fig. 8. Sample result from famous research: (a) Gatys's NST, (b) WarpGan, (c) ChipGan, (d) CartoonGAN with Shinkai style

However, many methods like these above can only perform in a specific art field completely different from animation. Most of them work by highlighting semantic edges and filtering out image details. Meanwhile, our method learns the cartoon data distribution from a set of cartoon images to synthesize high-quality animation on diverse cases.

2.4 Unpaired Image-to-Image Translation

Image-to-Image Translation (I2I) focuses on translating images from a source domain to another target domain while preserving the content representation. This field has drawn increasing attention in recent years and made fantastic progress, thanks to its wide range of applications in many image processing and computer vision problems, such as image synthesis, image segmentation, impainting and restoration, image colorization, and style transfer, etc.

In order to perform domain translating, we need to train a mapping $G_{A \mapsto B}$ that generates image $x_{AB} \in B$ from the input source image $x_A \in A$ so as it is similar to the target domain of image $x_B \in B$. We can express the translation process in mathematical function as:

$$x_{AB} \in B : x_{AB} = G_{A \mapsto B}(x_A)$$

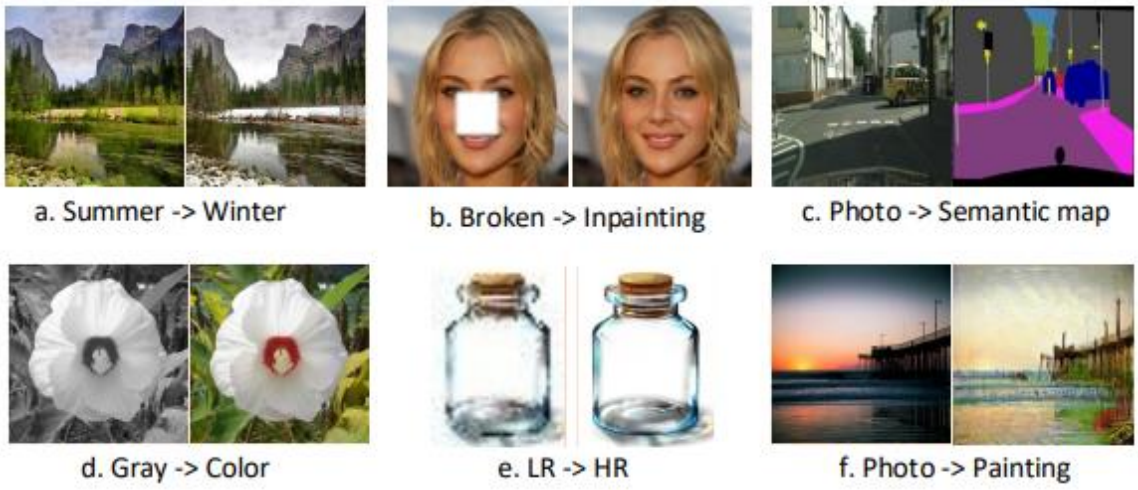


Fig. 9. Image-to-image translation applications in various graphic problems

Image-to-image translation can be applied in a wide range of graphic problems. Some common examples described in **Fig. 9** are domain translation (a), image recovery (b), semantic segmentation (c), image coloring (d), super-resolution (e), and artistic style transfer (f).

Combined with unpaired data, this approach becomes a general-purpose solution for many image processing tasks. Johnson et al. utilize and apply it to stylize photos into paints faster than the earlier optimization-based method [25]. Zhu et al. introduced CycleGAN with cycle-consistent loss and full-cycle transform [4], which uses bi-directional models to transform unpaired images of various styles. Their methods are extended and modified in many later studies. UNIT model, introduced by Liu et al., creates a shared-latent space by mapping source domain and target domain images to learn the joint distribution between them in an unsupervised manner [26]. Its framework is also based on both GANs and variational autoencoders VAEs. A multi-model version of UNIT, MUNIT can produce more diverse outputs [27] by incorporating Adaptive Instance Normalization (AdaIN) parameters and layers into the style decoder.

Meanwhile, Kim et al. involve attention mechanisms in UGATIT by generating an attention map and proposing an Adaptive Layer Instance Normalization (AdaLIN) to adaptively constrain the variation in shapes and textures [28]. Whitebox Catoonization by Wang and Yu propose representations filtering from images and different losses on each representation to improvise the result on domain translation [35]. On the other hand, project FUNIT, Few-shot Image-to-Image Translation, has tried to overcome style controllability caused by overlapped encoded instances by utilizing a few-shot classification for controlling the categories of output images [29]. **Fig. 10** below shows the results of different methods on the face2anime dataset.

