Dalian University of Technology
Undergraduate Capstone Project (Thesis)

Image Colorization Based on Generative Adversarial Network

Department: International School of Information Science & Engineering

Major: Software Engineering

Name: Xu Shizhuo

Student Number: 201694092

Supervisor: Prof. Li Haojie

Review Teacher: Prof. Wang Shengfa

Completion Date: 2020-06-10

大连理工大学

Dalian University of Technology

原创性声明

本人郑重声明：本人所呈交的毕业设计（论文），是在指导老师的指导下独立进行研究所取得的成果。毕业设计（论文）中凡引用他人已经发表或未发表的成果、数据、观点等，均已明确注明出处。除文中已经注明引用的内容外，不包含任何其他个人或集体已经发表或撰写过的科研成果。对本文的研究成果做出重要贡献的个人和集体，均已在文中以明确方式标明。

本声明的法律责任由本人承担。

作者签名：徐仕卓　　日　期：　2020 年 6 月 10 日

关于使用授权的声明

本人在指导老师指导下所完成的毕业设计（论文）及相关的资料（包括图纸、试验记录、原始数据、实物照片、图片、录音带、设计手稿等），知识产权归属大连理工大学。本人完全了解大连理工大学有关保存、使用毕业设计（论文）的规定，本人授权大连理工大学可以将本毕业设计（论文）的全部或部分内容编入有关数据库进行检索，可以采用任何复制手段保存和汇编本毕业设计（论文）。如果发表相关成果，一定征得指导教师同意，且第一署名单位为大连理工大学。本人离校后使用毕业毕业设计（论文）或与该论文直接相关的学术论文或成果时，第一署名单位仍然为大连理工大学。

论文作者签名：徐仕卓　　日　期：2020 年 6 月 10 日

指导老师签名：　　日　期：2020 年 6 月 10 日

摘　要

漫画作为一种艺术表现形式，以其独特的表现力受到了大众的喜爱。色彩更是一幅漫画作品中不可或缺的一环，与其说色彩是一副成功动漫作品的华丽外衣，不如说色彩是动漫作品的灵魂。色彩给动漫作品赋予了灵魂，将动漫作品中的人物、表情、动作表现的淋漓尽致。

但完成这项工作并非易事，它既需要上色者对色彩具有敏锐的感知力、一定的绘画基础，还需要一定的时间，因此线稿图像的自动上色是一个具有重要市场需求的研究领域。本文针对该问题提出了一种基于对抗生成网络的线稿上色方法，主要内容如下：

采用“分治法”的算法思想，将上色任务分阶段进行。在第一阶段模型中，网络根据学习到的特征，在对应的线稿区域泼洒各种丰富的颜色从而获得一个彩色草稿图。之后在第二个阶段模型中，针对这个彩色草稿图进行校正，它会检测到不自然的颜色和人为痕迹，并试图修正和完善。同时为了使上色的效果更加优越，对对抗生成网络进行了进一步优化，比如，使用缩放卷积来解决棋盘效应，优化损失函数等。通过 Python 爬虫技术抓取了 14 万张彩色漫画，使用神经网络方法提取手绘图作为训练模型的数据。通过图像处理技术模拟上色过程中的错误上色，生成纠正阶段所需的数据集。

关键词：上色；线稿；神经网络；媒体艺术；

Abstract

As a form of artistic expression, cartoon is loved by the public with its unique expression. Color is an indispensable part of a line art colorization work. Color is not so much the gorgeous coat of a successful line art colorization work as the soul of a line art colorization work. Color gives soul to line art colorization works, and the characters, expressions and movements in line art colorization works are vividly expressed.

However, it is not easy to complete this work, which requires the comic artist to have a keen sense of color, a certain painting basis, but also a certain amount of time. Therefore, the automatic coloring of line art colorization is an important research field with market demand. In this paper, a method of line art colorization based on Generative Adversarial Network is proposed. The main content is as follows:

Using the divide and conquer algorithm idea, the coloring tasks will be carried out in stages. In the first-stage model, the network sprinkles various rich colors in the corresponding line draft area according to the learned features to obtain a colored draft map. Afterwards, in the second stage model, the color draft is corrected, it will detect unnatural colors and artificial traces, and try to correct and improve it. At the same time, in order to make the coloring effect more superior, the GAN(generative adversarial network) was further optimized. For example, use scaling convolution to solve the checkerboard effect, optimize the loss function, and etc. Using python crawler technology to capture 140,000 color comics. The method of neural network is used to extract the data of hand drawing as training model. The image processing technology is used to simulate the wrong coloring in the coloring process, and the data set required for the correction stage is generated.

Key Words: Colorization, Sketch, Neural Network, Media art

概要（日本語）

漫画は一種の芸術的表現として、独特の表現力で大衆に愛されている。色は特に漫画作品の中で不可欠な一環である。色彩は成功したアニメ作品の華麗な上着というより、むしろアニメ作品の魂である。色はアニメ作品に魂を与えて、アニメ作品のキャラクター、表現、動きは生き生きと表現されている。

しかし、この作業を完了するのは簡単ではなく、人は色に対して鋭敏な感性を持っていて、ある程度の絵の土台を持っていて、それを完成するにはまだ時間がかかる。したがって、原稿画像画像の自動カラーリングは、市場需要がある重要な研究分野である。この問題を目指して、この論文では、敵対的な生成ネットワークに基づいてラインドラフトに色を付ける方法を提案する。主な内容は：

「分割統治」アルゴリズムのアイデアを使用して、着色タスクは段階的に実行される。第一段階のモデルでは、ネットワークが学習した特徴に応じて,対応するライン原稿領域に様々な色を撒き散らして 1 つのカラー図を得る。次に第二段階モデルでは、このカラー図に対して修正を行う、不自然な色や人の痕跡を検出し、修正や補完を試みる。同時に、カラー効果をより優れたものにするために、敵対的な生成ネットワークをさらに最適化する。たとえば、スケーリング畳み込みを使用してチェッカーボード効果を解決し、損失関数を最適化するなど。14 万枚のカラー漫画を Python 虫技術でキャッチし、ニューラルネットワーク法を用いて手書きの絵を学習モデルのデータとして抽出した。画像処理技術により、色上げの過程での色上げ誤りをシミュレートし、訂正フェーズに必要なデータセットを生成する。

キーワード：色付け、線画、ニューラルネットワーク、メディアアート

Catalogue

1 Introduction

1.1 Research background and significance

With the vigorous development of the comics and digital media industries, the emergence of various comics products is also gradually accelerating, and people's demand for comics products is no longer only satisfied with black and white line pictures. Anime works full of colors can bring a strong visual impact to consumers, attract their attention, and make products more popular with consumers. Thereby increasing the sales of their own comics and increasing their commercial value. But sketch colorization is not an easy work. Different from photo colorization which strongly relies on texture information, sketch colorization is more challenging as sketches may not have texture.

Coloring is one of the most important and time-consuming tasks in comic art creation. Creating an impressive and expressive drawing work is inseparable from an excellent color composition. However, it is not easy to complete this work. It requires both the sense of aesthetics and the experience of drawing.

Even professionals may spend a significant amount of time and effort in producing the right color composition and fine texture and shading details. Therefore, most of the comics on the market today only have a cover or the first few pages are in color. How to quickly convert sketches into color images is a problem that comic workers are troubled by.

Therefore, the automatic coloring of sketch is a research field with important market needs. An automatic or semi-automatic coloring system can bring great convenience to painters and allow ordinary people to try to touch painting and cultivate personal interest. The traditional approach is to use professional application tools, such as Photoshop and other third-party tools to complete the coloring of sketches.

However, in recent years, with the updating iterations of computing equipment and the tremendous improvement of computing power, deep learning has revolutionized research in the fields of images and texts, including the progress of image segmentation and image coloring. Deep learning simulates the nervous system of the human brain through a multi-layer neural network, and has the ability to discriminate autonomously by learning large amounts of data. It is the ability of deep learning technology to have this automatic learning feature that liberates human hands. Among the many network structures, the GAN [1] has attracted the attention of many researchers with its unique game ideas.

Since the GAN has been proposed, it has achieved good results in the field of image generation. It has also made some good progress in the field of image coloring. Therefore, the sketch colorization based on the GAN is worth studying.

1.2 The research status of image colorization

Research on image coloring technology first began in the 1970s with the use of computer-assisted technology proposed by Wilson Markle to colorize the lunar image obtained by the apollo moon project. [2] The technology used pseudo-color processing in the early stages. The method is to map a monochrome image into a color image by matching each gray level to a point on the color space. The purpose is to make the image easy to observe, adding different colors according to the grayscale of the image to highlight the details.

Later, it gradually developed to manually define the segmentation area according to the outline, and then use the brush to color. Then later, it is based on artificially designed texture, structure and other features to segment the area and then color according to the matching reference feature color distribution. Until now, the computer automatically learns abstract features from a large amount of data to segment the area and fills the corresponding colors into specific areas according to the learned feature semantics.

Existing coloring methods are mainly divided into two categories, one is coloring methods based on traditional image processing technology, and the other is coloring methods based on deep learning. Due to the results achieved color depth learning-based method is often better than traditional methods, so we mainly discuss coloring methods based on deep learning.

1.2.1 Automatic image colorization

[Cheng et al. 2015] [3] investigates into the colorization problem based on deep learning techniques which converts a grayscale image to a colorful version. Previous methods usually required manual adjustments to achieve artifact-free quality. For instance, it normally requires human-labelled color scribbles on the grayscale target image or a careful selection of colorful reference images.

But in their paper, they aim at a fully-automatic image colorization which based on deep learning techniques. They use an extremely large-scale reference database which contains sufficient color images is the most reliable solution to the image colorization problem with the assumption of a perfect patch matching technique. However, patch matching noise will increase with respect to the size of the reference database in practice.

But by the recent success in deep learning techniques which provide amazing modeling of large-scale data, their paper reformulates the image colorization problem so

that deep learning techniques can be directly employed. They proposed a post-processing step based on joint bilateral filtering, and further developed an adaptive image clustering technology to combine global image information, making its method superior to the previous algorithm.

[Yun Cao et al.2017][4] leverage conditional Generative Adversarial Networks (cGAN) to model the distribution of real-world item colors. They propose an unsupervised diversity coloring method based cGAN and develop a fully convolutional generator with multi-layer noise to enhance diversity. They use multi-layer condition concatenation to maintain reality, and with stride 1 to keep spatial information. In this method, the generator network receives two inputs, grayscale and random noise, and then outputs the uv component. And then, the grayscale and the uv component are combined to obtain a generated color image.

Generator G is given conditional information (grayscale image) together with noise z, and produces generated color channels. Discriminator D is trained over the real color image and the generated color image by G. The goal of D is to distinguish real images from the fake ones. Both nets are trained adversarially.

The generator network does not use a codec structure. In order to avoid losing spatial information in the convolution process, the first layer of the generator participates in the calculation of each subsequent layer with a jump connection, so each layer inputs random noise, which increases color diversity.

However, this method relies heavily on image texture and grayscale information. The coloring process is completely random and uncontrollable, which is prone to color overflow.

[Paulina Hensman et al. 2017] [5] propose a manga colorization method based on conditional generative adversarial networks(cGAN). Unlike previous cGAN approaches that use many hundreds or thousands of training images, this method requires only a single colorized reference image for training, avoiding the need of a large dataset. This method intercepts dozens of color cartoons of a specific style according to parts, and then generates training images for each part.

To a certain extent, this method solves the shortcomings of deep learning that requires a large amount of data. It performs well on images of a certain specific style, but there are also situations where the color is dim and depends on a specific image style. In addition, image-to-image translation methods like pix2pix can also translate line art images into drawings, which is also achieved by relying on the adversarial generation network.

1.2.2 User guided and interaction colorization

[Qu Yingge et al. 2006] [6] This paper proposes a novel colorization technique that propagates color over regions exhibiting pattern continuity as well as intensity continuity. The proposed method works effectively on colorizing black-and-white manga which contains intensive amount of strokes, hatching, halftoning and screening.

Such fine details and discontinuities in intensity introduce many difficulties to intensity-based colorization methods. Once the user scribbles on the drawing, a local, statistical based pattern feature obtained with gabor wavelet filters is applied to measure the pattern-continuity.

The boundary is then propagated by the level set method that monitors the pattern-continuity. Regions with open boundaries or multiple disjointed regions with similar patterns can be sensibly segmented by a single scribble. With the segmented regions, various colorization techniques can be applied to replace colors, colorize with stroke preservation, or even convert pattern to shading. Several results(Figure1.3) are shown to demonstrate the effectiveness and convenience of the proposed method.

[Chie Furusawa et al. 2017] [7] developed comicolorization, a semi-automatic colorization system for manga image. Given a monochrome manga and reference images as inputs, the system generates a plausible color version of manga. This is the first work to address the colorization of an entire manga title. This method is to semi-automatically color the entire page (rather than a single panel), and use the same color for the same characters in the multiplexer. In order to color the target character according to the color of the reference image, they extract a color feature from the reference image and provide it to the colorization network to help coloring. This method uses adversarial loss to encourage the effect of color features.

[PaintsChainer. 2016] [8] PaintsChainer developed by japan preferred networks appeared in 2016. By uploading a sketch, you can complete the coloring of the sketch in a short time. It becomes a powerful and popular tool for painters. Users only need to input a sketch to get a colorized painting. They can add pointed color hints for better results. The visual effects are stunning. Although it has made good progress in the coloring of comics, there are problems with color overflow in its current series of versions.

[Zhang Lvmin et al. 2017] [9] proposes residual U-net and auxiliary classifier Generative Adversarial Networks (AC-GAN) as a solution and the network can directly generate a combination of sketch and style image. This model is fully feed-forward to achieve high-speed composition of painting. In addition, they find that the performance of U-net and cGAN is relatively degraded in the case of unbalanced input and output information. They recommend using the remaining U-net with two guide decoders. In

addition, they also compared a variety of GAN, found that conditional GANs are not suitable for this task, and finally adopted the auxiliary classifier generative adversarial networks. The main contributions in this paper are as follows:

1) A feed-forward network to apply the style of a painting to a sketch.
2) An enhanced residual U-net capable of handling paired images with unbalanced information quantity.
3) An effective way to train residual U-net with two additional loss.
4) A discriminator modified from AC-GAN suitable to deal with paintings of different style.

1.3 Research content and innovation

In view of the fact that most of the past automatic coloring technology frameworks are based on grayscale images, there are few studies on coloring of sketches, and the image colorization based on grayscale images cannot be effectively migrated to sketches. Therefore, combined with the current research status and technical development level, this article conducts research on how to quickly color sketches and maintain a good visual effect. This article concludes that deep learning technology is applied to the task of coloring sketches. Using the idea of divide and conquer, two models for coloring sketches based on GAN are built. Given the sketch and the constraints, the global boundary semantic features are extracted by deep convolution, and then the regions are segmented according to the semantic features, and the corresponding colors are obtained from the color distribution for smearing. In the end, a color cartoon image with reasonable and good visual effect is finally obtained.

The main task of the first model is to make a rich coloration of the input line art image, without paying attention to the details, just need to cover it with colors. In the second model, the color image obtained by the first model is corrected once again. The model principle is based on conditional Generative Adversarial Networks [10]. By adding restrictions, the color image obtained by the first model is refined to the final color cartoon image. In order to make our task achieve better results, we have optimized the loss function, network structure, and dataset.

Aiming at the problem of coloring of sketches, we focus on the principle of GAN, adding L1 regular terms to the loss function to make the network converge better and have stronger generalization capabilities.

In terms of network structure, combined with previous experience, the discriminator mode is optimized to distinguish true and false according to the local area of the real image and the local area of the generated image, and then average to obtain the final judgment

result. This is different from the previous overall discrimination, which enhances the discriminator model capability. At the same time, the forward propagation algorithm is used to encourage the model to produce more vivid images.

In addition, in view of the shortcomings of the deconvolution layer and the problem of the checkerboard effect, scaling and convolution are used instead of deconvolution to optimize, reducing the coloring defects caused by the checkerboard effect.

In the processing of datasets, there are almost no data sets where the existing sketch matches the colored image, and most of them are pure color comic images. In response to the different needs of the two models, deep learning technology and digital media processing technology were used to produce line art images and simulated color images.

2. Related theory of image colorization

2.1 Digital image

Digital image is the expression of continuous light signal in the spatial domain after sampling by the sensor. And digital image is an image composed of picture elements, also known as pixels. Try to enlarge a picture in the "drawing picture" software will have a more intuitive feeling, the following digital image(Figure2.1) is the effect of the enlarged animation picture.

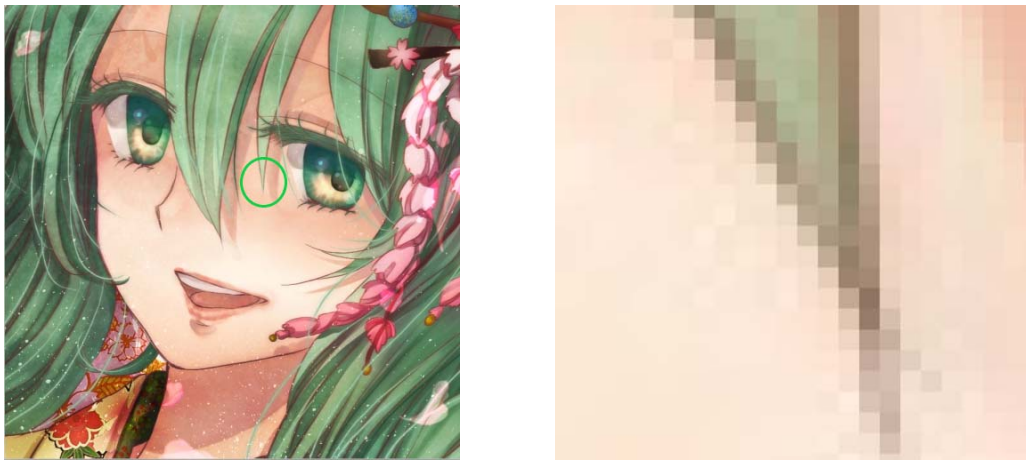


Figure 2.1 Digital image

It can be seen from digital image (Figure2.1) that the image is composed of many small grids, each of which has only one color. This is the smallest unit that makes up the image, and is called a pixel. Different pixel values represent different colors. The pixel value (Figure2.2) range is generally between 0 and 255 (inclusive), which is 256 integers.

Therefore, the entire value range of unsigned char type can be used, so pixel values are generally expressed by unsigned char type. But 0-255 can not be mapped to the color as shown above, but only corresponds to the grayscale between black and white. To express colors, the concept of color space needs to be derived, as described Ch2.2 Color.

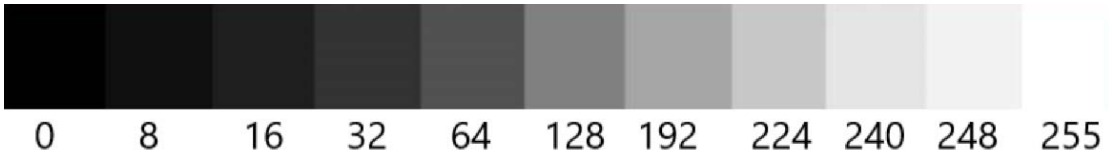


Figure 2.2 Grayscale value

2.2 Color

Color is the characteristic of visual perception described through color categories, with names such as red, orange, yellow, green, blue, or purple. This perception of color

derives from the stimulation of photoreceptor cells (in particular cone cells in the human eye and other vertebrate eyes) by electromagnetic radiation (in the visible spectrum in the case of humans). Color categories and physical specifications of color are associated with objects through the wavelength of the light that is reflected from them. But in computers, we need to digitally abstract colors. This leads to a new concept, color space.

A color space is a specific organization of colors. In combination with physical device profiling, it allows for reproducible representations of color, in both analog and digital representations. A color space may be arbitrary, with particular colors assigned to a set of physical color swatches and corresponding assigned color names or numbers, or structured mathematically.

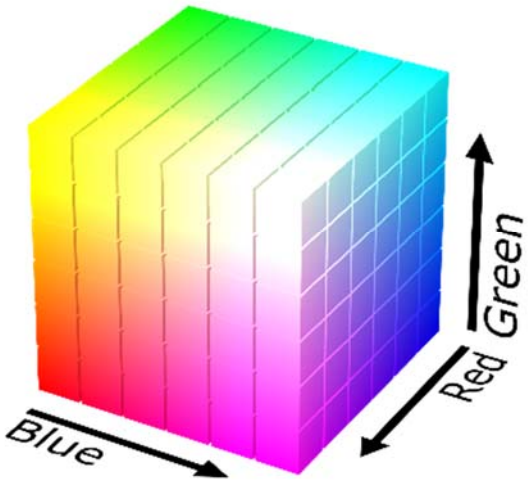


Figure 2.3 RGB space

For example, in the RGB space (Figure2.3), to represent color pixels, first recall the three primary colors of junior high school physics, red, green and blue (RGB), and the saturated red, green, and blue colors are white when they are superimposed. If one of the colors is less "saturated" then it can represent the other colors. Adjusting the ratio of the three colors can represent the 24-bit color we often see. The color space of the gray value can be represented by a straight line geometrically, while the RGB color space corresponds to a cube geometrically.

Therefore, to represent color values, we need three dimensions, that is, three image channels. Each pixel value is represented by three numbers, such as (255,255,255) for white, (255,0,0) for red, (255,255,0) for yellow.

2.3 Image segmentation

Image segmentation is a classic problem in computer vision research. Image segmentation is the first step in image analysis, the foundation of computer vision, and an important part of image understanding. Image segmentation refers to dividing the image into several disjoint areas based on grayscale, color, spatial texture, geometric shape and other features, so that these features show consistency or similarity in the same area, but between different areas make a clear difference. Simply put, it is to separate the target from the background in an image. This is also a fundamental part of our sketch colorization models.

The methods of image segmentation can be roughly divided into three categories: traditional segmentation algorithms, image segmentation algorithms combined with specific tools, and segmentation algorithms based on deep learning technology.

The common traditional algorithms are threshold based segmentation method and edge detection based segmentation method.

Image segmentation algorithms combined with specific tools mainly include image segmentation methods based on wavelet analysis and wavelet transform, image segmentation based on genetic algorithm, and segmentation method based on active contour model.

Common segmentation methods based on deep learning are feature encoder based, regional proposal based, upsampling, deconvolution based segmentation methods (such as FCN [11]), feature-based segmentation methods, use CRF[12] and MRF [13]method.

2.4 Image colorization

This part has basically been presented in the introduction, here only to supplement some methods not mentioned before.

Image colorization is the process of coloring an image, and has high research and application value. Existing coloring methods are mainly divided into two categories, one is coloring methods based on traditional image processing technology, and the other is coloring methods based on deep learning.

Due to the results achieved color depth learning-based method is often better than traditional methods, so we mainly discuss coloring methods based on deep learning. Image colorization based on deep learning technology is mainly divided into two coloring methods, one is automatic image colorization [3,4,5], and the other is user guided interaction colorization [6,7,8,9].

In the user guided interaction colorization (Figure2.4), it also includes colorization based on color strokes[6,8,9,14,15], colorization based on reference color image[7,16], colorization based on color palette[7,17], colorization based on language(text)[18,19].

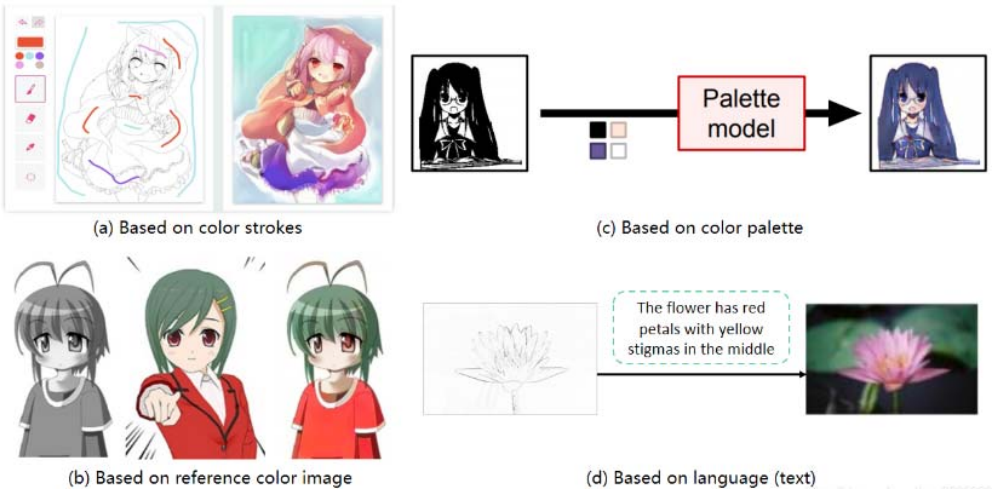


Figure 2.4 The methods of user guided interaction colirization[7,8,18]

3. Related theory of GAN

3.1 The introduction of GAN

A generative adversarial network (GAN) is a class of machine learning frameworks invented by Ian Goodfellow in 2014. GAN has achieved great success in image generation, which undoubtedly depends on GAN's continuous improvement of modeling capabilities under the game theory, and ultimately the realization of realistic image generation. The basic idea of GAN is derived from the two-player zero-sum game of game theory. It consists of a generator and a discriminator, and is trained by adversarial learning. The purpose is to estimate the potential distribution of data samples and generate new data samples. GAN is being widely used in style transfer, color filling, image-to-image translation, image restoration and other fields.