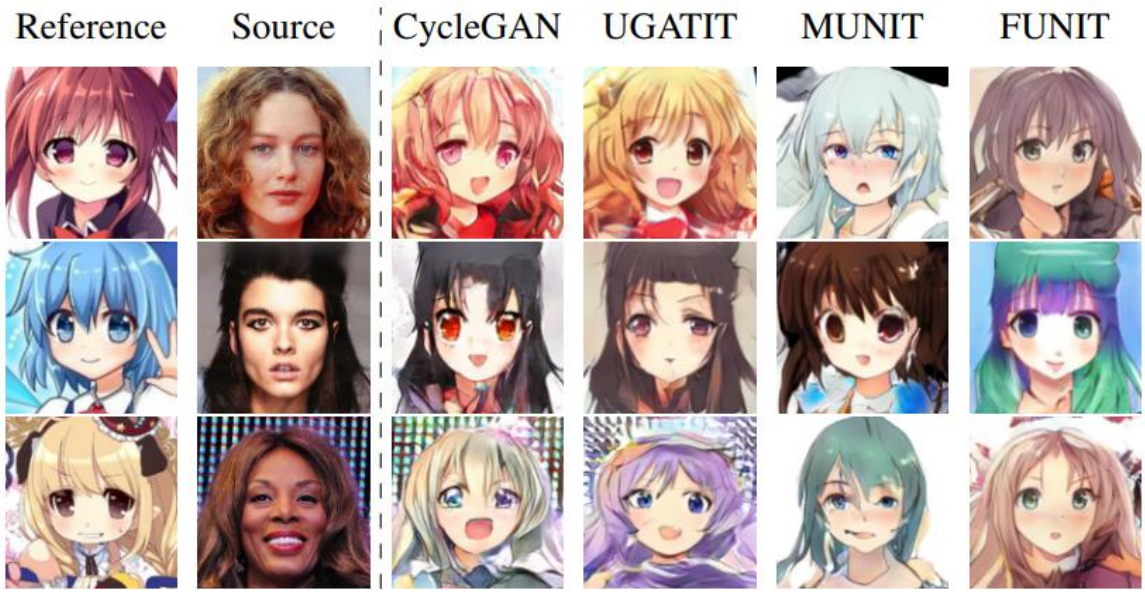


Fig. 10. Performances of different models on the face2anime dataset

All of the above studies prove that interdomain translation is always an attractive field. Many interesting and effective methods were proposed to solve various problems in style-transferring and image translation. However, some problems still require answers, such as unclean results caused by outliers [37], insufficient data [4], or poor style generalization caused by partial images segmentation of specific types.

An independent anime dataset was collected to conduct this study. We also utilize new algorithms to calculate texture loss, aiming to reduce noises and unwanted edges that many methods have faced. Experiments with different variables also show promising translation results, which are then fed to the evaluation metric for comparison. The

obtained results demonstrate that this model is lighter, can perform style transferring faster and easier than other models.

CHAPTER 3

METHODOLOGY

Our GAN framework contains two types of 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 design the generator and discriminator networks to suit the particularity of cartoon images and can be easily fine-tuned and modified.

Our Generator is a UNet-based generator capable of generating cartoon images in a short amount of time. More specifically, the UNet is a convolutional network architecture for fast and precise segmentation of images. The architecture is symmetric and consists of two major parts of contracting networks by successive layers, where upsampling operators replace the right part. Hence these layers can then learn to assemble a precise output from what they get. A simple representation of the Generator architecture is shown in **Fig. 11** below.

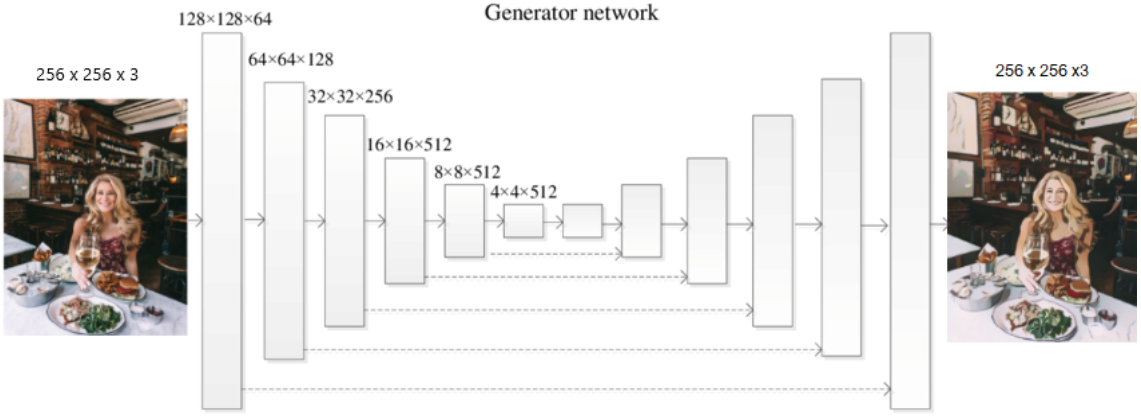


Fig. 11. Generator architecture

After going through the Generator, images are decomposed into the surface representation, the structure representation, and the texture representations. Three independent discriminators and losses are also proposed to extract information as described in **Fig. 12**.

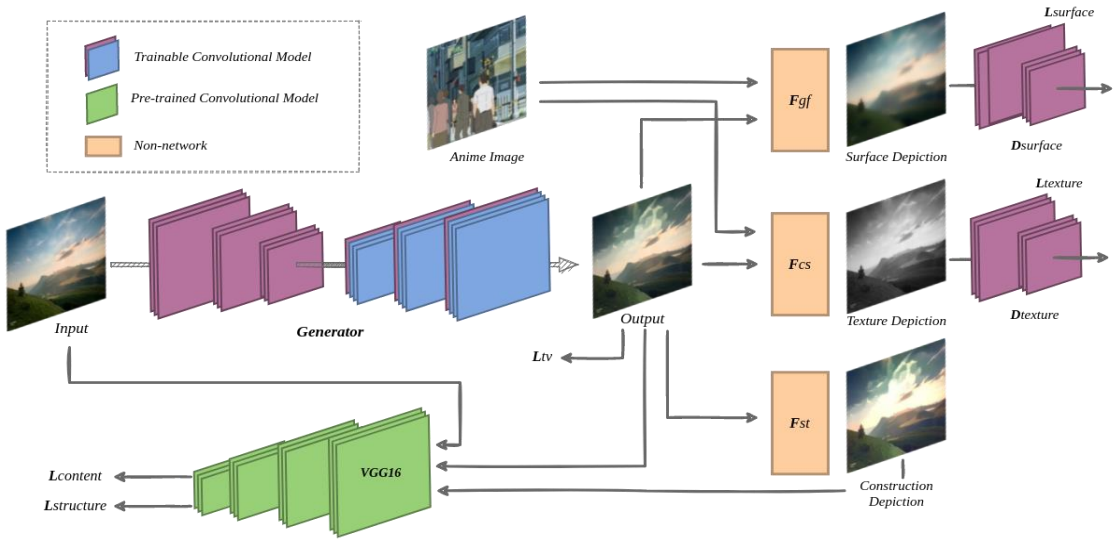


Fig. 12. Model architecture

To be more specific, surface loss aims to distinguish between surface representation extracted from model outputs and cartoons. Texture loss is used to differentiate input photos and outputs between texture representations extracted from outputs and cartoons. A pre-trained VGG network [20] is used as the discriminator to extract high-level features and impose spatial constraints on global contents between extracted structure representations and outputs.

3.1 Structure Loss

The structure loss aims to imitate the animated style of clear edge, high-level simplification and abstraction, and sparse color blocks. After experiencing different methods, we achieved high-performance results with Felzenszwalb and Huttenlocher segmentation and hierarchical grouping, as shown in **Fig. 13** [14, 30]. In the first step, we use Felzenszwalb algorithm to segment the image into separate regions. After that, a greedy algorithm works to group those regions together iteratively. Now we would have the similarities between all neighboring regions calculated.

The final result would be the two most similar regions grouped together, and new similarities are calculated between the resulting region and its neighbors. The process of grouping the most similar regions is repeated until the remaining regions are equal to the given segmentation number.

Input: (color) image
Output: Set of object location hypotheses *L*
Obtain initial regions $R = \{r_1, \cdots, r_n\}$
Initialize similarity set $S = \emptyset$
foreach *Neighboring region pair* $(r_i,\ r_j)$ **do**:
  Calculate similarity $s\ (r_i,\ r_j)$
  $S = S \cup s\ (r_i,\ r_j)$
While $S\ \neq n$ **do**:
  Get highest similarity $s\ (r_i,\ r_j) = \max(S)$
  Merge corresponding regions $r_t = r_i \cup r_j$
  Remove similarities regarding $r_i : S = S \setminus s\ (r_i,\ r_x)$
  Remove similarities regarding $r_j : S = S \setminus s\ (r_j,\ r_x)$
  Calculate similarity set S_t between r_t and its neighbors
  $S = S \cup S_t$
  $R = R \cup r_t$
Extract object location boxes *L* from all regions in *R*

Fig. 13. Hierarchical Grouping Algorithm

In order to enforce spatial constraints between the result and the extracted structure representation, we use a pre-trained VGG-16 feature extractor as a structure discriminator. Let F_{sr} be the extracted structure representation, and the final loss is formulated as:

$$L_{structure} = \left\| VGG\big(G(I_x)\big) - VGG\left(F_{sr}\big(G(I_x)\big)\right) \right\|$$

where F_{sr} is the structure extraction filter, and I_x is the input image to the generator *G*.

3.2 Surface Loss

The surface loss will try to force the model to learn the cartoon painting style where artists usually draw drafts with coarse brushes and have smooth surfaces similar to cartoon

images. In order to do that, we use a differentiable guided filter from JoyceMar et al. [36] for edge-preserving filtering. We can express the surface loss as:

$$L_{surface} = \log D_s\left(F_{gf}(I_y, I_y)\right) + \log\left(1 - D_s\left(F_{gf}(G(I_x), G(I_x))\right)\right)$$

where the F_{gf} is the smooth filter, which will take an image as input and return a smooth, blur version. $F_{gf}(I_x, I_y)$ is the image with texture and details removed, with I_x is the input photo, and I_y represents the reference cartoon images. A simple discriminator D_s is used to decide if the generated output has the same surface as the animated picture to help the generator *G*.