3.1.1 Fundamental principle of GAN

As its name “Generate Adversarial Network”, GAN is actually composed of two neural networks that are relatively independent in structure, namely generator and discriminator. The purpose of the generator is to try to learn and capture the potential distribution of real data samples, and generate new data samples; the discriminator is a binary classifier, the purpose is to try to correctly determine whether the input data comes from real data or from the generator. In order to win the game, the two game participants need to constantly optimize and improve their own generating ability and discriminating ability. This learning and optimization process is a minimax game problem. The purpose is to find a nash equilibrium between the two.

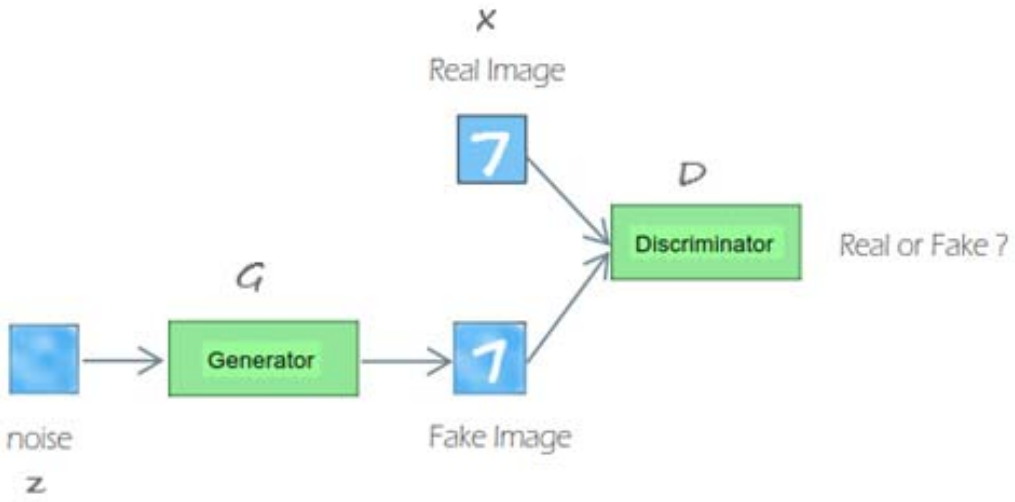


Figure 3.1 The simplified structure of GAN

The simplified structure is shown as the picture(Figure3.1). G (generator) is a network that generates images. It receives a random noise and generates a noise picture $G(z)$ from the noise, and then inputs $G(z)$ as a parameter to the D (discriminator). D is a discriminating network that discriminates whether a picture is real. Its input parameter is x, x represents a picture, and the output $D(x)$ represents the probability that x is a real picture. After $G(z)$ is passed into D, D gives the probability that the picture is true. If $D(G(z))$ is 1, it means that 100% is the real picture, and if the output is 0, it means that it is impossible to be the real picture. In the training process, the purpose of generating the network G is to generate as many real pictures as possible to deceive the network D, and the purpose of D is to distinguish the generated pictures from the real pictures as much as possible. The most ideal state: the final picture $G(z)$ generated by the generator G can make the discriminator D unable to judge the true or false, that is, $D(G(z))$ = 0.5.

3.1.2 Loss of the original GAN

So how to make our network model reach the ideal state? First of all, we first understand the training process of GAN.

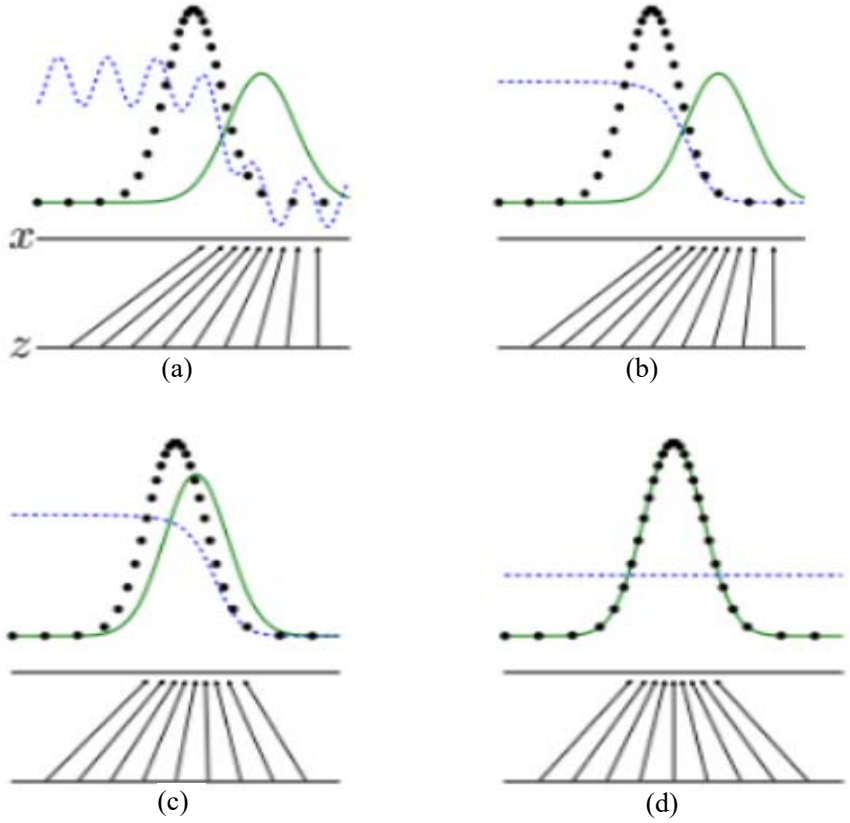


Figure 3.2 The training change of Generative Adversarial Networks. (a) Before training (b) Start training (c) Continue training (d) Complete training [1]

As shown in the GAN training change chart(Figure 3.2) above that the black dotted line is the gaussian distribution of real data, the green line is the forged distribution learned by the generation network, and the blue line is the probability that the network is judged to be a real picture. The horizontal line marked x represents the sampling space subject to the gaussian distribution x, and the horizontal line marked z represents the sampling space subject to the uniform distribution z. It can be seen that the mapping relationship from the space of z to the space of x is learned. G(generator) and D(discriminator) can be said to be two completely independent models. Generally, D is connected in series after G during training. Use a separate alternating iteration method to train both.

The training process is roughly as follows:

1) Freeze the hyperparameters of G and train D to make it possible to distinguish whether the samples come from real or fake distributions.
2) Freeze the hyperparameters of D and train G so that it can generate samples that approximate the true distribution.
3) Repeat the above process until the network converges.

It can be seen from the training change graph that there is a large gap between the distribution of real data and fake data at the beginning of training. With the advancement of the training process, the distribution fit between the generated data and the real data is getting better and better, and the ability of D to distinguish the samples has also improved. In the end, the distribution of generated data is consistent with the distribution of real data. D cannot distinguish the authenticity of the samples and the model is trained. The mathematical language is used to describe this process.

$$\min_G \max_D \left\{ f(D,G) = E_{x\sim Pdata(x)}[\log D(x)] + E_{z\sim Pz(z)}\left[\log\left(1 - D\left(G(z)\right)\right)\right]\right\} \quad (3.1)$$

$E_{x\sim Pdata(x)}$ refers to taking the real sample in the training data x, and $E_{z\sim Pz(z)}$ refers to the sample extracted from the noise distribution. x represents a real image, and z represents a noise. The $G(z)$ represents an image that is generated by the G network. $D(x)$ represents the probability that the D network judges whether the image is real. $D\left(G(z)\right)$ is the probability that D determines whether the image generated by G is true.

It can be seen from the above formula that the loss function of GAN is a maximum and minimum optimization problem, and the optimization process is divided into two steps. For G, it is necessary to make the value of this function as small as possible, which means that the larger $D\left(G(z)\right)$ is. That is, the more realistic the image generated by our

generation network. For D, it wants to use this function as large as possible, which means that $D\big(G(z)\big)$ is as small as possible. That is, D can judge all the generated images.

3.1.3 Insufficient of original GAN

As an original work, GAN has many shortcomings. These problems include: the generator gradient disappearance problem caused by the high degree of discriminator training, the gradient instability problem under the optimal discriminator, and collapse mode.

When the discriminator training degree is high, the loss of the generator is close to the JS divergence between the generated image distribution and the real image distribution. When the supporting set of the generated image distribution and the real image distribution is a low-dimensional manifold in a high-dimensional space, the probability that the overlap measure of the two is 0 is 1. At this time, the JS divergence will be constant, and the Loss of the generator will not change, that is, the generator will not be optimized, and the gradient disappears.

Minimizing the generator loss function will be equivalent to minimizing an unreasonable distance measurement, leading to two problems, one is unstable gradient, and the other is collapse mode. Let's first understand what is meant by "minimizing the generator loss function, which is equivalent to minimizing an unreasonable distance measurement", let us interpret the following formula(3.2).

$$\begin{aligned} E_{x\sim Pg}[-logD^*(x)] &= KL\big(P_g\big||P_r\big) - E_{x\sim Pg}\log[1-D^*(x)]\big] \\ &= KL(P_g||P_r) - 2JS(P_r||P_g) + 2log2 + E_{x\sim Pr}[logD^*(x)] \end{aligned} \tag{3.2}$$

The left term of the above formula is the loss of the generator, and the last two terms of the right term have nothing to do with the generator, so if you want to minimize the generator loss, it is equivalent to minimizing: $KL(P_g||P_r) - 2JS(P_r||P_g)$. Intuitively, it is to minimize KL divergence and maximize JS divergence. Since both are similarity measures, the impact is that two divergences require one to draw closer to the true distribution and one to push farther away. This is intuitively ridiculous, and numerically results in gradient instability.

So, for collapse mode, let's first read the following formula about the different penalty mechanisms for KL divergence for two types of errors generated by generator generated images.

$$P_g(x) \to 0 \;\; P_g(x) \to 1, P_g(x) \log \frac{P_g(x)}{P_r(x)} \to 0, the\ contribution\ of\ \ KL(P_g||P_r) \to 0 \quad (3.3)$$

$$P_g(x) \to 1 \;\; P_g(x) \to 0, P_g(x) \log \frac{P_g(x)}{P_r(x)} \to +\infty, the\ contribution\ of\ \ KL(P_g||P_r) \to +\infty \quad (3.4)$$

The first formula(3.3) represents a lack of diversity, but because accuracy is guaranteed, the KL divergence will not pay too much attention to this problem, that is, as long as the accuracy is guaranteed, it does not matter if the repeated pictures are generated. The second formula represents the lack of accuracy. As can be seen from the above formula(3.4), the KL divergence attaches great importance to this problem, that is, the accuracy must be guaranteed. Therefore, the generator would rather generate some repeated samples rather than generate diverse samples. This is the collapse mode.

3.2 Variants of GAN

GAN is a method of unsupervised learning. The generator and discriminator constantly improve the performance in the game. After reaching a nash equilibrium, the generator can produce images that are infinitely close to the real thing. However, the nash equilibrium only exists in the theory, and there are still some problems in the actual GAN, such as unstable training and collapse mode. Even so, the use of the original GAN has had some good success. On this basis, there are constantly new versions of the derivative.

3.2.1 WGAN

WGAN [20] is wasserstein GAN, mainly from the loss function to optimize the original GAN. The core idea is to use wasserstein distance instead of JS divergence and KL divergence. WGAN uses earth-mover instead of JS divergence to measure the distance between the real sample and the generated sample distribution. Compared with KL divergence and JS divergence, the earth-mover distance is smooth. Even if the two distributions do not overlap, they can still reflect their distance, thus providing a meaningful gradient.

WGAN pioneered the following:

1) Completely solve the problem of GAN training instability, no longer need to carefully balance the training level of the generator and discriminator.
2) Basically solved the collapse mode phenomenon, ensuring the diversity of generated samples.
3) During the training process, there is a value such as cross entropy and accuracy to indicate the progress of the training. The smaller the value, the better the GAN training and the higher the image quality generated by the generator.
4) The most simple multi-layer fully connected network can be achieved without elaborate network architecture.

Compared with the original GAN, the improved algorithm implementation process has only changed four points:

1) The last layer of the discriminator removes the sigmoid.
2) The loss of the generator and the discriminator does not take the log.
3) The updated weight is forcibly truncated to a certain range, such as [-0.01, 0.01], to satisfy the lipschitz continuity condition mentioned.
4) WGAN also recommends the using of SGD, RMSProp and other optimizers, not based on the using of momentum optimization algorithms, such as Adam.

3.2.2 DCGAN

The improvement of DCGAN [21] over the original GAN mainly lies in the network structure. DCGAN integrates CNN with supervised learning and GAN with unsupervised learning. DCGAN greatly improves the stability of GAN training and the quality of the final results.

The main contributions of DCGAN are:

1) A series of restrictions are set for the network topology structure of CNN so that it can be trained stably.
2) The use of DCGAN to learn useful features from a large amount of unlabeled data (image and voice) is equivalent to the use of unlabeled data to initialize parameters of DCGAN generator and discriminator for supervised scenes such as image classification, so as to obtain good results to verify the expression ability of the generated image features.
3) Qualitative analysis of the filter learned by GAN was conducted to try to understand and visualize how GAN works.
4) It shows the vector computing characteristics of the generated feature representation.

The main reasons why DCGAN can improve the stability of GAN training are:

1) Remove the pooling layer in the generator network and discriminator network.
2) Using Batch Normalization in both generator and discriminator networks.
3) Remove the fully connected hidden layer.
4) Using the LeakyReLU activation function in the discriminator instead of ReLU to prevent the gradient sparse. The generator still uses ReLU, but the output layer uses Tanh.
5) LeakyReLU was used at each layer of the discriminator network.

3.2.3 CGAN

Compared with other generative models, this competitive method of GAN no longer requires a hypothetical data distribution, that is, formulate p (x) is unnecessary. Instead, it uses a distribution to directly original sampling, so as to truly achieve the theoretical approach to completely approximate the real data, which is also the biggest advantage of GAN. However, the disadvantage of this method without pre-modeling is that it is too free. For larger images and more pixels, the simple GAN based approach is less controllable. In order to solve the problem that GAN is too free, a very natural idea is to put some constraints on GAN, so there is conditional Generative Adversarial Networks (CGAN)(Figure3.3).

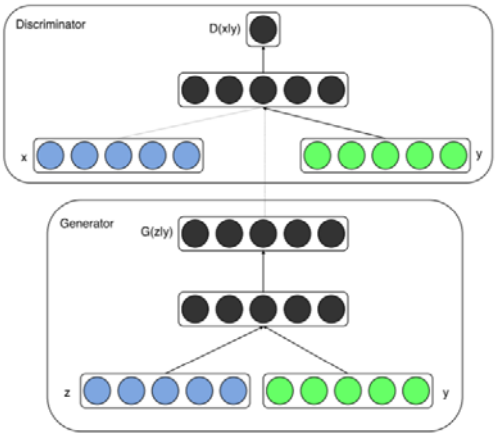


Figure 3.3 Conditional adversarial net [10]

The objective function of CGAN would be as:

$$\min_G \max_D V(D,G) = E_{x\sim Pdata(x)}[logD(x|y)] + E_{z\sim Pz(z)}\left[\log\left(1 - D\left(G(z|y)\right)\right)\right] \quad (3.5)$$

Generative adversarial nets can be extended to a conditional model if both the generator and discriminator are conditioned on some extra information y . y could be any kind of auxiliary information, such as class labels or data from other modalities. We can perform the conditioning by feeding y into the both the discriminator and generator as additional input layer. In the generator the prior input noise $Pz(z)$, and y are combined in joint hidden representation, and the adversarial training framework allows for considerable flexibility in how this hidden representation is composed. In the discriminator x and y are presented as inputs and to a discriminative function.

4. Sketch image colorization based on GAN

Sketch colorization is a research field with significant market demand. Different photo colorization which strongly relies on texture information, sketch colorization is more challenging as sketches may not have texture. Even worse, color has to be generated from the abstract sketch lines [22]. In order to achieve a good coloring effect, we investigated the drawing process of comic artists.

When professional painters color line drawings, they like to color the overall outline first to form a preliminary color sketch. Then fine-tune the local details and dissatisfaction in the color sketch to gradually form the final painting. In other words, comics are not created in one step, they are created by the accumulation of multiple steps.

But most of our current sketch colorization models do not take multiple steps to color a sketch. At present, most of the coloring network is: input a sketch, after the network processing, directly output color comics. The picture output in this way is usually not very good in coloring, and has more or less defects and deficiencies.

In order to improve the coloring quality of the deep network, this paper draws inspiration from the artist's coloring process and proposes a coloring model divided into two stages, as shown in the figure4.1.

The first stage is called drafting stage, this stage is similar to the principle of most of the current automatic coloring network, by inputting a line image, and processing the network to get the final color, this fully automatic The effect of coloring is generally not very good. However, the purpose of this stage is to increase the richness of color matching. The resulting sketch may contain some coloring errors and blurry textures, but it is full of rich, vivid colors.

The second stage is called refinement stage, which focuses on correcting details and clarifying the blurry texture to get the final painting. The two models are trained separately, and then connected together during the testing phase to generate the final output. And Figure 4.1 shows the framework.

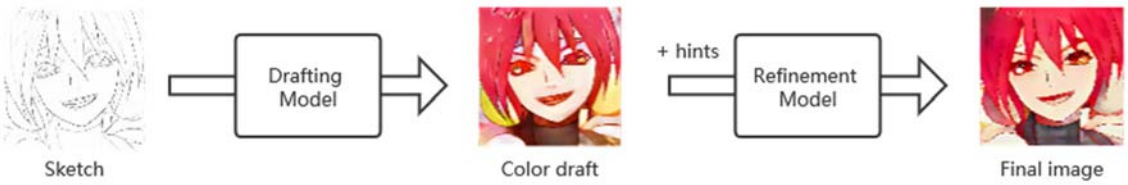


Figure 4.1 Overview of our framework

Comparing to existing approaches, this two-stage design effectively divides the complex colorization task into two simpler and goal-clearer subtasks. This eases the learning and raises the quality of colorization.

4.1 Divide and conquer algorithm

The two-stage coloring model proposed in this paper adopts the divide and conquer algorithm to decompose the complex coloring problem into two simpler sub-problems, and each sub-problem has a clearer task aims. In computer science, divide and conquer is an algorithm design paradigm based on multi-branched recursion. A divide and conquer algorithm works by recursively breaking down a problem into two or more sub-problems of the same or related type, until these become simple enough to be solved directly. The solutions to the sub-problems are then combined to give a solution to the original problem.

Algorithm : Divide-and-Conquer.

```
if |P| ≤ N_0 then
    return ADHOC(P)
end
else
    Decompase P → P_1, P_2...P_k;
    for i = 1:k do
        Divide-and-Conquer(P_i)→ y_i;
        MERGR(y_1, y_2...y_k)→ T;
        return T;
    end
end
```

Where $|P|$ represents the scale of the problem P. And n_0 is a threshold, indicating that when the scale of the problem P does not exceed n_0, the problem is easily solved directly, and no further decomposition is needed. $ADHOC(P)$ is the basic sub-algorithm in this divide-and-conquer method, which is used to directly solve the small-scale problem P. Therefore, when the scale of P does not exceed n_0, it is directly solved by the algorithm $ADHOC(P)$. $MEGRE(y_1, y_2 \dots y_k)$ is the merge sub-algorithm in this divide and conquer method. It is used to merge the corresponding solutions of the sub-problems $P_1, P_2 \dots P_k$ of P into the solution of P.

The problem-solving idea of this article is derived from this, to simplify a complex problem. The advantages of this design scheme are: (1) The network can learn better coloring ability. (2) The model has better robustness.

4.2 Drafting model

4.2.1 Introduction of the drafting model

Regarding the coloring model at this stage, we plan to use a fully automatic coloring method to complete the coloring. Since we will have further modification work later, the accuracy of this part does not need to be too high, only the rough coloring task needs to be completed quickly and easily. Automate the coloring process and bring progress to the coloring of comics in terms of time and efficiency.

We have previously investigated relying on the continuity of gray areas to find areas with the same gray level to assist in the segmentation of the image and then complete the colorization.

However, the sketch image only has image outline information, and unlike grayscale images, continuous grayscale areas with different levels can be used for image segmentation. Therefore, the method of colorizing the grayscale image cannot effectively and normally process the coloring of the hand drawing.

Therefore, we set our sights on the field of deep learning. Finally, the GAN that has achieved good results in the field of image generation is selected.

The original GAN does not meet our needs very well, because our goal is not to generate random images aimlessly. Instead, a color image is generated on a sketch image. This color image should continue to retain all the edge information of the original sketch. Therefore, we need to add a constraint to the GAN network, which is the conditional Generative Adversarial Networks (cGAN) selected in our model.

Throughout the process of coloring the sketch, the sketch image is used as the input of the generator model, and the generator model extracts the key features of the image layer by layer for the image. Then combine random noise to select the highest confidence color from the data color distribution according to the extracted key features to generate a color comic image.

Then, the generated color cartoon images and real color cartoon images are input into the discriminator network. The discriminator mainly performs a two-classification task. By extracting features from these true and false images, a feature matrix is obtained, and then the judgment probability is obtained using the Sigmoid function.

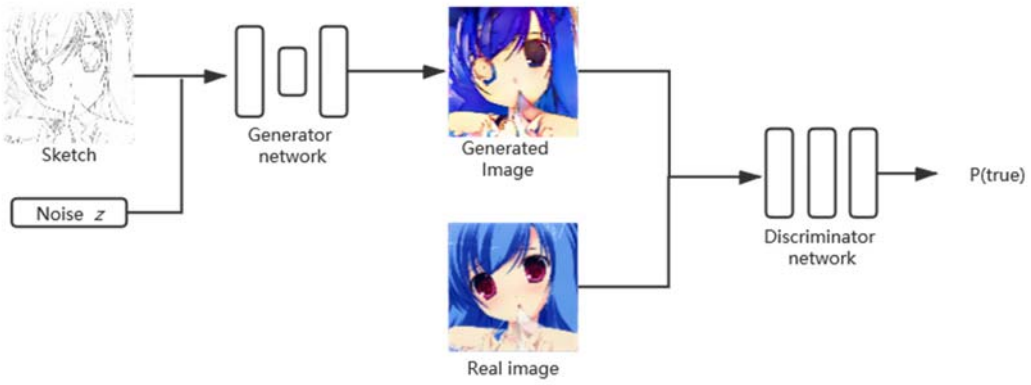


Figure 4.2 Training of the drafting model

As shown in the figure above (Figure4.2), our model is based on cGAN. Here we set $\{x, z\} \rightarrow y$. The generator G generates a color image based on the input, and the discriminator D completes its own training based on the image generated by the generator and the real image. The goal of the generator G is to produce an image that can deceive the discriminator D and the goal of the discriminator D is to distinguish the generated image from the real image. The two networks continually optimize training in a confrontational manner, constructing a mapping relationship between hand-drawn images and color images, and finally making the network have the function of automatic coloring.

The objective function for cGAN can be expressed as the formula(4.1).

$$L_{cGAN}(G, D) = E_{x,y\sim Pdata(x,y)}[logD(x, y) + E_{x\sim Pdata(x),z\sim Pz(z)}[\log\big(1 - D\big(x, G(x, z)\big)\big)] \quad (4.1)$$

$D(x, y)$ and $D\big(x, G(x, z)\big)$ are to judge the image true or false. Ideally, our discriminator should be $D(x, y)$ close to one and $D\big(x, G(x, z)\big)$ close to zero. That means that discriminator wants our objective function to be as big as possible. And $G(x, z)$means the image which generated by generator. Generator wants $G(x, z)$ to be as close to one as possible. That means that generator wants our objective function to be as small as possible.