3.3 Texture Loss

Along with the color and luminance factor, which have been focused on with the two losses above, cartoon styles also have unique characteristics with high-level simplification and high-frequency features that can be treated as key objectives that can make it easy to distinguish between cartoon images and real-world photos. Due to this, we propose using a simple random color shift algorithm F_{cs} to convert the image to a grayscale feature map that still contains information about all the high-frequency textures:

$$I_{grayscale} = \frac{\beta_1 \cdot I_r + \beta_2 \cdot I_b + \beta_3 \cdot I_g}{\beta_1 + \beta_2 + \beta_3}$$

where $I_{grayscale}$ is our image in grayscale, I_r, I_b, and I_g are images in three channels RGB, and $\beta_1, \beta_2, \beta_3$ are trainable parameters. We also use a simple discriminator D_t to separate the grayscale feature map extract from the generated image and the animation image, with F_{cs} being the color shift filter.

$$L_{texture} = \log D_t\left(F_{cs}(I_y)\right) + \log\left(1 - D_t\left(F_{cs}(G(I_x))\right)\right)$$

3.4 Total-Variant Loss and Superpixel Loss

In order to make the content of the generated photo to be of high quality, the total-variation loss L_{tv} is used to impose spatial smoothness on generated images and reduce

high-frequency noises such as salt-and-pepper noise. For *H, W, C* represent spatial dimensions of images, we have:

$$L_{tv} = \frac{1}{H \cdot W \cdot C} \left\| \nabla_x \left(G(I_x) \right) + \nabla_y \left(G(I_x) \right) \right\|$$

Finally, we propose a superpixel loss L_{sp} to maintain important content from the input photo, which ensures that the cartoonized results and input photos are semantically unchanged. We also use a pre-trained VGG-16 model to calculate it, similar to the structure loss:

$$L_{sp} = \left\| VGG\left(G(I_x) \right) - VGG(I_x) \right\|$$

3.5 Full Model

With all of the losses mentioned above, we can write our final loss function as:

$$L_{generator} = \beta_1 \cdot L_{tv} + \beta_2 \cdot L_{surface} + \beta_3 \cdot L_{structure} + \beta_4 \cdot L_{texture} + \beta_5 \cdot L_{sp}$$

where the parameter $\beta_1, \beta_2, \beta_3 \cdots$ can be changed for separate uses.

CHAPTER 4

EXPERIMENTAL RESULTS

4.1 Implementation

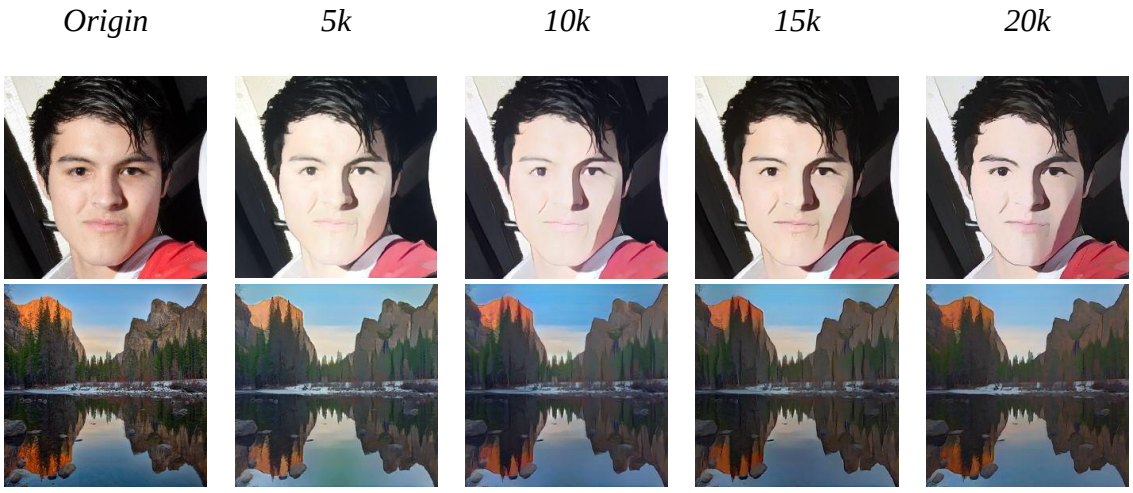


Fig. 14. Model result after 5k-10k-15k-20k training iterations

This GAN model is implemented in Tensorflow [31]. The trained models in our experiments are available to facilitate the evaluation of future methods. All experiments were performed on an NVIDIA 1060Ti GPU. We use Adam algorithms with a learning rate of 1.5×10^{-4} [32] and train the model with batch size 16 for 20000 iterations. **Fig. 14** describes our training progress with translated pictures of human faces and outdoor scenery after 5000 to 20000 training iterations. As the figure shows, our model can be trained quite fast and reach the state-of-the-art lever in just 20000 iterations.

4.2 Dataset

The training dataset contains four folders, real and cartoon images for scenery and human face. The cartoon data is collected manually by us, which we crop out from real anime videos, mainly from Shinkai Makoto's films. For the cartoon data, we use 10000 for scenery and 5000 for human faces. In terms of the real-world data, we use 10000 scenery images crawled from the Internet and 5000 human faces from the FFHQ dataset [33]. And for the validation dataset, we use 1000 real-life images and 1000 cartoon images collected from the Internet. Some examples from our dataset are shown in **Fig. 15** below.

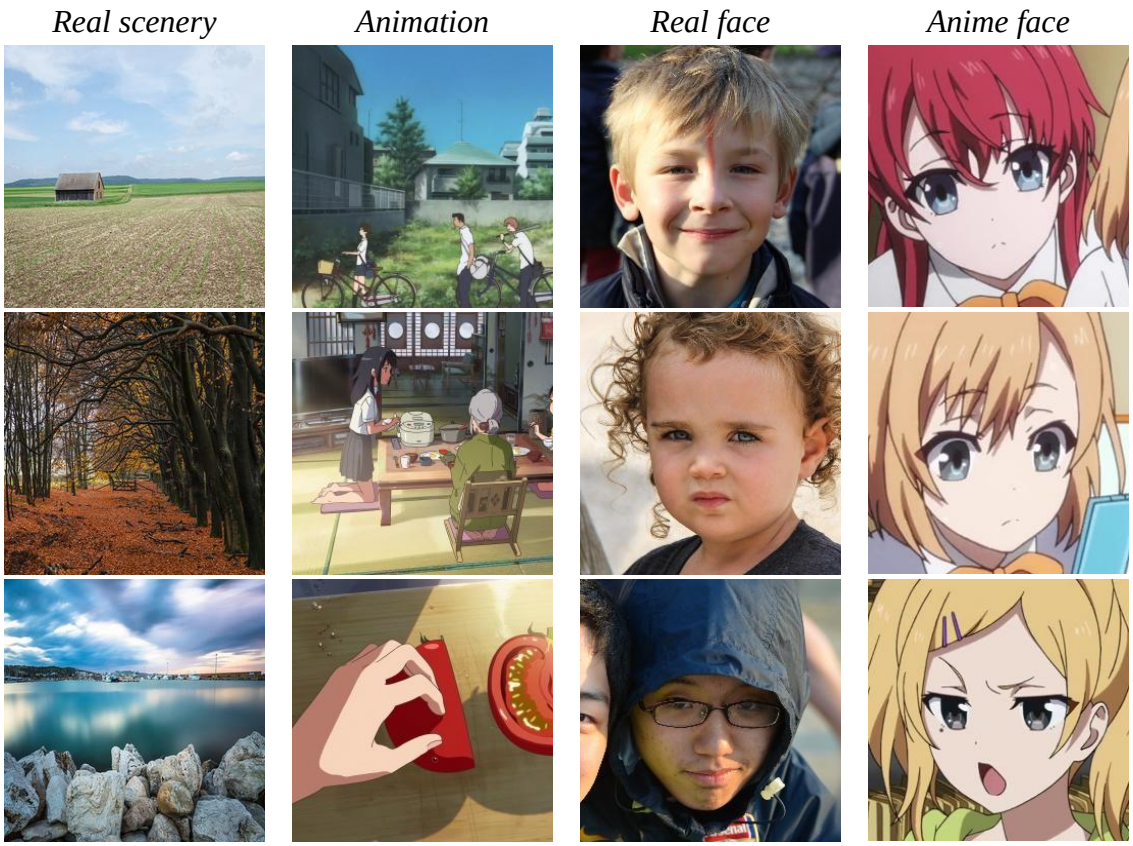


Fig. 15. Example images from our dataset

4.3 Time Performance and Model Size

Our method has a relatively low number of parameters and running time. On our GPU, we could reach the time of 17ms to process a 720*1280 image, which is much faster than other related works and can be totally capable of real-time high-resolution video processing tasks. Our model only has about 1.5 million parameters with the size of 5.6MB, which can be used to deploy on mobile apps. The table below will show some comparison with other methods in terms of speed and the number of parameters.