The generator tries to minimize the objective function, and the discriminator tries to maximize the objective function, that is:

$$G^* = arg \min_G \max_D L_{cGAN}(G, D) \quad (4.2)$$

If we directly use the above objective function to train our model, although we will get some fancy results, we will find that some image areas are wrong. The problem is that GAN always likes to create images by himself, as long as the output image can deceive the discriminator. So in order to reduce the occurrence of this situation, we added an L1 loss at the end of the generator. The output generated in this way can be kept clear, and there will be no areas in the picture that do not conform to the deeds.

$$L_{L_1}(G) = E_{x,y\sim Pdata(x,y),z\sim Pz(z)}[||y - G(x,z)||_1] \qquad (4.3)$$

y is the real image, and $G(x,z)$ is the generated image. So L1 is just calculating the minimum absolute deviation of the two distributions.

Previous conditional generative adversarial networks have found that it is more advantageous to combine the GAN objective function with traditional loss functions such as L2 distance. The task of the discriminator model remains the same, but the task of the generator model not only deceives the discriminator model, but also needs to be close to the real output on the basis of L2 distance. Use L1 distance instead of L2 distance, because L1 distance produces clearer results, and the final objective function is as follows:

$$G^* = arg \min_G \max_D L_{cGAN}(G,D) + \lambda L_{L_1}(G) \qquad (4.4)$$

4.2.2 Network structure of the drafting model

According to the coloring characteristics of the line draft, we can find that although our input and output are different in color performance, they have the same outline structure. The structure of the input and the structure of the output are almost the same, so the model architecture of our generator also revolves around this feature. In the problem of coloring images, most of the schemes use a self-coding network, like figure4.3.

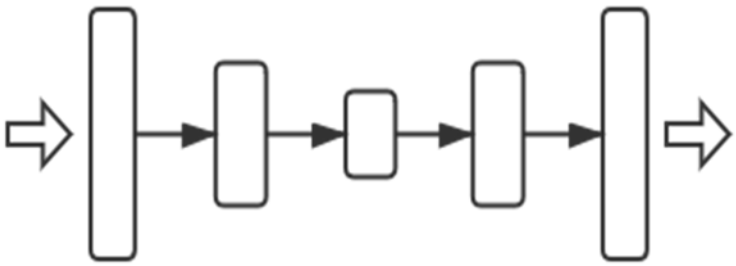


Figure 4.3 The self-encoding network structure

The encoder part extracts high-level features through continuous downsampling. When we finally reach the bottleneck layer, the features we extract often have a lot of structured information, which provides semantic information for segmentation and color selection for our subsequent network. After the bottleneck layer is the decoder. The decoder continuously restores the original image size through continuous upsampling. During the upsampling process, semantic segmentation is performed to obtain some colored regions, and then the most suitable colors are selected from the color distribution of the training data to fill according to these semantic features.

For problems such as image coloring, a large amount of low-level information is shared between input and output. For example, the input and output share the position of the edge. Therefore, we may wish to let this information directly reach the deep network through the shallow network, so that each layer of feature maps can be effectively used in subsequent calculations, which is skip connection(Figure 4.4).

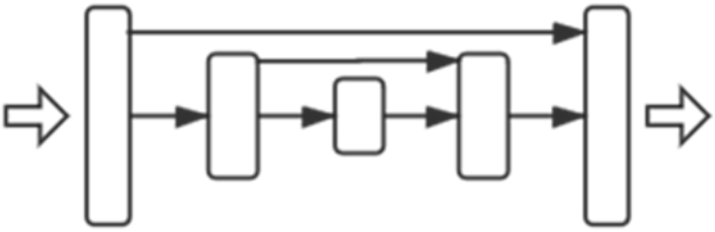


Figure 4.4 The structure of self-encoding network with skip connection

According to the ideas above, I used a generator network based on U-net network structure in this article (Figure4.5).

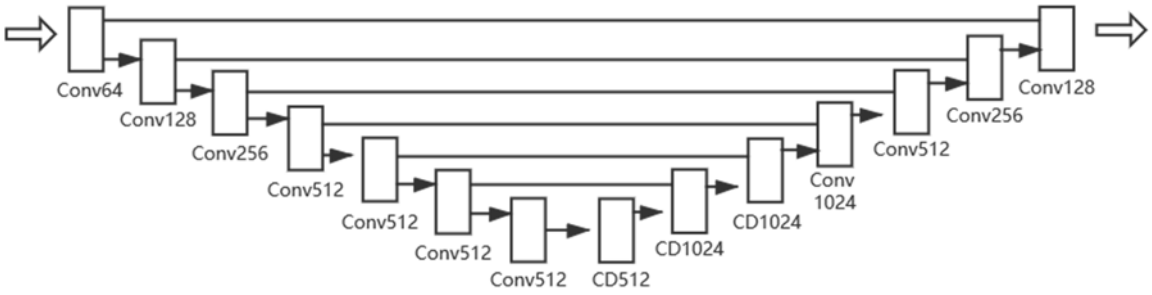


Figure 4.5 Generator network structure of the drafting model

The generator model contains two parts: encoder and decoder. Conv64, conv128, conv256, conv512, conv512, conv512 and conv512 are the encoder part. CD512, CD1024, CD1024, conv1024, conv512, conv256 and conv128 are the decoder part.

Among them, each convolution layer of the encoder part uses step convolution instead of downsampling, which avoids the use of the pooling layer and avoids the problem of sparse gradients. Similarly, in the downsampling process, we use LeakyReLU to replace the native ReLU function. Because compared to ReLU, LeakyReLU allows a smaller negative activation value, thus relaxing the sparsity limit. All activation functions in the encoder are LeakyReLU with a slope of 0.2. The decoder uses the ReLU activation function.

In order to avoid the checkerboard effect in the decoder, the upsampling and convolution method is used instead of the deconvolution layer. The convolutional layer at the beginning of the CD is with dropout. In addition to the regularization function, dropout here also functions to add random noise. Dropout in the decoder part is used as random noise input, which can provide randomness for color filling when upsampling, and our colors are more abundant.

To give the generator a means to circumvent the bottle-neck for information like this, we add skip connections, following the general shape of U-Net. Specifically, we add skip connections between each layer i and layer n − i, where n is the total number of layers. Each skip connection simply concatenates all channels at layer i with those at layer n − i.

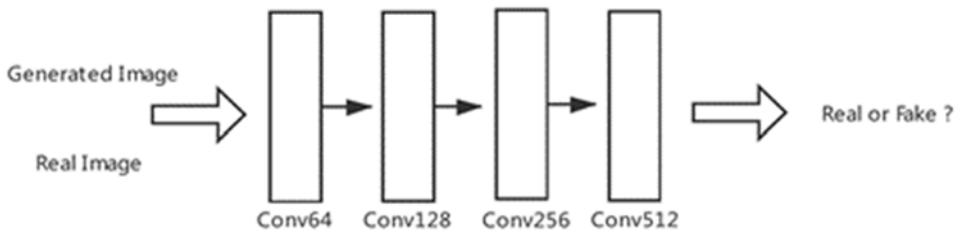


Figure 4.6 Discriminator network structure of the drafting model

As mentioned above, using the L1 distance can make the image clearer. At the same time, it can also capture the low-frequency information of the image in many cases. In this case, when designing the GAN, we only need to consider how to model the high-frequency information.

And when modeling high-frequency information, it is enough to focus only on the local image, so it is suitable to use PatchGAN. The discriminator structure (Figure4.6) uses PatchGAN, that is, the output sample is divided into several patches, then each patch is scored, and finally the average score of all patches is used as the final score. And the small patch size can still obtain high-quality structure, while PatchGAN has fewer parameters and faster training speed, and can be used on arbitrarily large images.

4.3 Refinement model

The coloring quality of the sketches generated by the drafting model is low, and often contains more coloring errors and defects. For example, color mistakes, color bleeding, and color distortion. In order to improve the coloring quality, the algorithm must be able to detect and correct areas containing defects. In order to achieve this goal, this paper proposes the refinement model. Unlike the drafting model, this model is not an automatic coloring model, but a sketch colorization model based on user interaction (Figure 4.7).

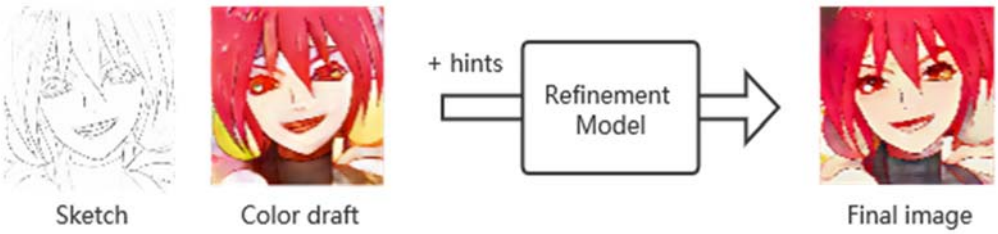


Figure 4.7 Refinement model

4.3.1 Simulating user interactions

To train deep coloring networks based on color hints, you must have color hint data. The problem is that not only is it expensive to collect such data, it is actually not feasible! Because collecting color hint data is a chicken and egg problem.

Specifically, the color hint data is generated during the user's interaction with the coloring system. After the user evaluates the preliminary coloring result output by the system, the user can add the next color hint at the unsatisfactory location.

However, since there is no existing color hint data. Therefore, there is no trained coloring network, so the initial coloring results cannot be presented to the user. That is to say: there is a cyclic dependency relationship between the color prompt of the user and the coloring effect presented by the system.

In order to break this cyclic dependency, we use simulated user color hint data to train our network(Figure4.8). The specific method is based on the method proposed in [23].

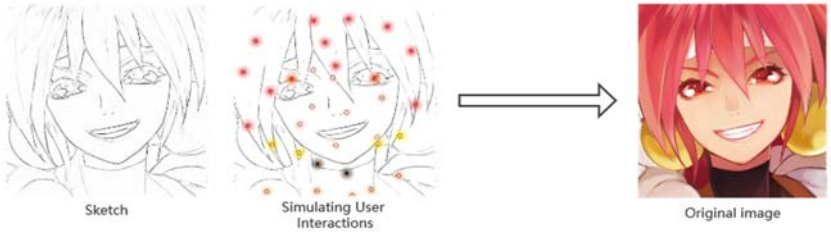


Figure 4.8 Simulate color prompt points

We sample small patches and reveal the average patch color to the network. For each image, the number of points are drawn from a geometric distribution with p = 1/10 .

Each point location is sampled from a 2-D gaussian with $\mu = \frac{1}{2}[H, W]^T, \Sigma = diag(\left[\left(\frac{H}{4}\right)^2, \left(\frac{W}{4}\right)^2\right])$. As we expect users to more often click on points in the center of the image.

When the user selects the color of the cue point, there is often a certain deviation. So, how to choose the color for the color cue points generated by the simulation? We consider that there is often a certain error between the color tips added by the user and the actual colors of the pixels corresponding to the image. Therefore, we determine a smaller rectangular area with the hint point as the center, and take the average color value of the area.

Considering that our data set is composed of color avatars, the composition is relatively simple. Here we have made some improvements compared to the original method, which increases the effective area of the cue points. The size of the area is randomly selected from a uniform distribution of 4 × 4 to 16 × 16.

4.3.2 Network structure of refinement model

The network structure at refinement model is still GAN, but unlike the fully automatic coloring of the drafting model, we use semi-automatic coloring based on user prompts in the refinement model. So it is different from the input in the drafting model. We added a user's color prompt information and simulated color sketch information to the input of generator G. The simulated color sketch information mainly comes from the color features of the sketch extracted by the color feature extractor. Why use simulation to synthesize color sketches instead of the images generated by drafting model will be described in detail in the next section. Through a series of feature information, the generator G generates a color picture, and inputs the generated color picture into the discriminator D. The discriminator D judges the input image and determines whether it is true or false. The two networks constantly optimize training in a confrontational manner, and finally achieve the effect of repairing the color sketch. The network structure is shown in (Figure 4.9).