Methods	**AnimeGan[2]**	**CartoonGan[3]**	**CycleGan[4]**	**Ours**
HR, GPU(ms)	45.53	148.02	106.82	17.23
Parameter(million)	3.96	11.38	11.13	1.48

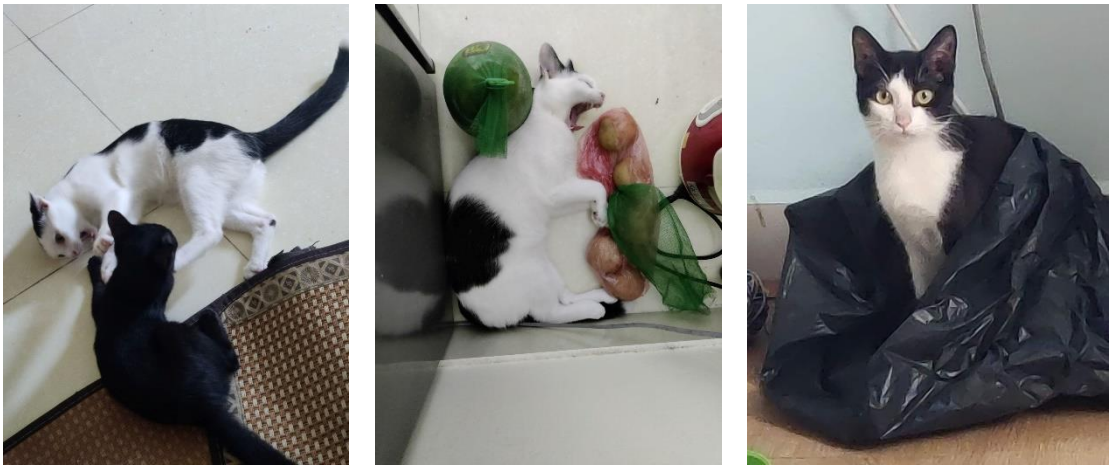
Table 1. Parameters and Time Performance comparison

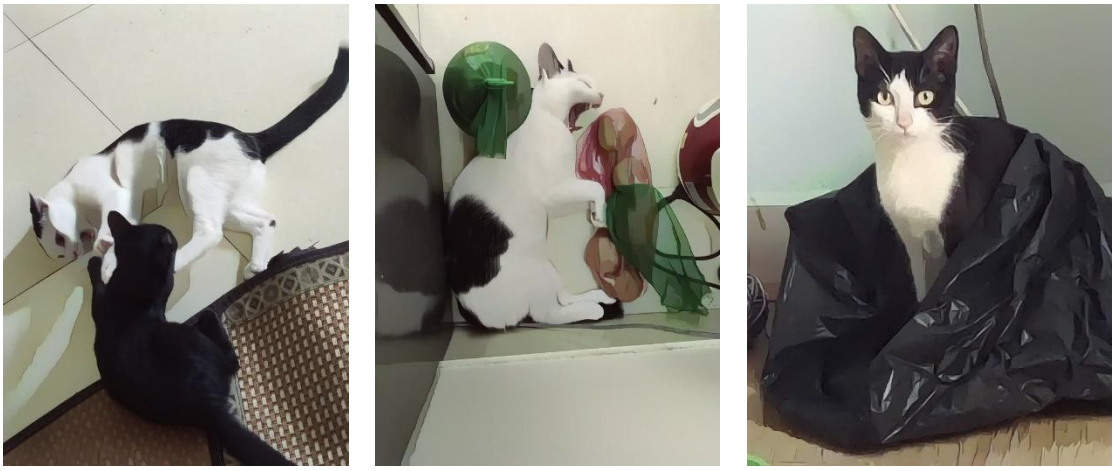
4.4 Evaluation metrics

For qualitative evaluation, this thesis will present results for different objects compared to the results from other relevant research. Besides, Frechet Inception Distance (FID) is proposed for quantitative evaluation to compare the generated images with the target images [34].

4.5 Result demo

Human portraits





Cat pictures

Food pictures






Scenery pictures

Street pictures

Fig. 16. Model results on different categories

As shown out above in **Fig. 16**, our model could indeed be general to diverse use cases. We apply the model in different real-world scenery, including human faces, animals, foods, and city street images.

4.6 Qualitative Comparison

Fig. 17. Original images with their results using different model

Comparison between our method and some related works are shown in **Fig. 17**. CartoonGan generates quality results with good texture and clear edges but lacks abstraction and tends to distort colors. Our model, on the other hand, prevents improper color modifications. AnimeGan generates darkened images and cannot really show out the Cartoon/Anime style. The WhiteBox model has really clear boundaries but focuses a little too much on smoothing the image with color blocks and enhancing edges, which leads to the lack of small details from the original image and creates noises. Finally, our model is not perfect, but it gives a balanced result in color, texture and still generates cartoon feel-like images. To conclude, our method outperforms previous methods in generating images with harmonious color, clean edges, fine details, and less noise.

4.7 Quantitative Comparison

Method	Real Photo	CartoonGAN[3]	AnimeGAN[2]	WhiteBox[35]	Ours
FID to Cartoon	160	125	130	118	110

Table 2: Performance evaluation based on FID

FID is widely used to evaluate the quality of synthesized images quantitatively. Therefore, our work proposes Heusel's method to calculate the distance between two image distributions [34], namely the generated and actual anime images. FID results used to calculate this study's performance and performance of related works are shown in **Table 2**. This method generates images with the smallest FID score, proving that it has the most Cartoon/Anime style result, outperforming the others.