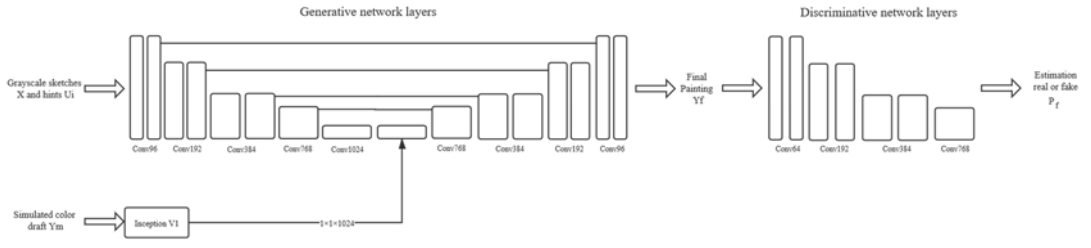


Figure 4.9 Network structure of generator and discriminator in refinement model.

It can be seen that in our generator network structure, we still use the U-net structure. Use U-net to complete the conversion from sketch to color comics. U-net is divided into encoder and decoder.

Because the refinement model and the drafting model are independent, we still need to input sketches in the refinement model to complete our training. But, we preprocessed the line art image, turned it into a grayscale image, and then input it. This is because it helps to improve the coloring quality [9].

In the encoder part of U-net, the semantic segmentation feature and color hints feature are obtained through the input sketches and the simulated color hint points. The color sketches generated by our simulation are not directly input into the U-net network, but the color sketch coloring features are extracted through a color feature extractor. Then, the obtained semantic segmentation feature, color hints feature and color sketch coloring feature are input into the decoder part together. Get the color comics we generated.

The color feature extractor uses the GoogLeNet[24] network, and does not use the VGG network. These two networks have a common feature, that is, go deeper, but there are great differences in specific implementation. The VGG[25] network is more simple and rough. It keeps rolling the base layer on the basis of Alexnet to expand the depth of the neural network, and has achieved good results. It also makes people realize that deepening the network is an effective way to improve the quality of the model. But it also faces problems with too many parameters, slow training, and disappearing gradients. GoogLeNet calculates loss at different layers and proposes inception structure, which not only deepens the network, but also widens it and reduces the number of parameters. We compared the number of model parameters of the two networks, and GoogLeNet does not contain auxiliary classifier.

Figure 4.10 The model parameters of VGG and GoogLeNet

Based on our final coloring goals and computer computing power considerations, the GoogLeNet network model was selected.

Combining the network structure of our model, we arrive at the final objective function:

$$arg \min_{G} \max_{D} E_{x,u_i,y_m,y_f \sim Pdata(x,u_i,y_m,y_f)}[-\lambda \log (D(y_f)) - \lambda \log \left(1 - D\left(G(x,u_i,y_m)\right)\right) + ||y_f - G(x,u_i,y_m)||_1] \quad (4.4)$$

We synthesize the erroneous color drafts y_m by simultaneously applying the three methods. The sketch x, user hints ui and the synthesized color draft y_m are regarded as inputs, and $Y_f$are the outputs. The ground truth painting y_f is regarded as the label. We train our model with mean absolute error (MAE) and the adversarial loss. We use the pretrained imagenet inceptionV1 [26]as the initial weights of the encoder for synthesized color draft y_m. And λ = 0.01.

4.3.3 Color sketch simulation synthesis algorithm

For deep networks to learn to automatically detect and correct coloring defects, sufficient training data is required. One possible solution is to invite professional painters to revise the color sketches generated in the drafting model one by one to form paired images for training {color sketches output by the network, painter's correction map}. But the plan is too time-consuming and laborious. Another solution is to use the color sketches generated in the previous stage and the original color cartoons in the data set as the training set to form paired images for training.

However, the drawback of this scheme is that it may overfit specific sketches and reduce the generalization ability of the model. In addition, to make matters worse, the scheme is equivalent to training two models of the network at the same time, making the two-stage model framework designed in this paper meaningless.

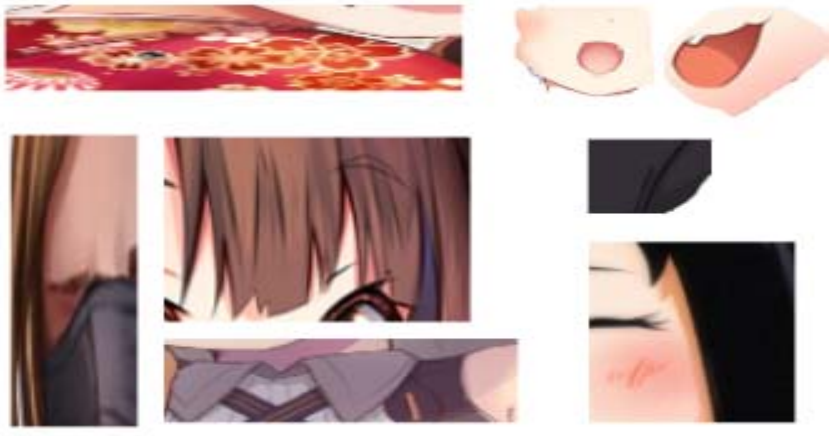
To solve this problem, we propose an algorithm for simulating and synthesizing sketches containing coloring defects. The algorithm synthesizes sketches with color defects based on the original color comics and form paired images for training. This algorithm can help the refinement model learn how to correct different types of coloring errors.

In addition, the algorithm can also enhance the generalization ability of the refinement model, so that the refinement model can not only modify the sketch generated by the drafting network in this article, but also modify other deep coloring networks.

We used three methods to simulate the three major types of sketch colorization problems we can encounter.

(1) Random region proposal and pasting

In the process of coloring, we often have the wrong color applied to a certain area. This is also a common occurrence in the real world. Usually, comic artists choose to use other colors to cover those wrong colors. Then, based on this idea, we can derive inversely, can we use the wrong color to cover the entire color to simulate this kind of error? So we chose to randomly intercept the patch and paste it onto the correct image (Figure4.11).



(a) extracting patch

(b) pasting patches

Figure 4.11 Random region proposal and pasting

The simulated coloring defect is color mistakes. By randomly cutting the 1,000 color comics we prepared, some rectangular areas(patches) were obtained. The size of these areas is between 16 × 16 and 128 × 128. In order to make the simulation effect more realistic, we use image processing technology to randomly distort the obtained area. Finally, paste these area images on our original comics to simulate color mistakes.

(2) Random transform

Also in our coloring process, we sometimes encounter some coloring offset errors. This is because our model incorrectly identified some boundaries during training. This will cause two areas that do not seem to be connected to be filled with the same color. In order to simulate the situation of boundary recognition error, we choose to artificially distort the boundary.

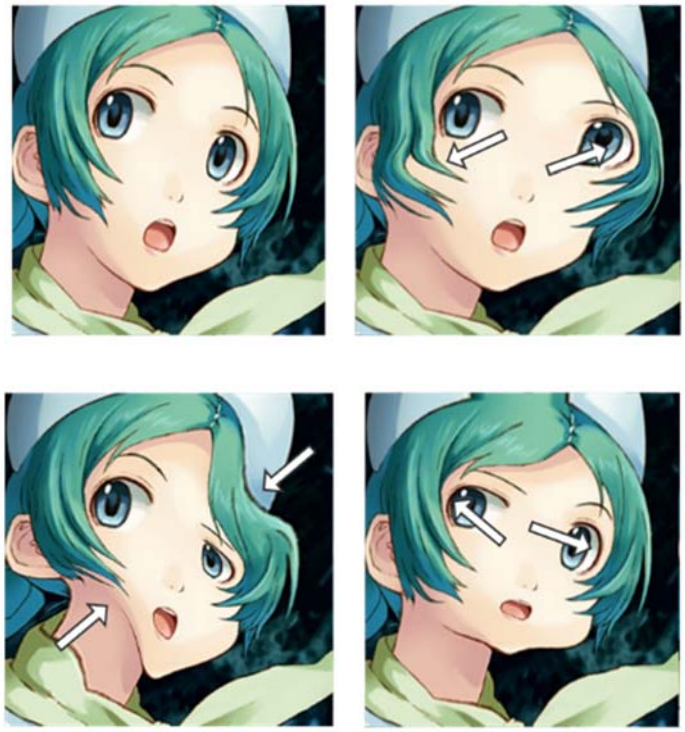


Figure 4.12 Random transform

The simulated coloring defect is color distortion. We use the transform to simulate the blurry and the distorted colors, as demonstrated in (Figure 4.12).

(3) Random color spray

The simulated coloring defect is color bleeding (Figure4.13). Choose a color randomly from the original comic. Use a larger radius spray gun to spray this color randomly onto the comic. The radius is randomly selected from the uniform distribution in the range [16,64]. The number of spray paths is randomly selected from the uniform distribution in the range [1, 3].

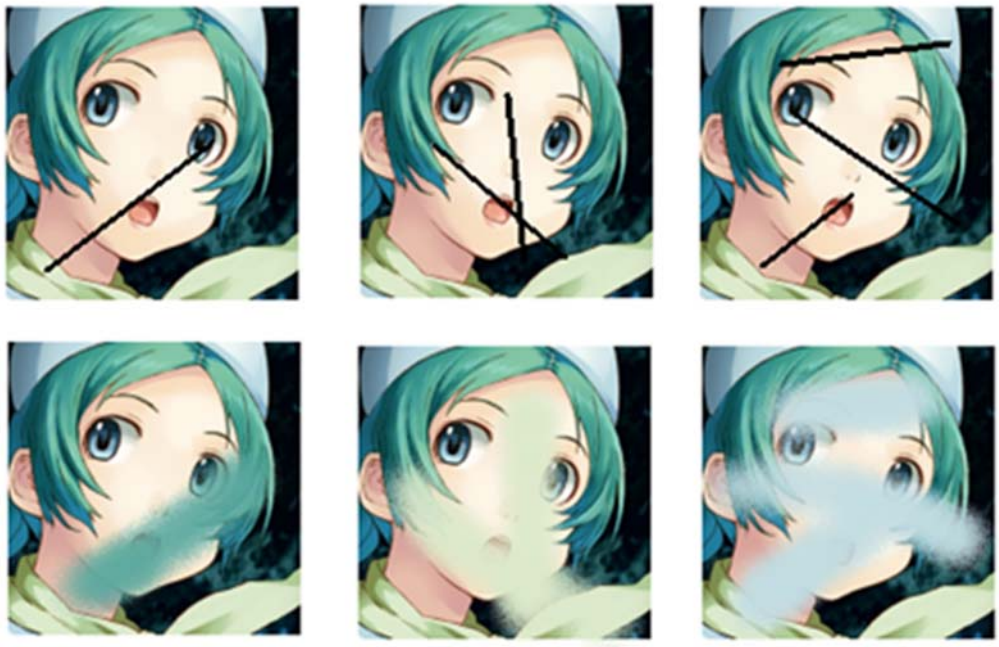


Figure 4.13 Random color spray

4.4 Data set

Most of the comic data sets used in the previous sketch coloring models are from danbooru. In this data set, each anime character is marked, but this data set is too large. In the case of limited computing power, it is not suitable for using such a huge data set for model training. Moreover, we found through investigation that the data types in the previous comic data sets are very complicated. There may be pure character pictures, pure landscape pictures, and pure animal pictures in the data set. But we know that people, animals, and landscapes are colored differently. So we plan to build our dataset from a single category.

Among the anime works, the most common images we see are anime characters. Therefore, we limit the data images in our dataset to anime character images. For the consideration of model and computer computing power, we did not choose the complex cartoon character full-body image as training data, but chose the cartoon character avatar with relatively simple texture as our training data. This not only facilitates training, but also people have a certain market demand for color anime avatars.

The cartoon data set used for training in this article uses python crawler to crawl about 140,000 color anime face images (Figure4.14) as our cartoon data set. These images are mainly from some anime avatar sites and some third-party open source platforms. These

images are all colored cartoon character avatars. Then use opencv to batch process the image and scale its size to 512 × 512.

In our data set, all data images are anime character avatars, which makes our model better adapted to the coloring of character avatar sketches. In other words, we limit the type of data to improve the ability of our model to color a certain type of line art imageAt the same time, the fixed size also makes it easier for us to import data when training the models.

Figure 4.14 My cartoon data set

But only the comic data set is not enough, we also need to have corresponding line art images to meet our model training. This means that we need to find a way to extract our line art images through comics. Currently there are many edge-detecting algorithms or neural networks. But few of them has good performance on paintings, especially those from comic or animate. Most of these existing methods just detect the edge and then add lines to the edge. However, we need a method to convert the painting to a sketch which looks like a painter drew the outline of picture. It is important when we want to train a neural networks to color pictures. In order for our model to achieve better training results, our sketch data set must be of higher quality. So we compared most methods of extracting line drafts currently on the market.