4.8 Illustration of Controllability

Fig. 18. Output result controlled by changing opponents in the loss function

As shown in **Fig. 18**, we can adjust the cartoonized results' style by tuning the weight of each representation in the loss function. By choosing a higher weight for texture representation, our model could add more small details to the result; details such as grassland, stones, or cloud curves are well preserved. The explanation is our model could maintain and highlight high-frequency details stored in texture representation. In contrast, fewer details and smoother images are generated with a higher weight of surface representation. The reason is the model is now focusing more on the smooth picture created by the guided filter. Finally, to achieve more abstract and sparse images, we can increase the weight of structure representation, and the details will be well-segmented into sparse color blocks. The selective search algorithm has flattened the training data and abstracted them into structure representations. In conclusion, our white-box method is indeed controllable and can be easily adjusted, unlike black-box models.

4.9 Analysis of Each Component

Original photo (a)W/O Texture Loss (b)W/O Structure Loss (c)W/O Surface Loss Full Model

Fig. 19. Results of removing components in the loss function

We show the results of the ablation study in **Fig. 19**. The ablated texture representation will cause confusion in details. As shown in **Fig. 19** (a), the irregular textures on the grass and dog legs still exist. This is because of the lack of high frequencies stored in the surface representation, which reduces the cartoonization ability of the model. The ablation structure representation will cause the high frequency noise in **Fig. 19** (b). A serious taste of salt and pepper appears on the grasslands and mountains. The reason here is that the structure representation flattens the image and removes high-frequency information. The ablated surface representation can cause noise and messy details. **Fig. 19** (c) shows unclear cloud edges and grass noise. The reason for this is that the guided filtering process suppresses high-frequency information and preserves a smooth surface. As a comparison, the result of our complete model is shown in **Fig. 19** (d), which has smoother features, clearer boundaries, and much less noise. To sum up, all three representations have their own contribution to the result of our method.

CHAPTER 5

CONCLUSION

This thesis proposes a lightweight and controllable approach for image cartoonization by translating actual footage into animation. Frechet Inception Distance is used as our evaluation metric to compare the generated images with the target images and receive promising results. The methodology we choose pays close attention to the animation painting process, encouraging us to distinguish three separate feature maps from generated pictures for independent handling. The Generator will continuously learn the cartoonization process from five different losses of surface representation, construction depiction, texture representation, and VGG based discriminators. The losses can be easily modified for personal improvement. Moreover, unpaired datasets are used for training so that it can be more easily fine-tuned for different art styles. We have also created our own dataset and used it along with available ones for the training process. The final experiments demonstrate that this model can generate better results by surpassing performance compared to many state-of-the-art models.

In future works, we would like to extend the application of this method on videos to generate smooth, anime-like cuts. Details on portrait and facial expression are also considered for improvement so that the character's emotion and sentiment would be more well-described in further research.

REFERENCES

1. Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative Adversarial Networks. arXiv preprint arXiv:1406.2661 (2014)

2. Chen, J., Liu, G., Chen, X.: AnimeGAN: A Novel Lightweight GAN for Photo Animation. In: Li K., Li W., Wang H., Liu Y. (eds) Artificial Intelligence Algorithms and Applications. ISICA 2019. Communications in Computer and Information Science, vol 1205. Springer, Singapore (2020).

3. Chen, Y., Lai, Y., Liu, Y.: CartoonGAN: Generative Adversarial Networks for Photo Cartoonization. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 9465-9474, doi: 10.1109/CVPR.2018.00986. (2018)

4. Zhu, J., Park, T., Isola, P., Efros, A. A.: Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. arXiv preprint arXiv:1703.10593 (2017)

5. Pęśko, M., Svystun, A., Andruszkiewicz, P., Rokita, P., Trzciński, T.: Comixify: Transform video into a comic. arXiv preprint arXiv:1812.03473 (2018)

6. Tomasi, C., Manduchi, R.: Bilateral Filtering for Gray and Color Images. In: Proceedings of the 1998 IEEE International Conference on Computer Vision, Bombay, India (1998)

7. He, K., Sun, J., Tang, X.: Guided Image Filtering. IEEE transactions on pattern analysis and machine intelligence. 35. 1397-1409. 10.1109/TPAMI.2012.213 (2013)

8. Farbman, Z., Fattal, R., Lischinski, D., Szeliski, R.: Edge-Preserving Decompositions for Multi-Scale Tone and Detail Manipulation. ACM Trans. Graph. 27. 10.1145/1360612.1360666 (2008)

9. Min, D., Choi, S., Lu, J., Ham, B., Sohn, K., Do, M. N.: Fast Global Image Smoothing Based on Weighted Least Squares. In: IEEE Transactions on Image

Processing, vol. 23, no. 12, pp. 5638-5653, Dec. 2014, doi: 10.1109/TIP.2014.2366600. (2014)