Here is a conclusion of existing methods to handle this problem.

1) Use opencv and implement a high-pass effect to get the sketches.
2) Use XDoG [27] to get the sketches.
3) Use Paintschainer to get the sketches.
4) Use InstantPhotoSketch software to extract sketches.

5) Use sketchKeras which combined algorithm and neural networks to extract sketches.

Figure 4.15 Sketches extracted by different methods

Take this image(a)(Figure 4.15) as an example. If we use the high-pass algorithm via opencv or something else, we may get the image(b).

Observing the image(b), we can find that its effect is not very good. People like to add shadow to their drawing by add dense lines or points. These are not reflected in this picture.

We use XDoG to get the image(c). Through this picture, we can find that the line thickness is basically the same, and the color density of the line is basically the same. This alone does not exist in our real hand-drawn images. The sketches reflect the richness of the picture through different thicknesses and black and white lines.

Next, we use the line art generation function attached to PaintsChainer to generate the line art we need. PaintsChainer has two different styles for generating sketches, one generates thin lines and the other generates thick lines. We can see their renderings through image(d) and image(e). Unlike the image obtained by XDoG, the line thickness in the line art image obtained by PaintsChainer can be reflected, giving us a rich composition space.

The result from neural networks looks different from those from algorithm. However, this is still not so good. The author of PaintsChainer use threslod to avoid noise and normalize the line. In image(d) and image(e), we can see clearly that the noise, especially near eyes and in the shadow of hair. Threslod can filter some noise but some useful lines is also dropped. But the lines are too coarse and thick.

And then, we use InstantPhotoSketch software to get the image(f). We have found that the sketch extracted through this third-party software is not bad. Although there are some lack of details, it has been greatly improved compared to the previous methods.

Finally, we use sketchKeras based on neural network to extract our line draft and get image (g). We can find that the details of this line draft picture are very good. The line is also very smooth, and the thickness of the line is in line with the hand-drawn traces. Compared with the previous image (f), we can find that image (g) is better than image (f) in details such as glasses, shadow of hair, and eyebrows. So based on the above comparison, we choose sketchKeras to extract our line draft.

In order to avoid the unity of the experiment, we randomly extracted 200 pictures from the cartoon image dataset. We use different methods to extract sketches. The final results all show that the line drawing method based on sketchKeras is closest to the line drawing drawn by hand. We can see more of the comparison results in Figure 4.16.

Figure 4.16 Sketches extracted by different methods

5. Evaluation

5.1 Result of sketch extraction

5.1.1 Test platform of sketch extraction

The test platform for sketch extraction based on sketchKeras is windows10 64-bit operating system, Intel CPU i5, memory 8GB, anaconda 3 integrated development environment, while using tensorflow deep learning framework, keras artificial neural network library, opencv computer vision library .

5.1.2 Experimental data processing of sketch extraction

The comic image dataset we used to extract the sketch was randomly crawled from various anime avatar sites using crawler technology. After manual screening, the data that is not suitable for training is removed, and 140,000 comic images are left (Figure 5.1). We batch process these data to fix its size to 512 × 512.

Figure 5.1 My cartoon data set

5.1.3 Results of sketch extraction

We used the experimental platform in Ch5.1.1, and it took about a month to extract 140,000 sketches, with an average speed of about 20s. Part of the results are shown in Figure 5.2.

Figure 5.2 The results of sketch extraction

We can observe that these sketcher are very consistent with the real sketches. The overall line is smooth, presenting the original texture information well. Especially in the eyes and hair, the subtle textures were extracted. These sketch data which approximating real sketch will provide more help for our training. By matching the sketch with the color image, we get our final data set.

5.2 Result of Our model

5.2.1 Test platform of our model

The test platform for colorization model is Window10 64-bit operating system, Intel CPU i5, memory 8GB, GPU is NVIDIA GTX 1050Ti, anaconda 3 integrated development

environment, while using tensorflow deep learning framework, pytorch deep learning framework, keras artificial neural network library, opencv computer vision library .

5.2.2 Experimental data processing of our model

We use the original comics shown in Figure 5.1 and the sketch images shown in Figure 5.3 as our training data.

Figure 5.3 Sketch data set

Regarding the data set of the refinement model, we did not choose to generate it directly, but used relevant image processing techniques to generate data on the original image during the training process. This is because the error information in the data set in the second stage of training should depend on our random user cue points. We need to add an error message to the original image corresponding to the user prompt. Because the user cannot make changes in the area without errors.

5.1.3 Results of our model

By using the paired data set we made to train our model, we finally got the following output(Figure 5.4).

In order to better demonstrate the advantages of staged coloring, we output the coloring results of our model according to each stage.

Figure 5.4 Results of our model

We can see that the first-stage model(Drafting model) has generated some color images with richer colors for us. This model can only be said to have painted some rough colors on sketch.

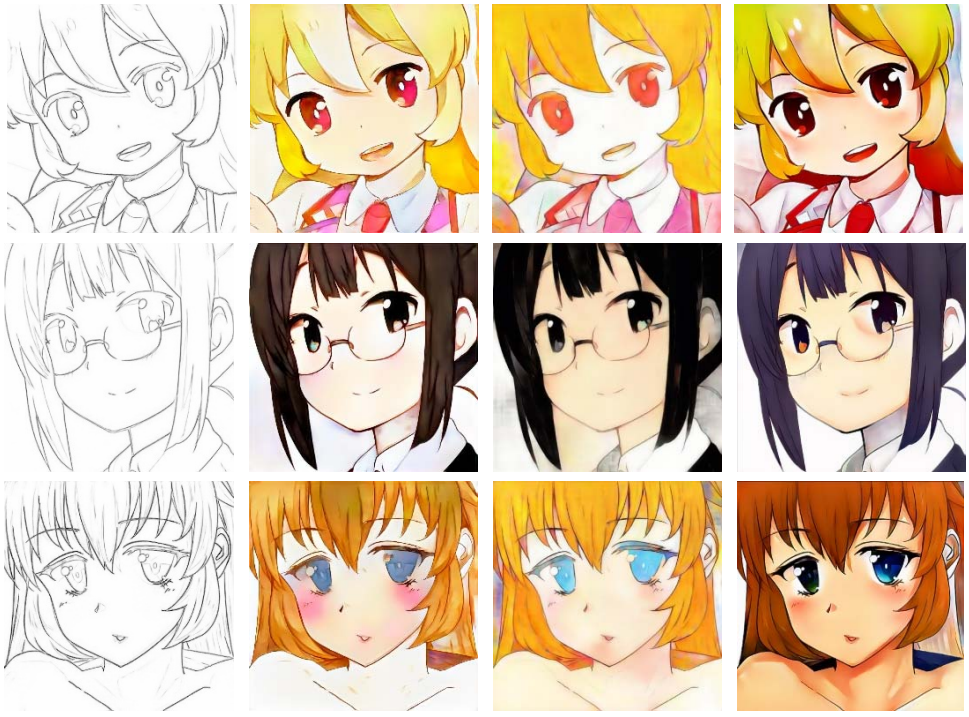
It also has many coloring defects. Let's look at the image generated by the drafting model. Basically, there is a phenomenon of color overflow in each part. In some places, even serious color errors occur, such as the eye area. In addition, we see that there are some erroneous color information in the line parts of some edges, and some areas will

have colors that are not in accordance with common sense. And the overall feeling is not very clear. For the above problems, I use the refinement model to modify these errors.

These color images have been compensated for some details through the second-stage model(Refinement model) refinement, resulting in a color image with better visual effects. First of all, we can intuitively feel that the overall picture quality has been greatly improved than before. The image becomes clearer. Comparing with the previous image, some obvious error messages have been corrected. According to the hint information we entered, the color of the previous eye error was corrected. The areas where the color overflows have also been improved. Although there are still some small color overflows, the visual effects are still in line with people's aesthetics. This means that our model can accomplish the task of sketch colorization.

5.3 Visual comparison

We visually compare the results from our method to that of other methods. In order to reflect absolute fairness, we use the color information of each part of the original picture to color. The same color information was input to our model, PaintsChainerV3, and Style2Paints V4.5 model. The resulting image effect is shown in (Figure 5.5).



(a)Sketch (b)our model (c) PaintsChainerV3 (d) Style2Paints V4.5

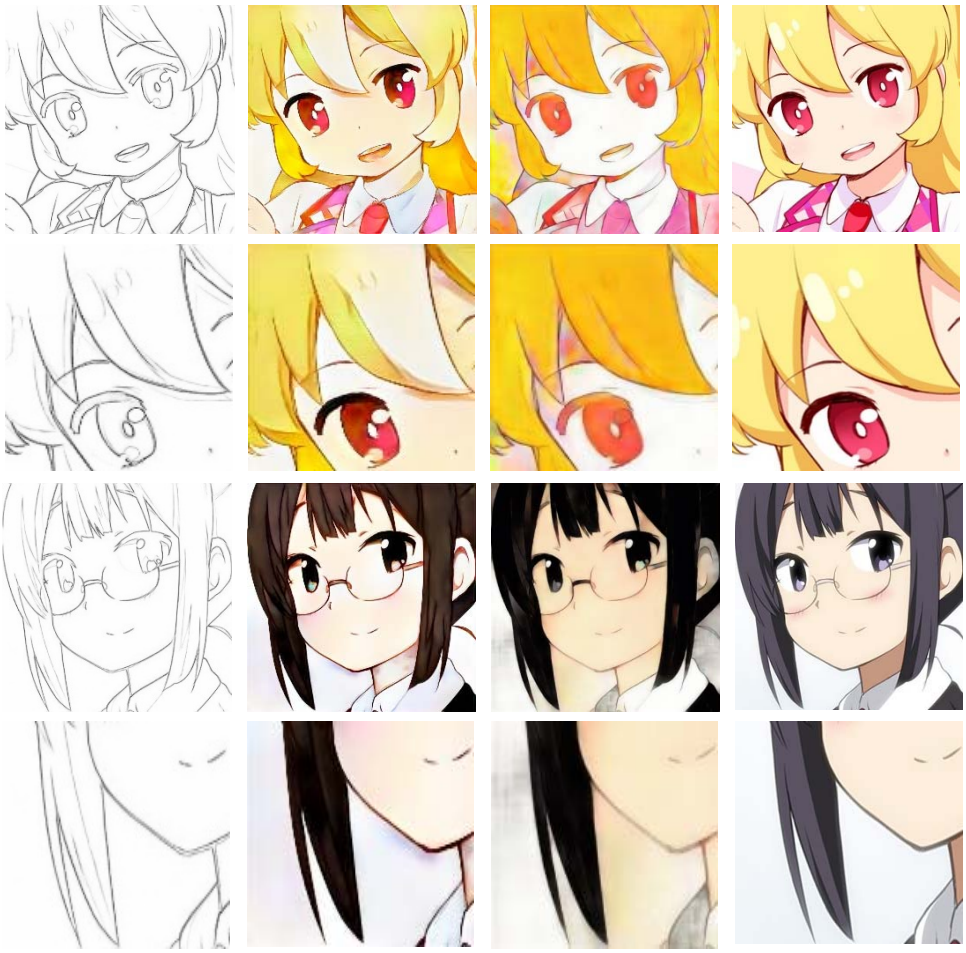
Figure 5.5 Sketch coloring effect of different methods

First of all, we can see that the three models have achieved a good coloring effect on the sketch images. It is hard to imagine that this was done by a machine without prior notice.

And then, We compare our model with the PaintsChainerV3 model. In order to allow us to compare the visual effects of the two more intuitively, we enlarged some areas.

From the overall coloring quality, my model is better than PaintsChainerV3. My model exceeds PaintsChainerV3 in overall clarity. Next, compare the two methods from the details.

We can find that compared to PaintsChainerV3 (Figure 5.6), our model does not appear to have a serious color bleeding problem. Observing the images obtained by PaintsChainerV3, we can see that there are obvious color bleeding problems in in the eyes, skin and other parts of the characters. This is avoided in our model.