10. Wu, H., Zheng, S., Zhang, J., Huang, K.: Fast End-to-End Trainable Guided Filter. arXiv preprint arXiv:1803.05619 (2018)

11. Moore, A. P., Prince, S. J. D., Warrell, J., Mohammed, U., Jones, G.: Superpixel Lattices. 2008 IEEE Conference on Computer Vision and Pattern Recognition, 2008, pp. 1-8, doi: 10.1109/CVPR.2008.4587471. (2008)

12. Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., Süsstrunk, S.: SLIC Superpixels Compared to State-of-the-artSuperpixel Methods. IEEE Transactions on Pattern Analysis and Machine Intelligence 34(11), doi: 10.1109/TPAMI.2012.120. (2012)

13. Yang, F., Sun, Q., Jin, H., Zhou, Z.: Superpixel Segmentation with Fully Convolutional Networks. arXiv preprint arXiv:2003.12929 (2020)

14. Felzenszwalb, P. F., Huttenlocher, D. P.: Efficient Graph-Based Image Segmentation. International Journal of Computer Vision 59, 167–181 (2004). https://doi.org/10.1023/B:VISI.0000022288.19776.77

15. Jin, Y., Zhang, J., Li, M., Tian, Y., Zhu, H., Fang, Z.: Towards the Automatic Anime Characters Creation with Generative Adversarial Networks. arXiv preprint arXiv:1708.05509 (2017)

16. Radford, A., Metz, L., Chintala, S.: Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. arXiv preprint arXiv:1511.06434 (2015)

17. Mirza, M., Osindero, S.: Conditional Generative Adversarial Nets. arXiv preprint arXiv:1411.1784 (2014)

18. Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive Growing of GANs for Improved Quality, Stability, and Variation. arXiv preprint arXiv:1710.10196 (2017)

19. Winnemoller, H., Olsen, S. C., Gooch, B.: Real-Time Video Abstraction. ACM Trans. Graph. 25. 1221-1226. 10.1145/1179352.1142018 (2006)

20. Simonyan, K., Zisserman, A.: Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv preprint arXiv:1409.1556 (2014)

21. Gatys, L. A., Ecker, A. S., Bethge, M.: A Neural Algorithm of Artistic Style. arXiv preprint arXiv:1508.06576 (2015)

22. He, B., Gao, F., Ma, D., Shi, B., Duan, L.: ChipGAN: A Generative Adversarial Network for Chinese Ink Wash Painting Style Transfer. In: Proceedings of the 26th ACM international conference on Multimedia (2018)

23. Cao, K., Liao, J., Yuan, L.: CariGANs: Unpaired Photo-to-Caricature Translation. arXiv preprint arXiv:1811.00222 (2018)

24. Shi, Y., Deb, D., Jain, A. K.: WarpGAN: Automatic Caricature Generation. arXiv preprint arXiv:1811.10100 (2018)

25. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual Losses for Real-Time Style Transfer and Super-Resolution. arXiv preprint arXiv:1603.08155 (2016)

26. Liu, M., Breuel, T., Kautz, J.: Unsupervised Image-to-Image Translation Networks. arXiv preprint arXiv:1703.00848 (2017)

27. Huang, X., Liu, M., Belongie, S., Kautz, J.: Multimodal Unsupervised Image-to-Image Translation. arXiv preprint arXiv:1804.04732 (2018)

28. Kim, J., Kim, M., Kang, H., Lee, K.: U-GAT-IT: Unsupervised Generative Attentional Networks with Adaptive Layer-Instance Normalization for Image-to-Image Translation. arXiv preprint arXiv:1907.10830v4 (2019)

29. Liu, M., Huang, X., Mallya, A., Karras, T., Aila, T., Lehtinen, J., Kautz, J.: Few-Shot Unsupervised Image-to-Image Translation. arXiv preprint arXiv:1905.01723 (2019)

30. Sande, K. E. A. van de, Uijlings, J. R. R., Gevers, T., Smeulders, A. W. M.: Selective Search for Object Recognition. International Journal of Computer Vision, 104(2):154–171 (2013)

31. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., Zheng, X.: TensorFlow: A system for large-scale machine learning. arXiv preprint arXiv:1605.08695 (2016)

32. Kingma, D. P., Ba, J.: Adam: A Method for Stochastic Optimization. arXiv preprint arXiv:1412.6980 (2014)

33. Karras, T., Laine, S., Aila, T.: A Style-Based Generator Architecture for Generative Adversarial Networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4401–4410 (2018)

34. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In: Advances in Neural Information Processing Systems, pages 6626–6637 (2017)

35. Wang, X., Yu, J.: Learning to Cartoonize Using White-Box Cartoon Representations. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2020)

36. JoyceMar, G. J., Begum, A. Rijuvana: Guided Filter Smoothing for Third Order Edge Mask. International Journal of Computer Applications. 120. 36-42. 10.5120/21403-4424 (2015)

37. Lei J.: AnimeGAN. https://github.com/jayleicn/animeGAN, 2017