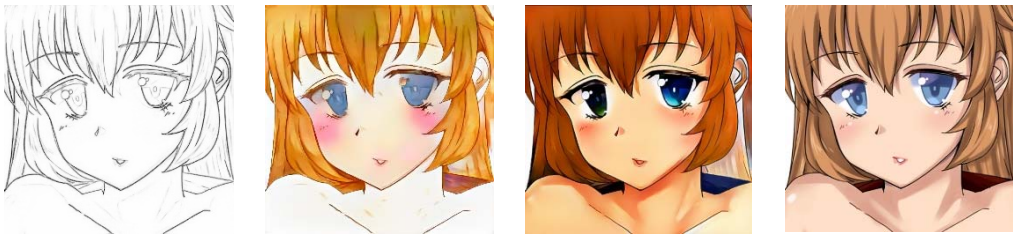
(a)Sketch (b)our model (c) PaintsChainerV3 (d) Original image

Figure 5.6 Compared with PaintsChainerV3

And then, we compared with the original picture. Compared with the original picture, our model still has a certain color overflow. In addition, the overall picture is not vivid enough.

Because PaintsChainerV3 is a model proposed in 2018, it may be a bit behind. After that, we compare our model with the best color model Style2Paints V4.5 currently on the market(Figure 5.7).

Style2Paints also adopts the method of coloring in stages, but unlike our method, it depends on the user's prompt information from the beginning. His ideas are also derived from the work flow of comic artists. Compared with our two steps to complete coloring, he is composed of three steps, namely fill color, add color gradient, add shadow. The last two steps are not available in our model.



(a)Sketch (b)our model (c) Style2PaintsV4.5 (d) Original image

Figure 5.7 Compared with Style2PaintsV4.5

First, both our model and Style2PaintsV4.5 perform a coloring process on the sketch image based on the color information of the original image. We observe the coloring results of both methods. Although there is no obvious problem of color overflow on both methods, the color vividness of Style2PaintsV4.5 is higher than that of our model.

And Style2PaintsV4.5 also contains these shadows and brightness effects, which makes the images produced by the Style2PaintsV4.5 has a strong three-dimensional sense. Looking at the original image, the original image also has this three-dimensional sense. But this is not achieved in our model, which leads to our coloring effect is not as good as Style2PaintsV4.5 visually.

Next, we go to see some details, such as eyes. It can be seen that the eyes of Style2PaintsV4.5 has a gradual color feeling, which looks more vivid. In our model, the coloring of the eye area lacks a sense of gradation.

And then, the most critical point is eyebrows. This is also a problem currently encountered in my model, that is, all eyebrow colors are either the same as the hair or the eyes. This coloring effect is not in line with the real situation. Normally, the color of our

eyebrows is black. We can find this from the colored image of Style2PaintsV4.5 and the original image. This problem requires me to study and solve it in the future.

Last but not least, we found that our model still has a certain gap compared with Style2PaintsV4.5. Whether it is from the color saturation or the rendering of the shadow part, Style2PaintsV4.5 has a relatively good result. Through the investigation of relevant materials, we found that Style2Paints V4.5 adds a unique light and shadow effect, which is not available in our model and is also a part of our future that is worth improving.

conclusion

Since the GAN network was proposed, it has demonstrated its strong vitality in the field of image generation. It learns the distribution of data through confrontation. The so-called confrontation refers to the confrontation between the generator network and the discriminator network. After the confrontation achieves the ideal Nash equilibrium, the network can produce images that infinitely approximate the real samples. It's a magical process from nothing to something. Similarly, the process of coloring our sketches is also a process from scratch. In the real world, manga artists add color to the manuscripts little by little with colored brushes until the final complete work is born. This process is actually a process of continuous confrontation between the human brain and the brush. After we added some color information, our brain went back to determine whether this was the final comic image. If it is not, we will continue to modify it until it finally matches the manga image expected in our brain. This is surprisingly similar to the GAN network. Therefore, we chose the method based on GAN to color our sketches.

With the basic network model, then it is necessary to consider what kind of coloring method of sketch images is the best. By observing the drawing process of some manga artists, we found that they generally give the rough subject color to the sketches, and then constantly modify the details until the final painting is born. This is a divide-and-conquer idea, which divides complex tasks into several simple small tasks. Then we gradually solve the small tasks to complete the original complex tasks. With the help of this idea, we designed our sketch coloring model. Our coloring model is mainly divided into two modules, namely the Drafting model and the Refinement model. These two models correspond to the two stages of painting by the manga artists. The Drafting model is responsible for applying as many colors as possible to the sketch image. The Refinement model fixes as many error messages as possible in the image generated by the Drafting model. In order to enable our model to better repair colored images, our Refinement model is based on user prompts. In this way, we can better prompt the model where the image information is wrong.

With the basic coloring model and network framework, then we are only one data set away from success. Data is the foundation, and any research is inseparable from data. A good data set can greatly improve the accuracy of the algorithm. Therefore, after determining the coloring task of the sketch, we searched for the relevant public data set in the field of sketch colorization of the sketch. But we did not find a data set that matched our sketch colorization task. Because the training sketch colorization task usually requires a sketch and a color drawing which matches the sketch, and the data sets published in the current research direction are only separate color manga images. So we decided to make a

data set by ourselves. We crawled 140,000 color anime images, and then used a method that based on neural network to extract sketches. It costs us about a month to make the data set which we needed. Since our two models are trained separately, the image generated by the drafting model cannot be directly used to train the Refinement model. Therefore, we generate data set on the original manga images based on three error simulation methods to train the Refinement model.

After a long training, our model training is completed. We randomly selected 200 sketch images to test our model. We found that our model achieved the coloring effect we expected.

But we also have some shortcomings. Some of the early parameters are not very reasonable, and it may be better to modify them. After comparing with the best coloring software on the market, we found that our model still needs to be improved. This is also the direction of our future work.

Acknowledgement

Four years as an undergraduate are drawing to a close, and when I looked back through the four years, my heart was full of feelings. Affected by the epidemic, I could only stay at home to finish my graduation thesis. Fortunately, I still have a lot of lovely people around me to help and support me. Without of the help of these people, it could never be completed, thus here I would like to express my sincere gratitude towards them.

First and foremost, I owe my heartfelt thanks to my distinguished and cordial supervisor Prof. Li Haojie, who influenced me with his insightful ideas and meaningful inspirations, guided me with practical academic advice and feasible instructions, and enlightened me while I was confused during the writing procedure.

And I would like to give my sincere gratitude to Prof. Wang Zhihui who with extraordinary patience and consistent encouragement, gave me an interest in academics and great help by providing me with advice of great value and inspiration of new ideas.

Then, I pleased to acknowledge my friends, Yu Senhao and Zhang Yongming . They put considerable time and effort into their comments on the draft. And I am also grateful to Wang Luodi, who gave me a lot of help in my study and life.

Finally, in particular, I would like to express my gratitude to my parents for their support and great confidence in me all through four years.

References

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, et al. Generative adversarial nets[J]. in Advances in Neural Information Processing Systems, 2014,2672–2680.

[2] 李志永. 黑白影像的彩色化研究[D]. 中国科学院研究生院（电子学研究所）.2007.

[3] Cheng Z, Yang Q, Sheng B. Deep colorization[C]. Proceedings of the IEEE International Conference on Computer Vison. 2015: 415-423.

[4] Cao Y, Zhou Z, Zhang W, et al. Unsupervised diverse colorization via generative adversarial networks[C]. Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Springer, Cham, 2017:151-166.

[5] Hensman P, Aizawa K. cGAN-based Manga Colorization Using a Single Training Image[C]. 2017 14th IAPR International Conference on Document Analysis and Recognition(ICDAR). IEEE, 2017,3:72-77.

[6] Yingge Qu, Tien-Tsin Wong, Pheng-Ann Heng. Manga colorization[J]. 2006, 25(3):1214-1220.

[7] Furusawa C , Hiroshiba K , Ogaki K , et al. Comicolorization: Semi-Automatic Manga Colorization[J]. 2017.

[8] TaiZan.2016. PaintsChainerTanpopo [CP]. PreferredNetwork (2016). https://petalica-paint.pixiv.dev/index_zh.html

[9] Lvmin Z, Yi J, Xin L, et al. Style Transfer for Anime Sketches with Enhanced Residual U-net and Auxiliary Classifier GAN [C]. 10.1109/ACPR42740.2017

[10] Mirza M, Osindero S. Conditional generative adversarial nets[J]. arXiv preprint arXiv:1411.1784, 2014.

[11] Long J , Shelhamer E , Darrell T . Fully Convolutional Networks for Semantic Segmentation[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2015, 39(4):640-651.

[12] Hanna M Wallach. Conditional Random Fields: An Introduction[J]. technical reports, 2004, 53(2):267-272.

[13] Stan Z Li. Markov Random Field Modeling in Image Analysis[M]// Markov random field modeling in image analysis /. Springer, 2009.

[14] Yifan L , Zengchang Q , Tao W , et al. Auto-painter: Cartoon image generation from sketch by using conditional Wasserstein generative adversarial networks[J]. Neurocomputing, 2018:S0925231218306209-.

[15] Ci Y , Ma X , Wang Z , et al. User-Guided Deep Anime Line Art Colorization with Conditional Adversarial Networks[J]. arXiv preprint. 2018. arXiv:1808.03240

[16] Xian W, Sangkloy P, Agrawal V, et al. TextureGAN: Controlling Deep Image Synthesis with Texture Patches[C]. CVPR, 2018.00882

[17] Chang H, Fried O, Liu Y, et al. Palette-based Photo Recoloring[J]. Acm Transactions on Graphics, 2015, 34(4):139:1-139:11.

[18] Chen J, Shen Y, Gao J, et al. Language-Based Image Editing with Recurrent Attentive Models[C],CVPR. 2018.00909.

[19] Zou C, Mo H, Du R, et al. LUCSS: Language-based User-customized Colourization of Scene Sketches[J]. arXiv preprint. 2018.arXiv:1808.10544

[20] Martin Arjovsky, Soumith Chintala, Leon Bottou. Wasserstein gan[J]. arXiv preprint arXiv:1701.07875, 2017.

[21] Alec Radford, Luke Metz, Soumith Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks[J]. International Conference on Learning Representations, 2016,795-808.

[22] Zhang, Li Lvmin, Wong Chengze, et al. Two-stage sketch colorization[C]. ACM Transactions on Graphics. 2018: 37(6):1-14.

[23] Richard Zhang, Jun-Yan Zhu, Phillip Isola, et al. Real-time user-guided image colorization with learned deep priors[C]. ACM Transactions on Graphics (TOG), 9(4), 2017.

[24] Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions[C]. Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 1 -9.

[25] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition[J]. arXiv preprint arXiv. 2014: 1409.1556.

[26] Szegedy C , Liu W , Jia Y , et al. Going Deeper with Convolutions[J]. 2014.

[27] Winnemoeller H , Kyprianidis J E , Olsen S C . XDoG: An extended difference-of-Gaussians compendium including advanced image stylization[J]. Computers & Graphics, 2012, 36(6):p.740-753.

修改记录

第一次修改记录：指导老师指出论文中格式的问题，参考文献要规范，如果直接引用别人论文中的图，需要添加引用的标注，图片的命名要准确。

我修改了不规范的参考文献，同时在引用别人的图片进行了标注，并对自己的图片进行了准确的命名修改。

第二次修改记录：指导老师指出页眉要一致，摘要部分精简一下，最好分成三段去写。

我将全文的页眉进行了修改，同时精简了摘要部分，按照结构层次进行了摘要叙述

第三次修改记录：指导老师指出，第二章中 It(Figure2.1) can be seen 语法错误，有很多地方首字母大小写不对，摘要里讲了 14 万数据集，怎么获取的，有什么特点？要介绍一下这个数据集。

我将 It(Figure2.1) can be seen 修改为 It can be seen from Figure2.1，同时将全文中首写字母不对的地方进行了修改，然后在章节 4.4 当中对 14 张漫画数据集进行了说明介绍。

第四次修改记录：评阅老师指出，第一章加入过多别人成果图，建议去掉，用文字描述即可；部分图表没有明确引用；部分公式符号没有解释；第五章是核心部分，结果图足够，但是对应解释说明较少，建议添加适当文字说明和分析；部分文献格式不规范，如[14]-[20]建议按照模板修改。

我去掉了过多的他人成果图，用文字描述代替。明确了图表的引用，解释了公式符号，然后对第五章添加了相应的解释说明和分析，最后规范了一下文献格式。

记录人（签字）：徐仕卓

指导老师（签字）：