CS229 Final Project
Automatic Colorization for Line Arts

Kaiyi Fu, Yang Wang, Baige Liu
Computer Science Department
Stanford University
Stanford, CA 94305
Email: alexkfu6, leonwang, liubaige@stanford.edu

***Abstract*—Coloring line art images has been a quite popular topic since artists are trying to create colorful images from black and white scratch. Previous methods focused on using hints to generate colors. We realize that the same image can be colored with multiple styles, therefore, we do not limit ourselves in trying to recover the original image. In our project, we intend to create a automatic colorization algorithm that can automatically generate hints. We implemented three methods, 1) CNN; 2) GAN; 3) WGAN. We also propose using color histogram in generating hints. With the help of these three methods, we are able to recover color line arts. Results for three methods are presented after the experiment to show the validity of our work.**

I. Introduction

Normally, when animation artists try to create a animation frame, they first scratch the general skeleton of the content and fill in the color accordingly. The job of colorization is very onerous since a picture may have hundreds of partitions and each may require a different color. Not to mention when trying to create a animation film which runs at thirty frames per second. While each frame needs to be colorized carefully, there will be 1,800 frames per minute which is a huge number. Here, if we can let computers do the work for us, huge amounts of time and effort will be saved. An illustration can be shown in figure 1.

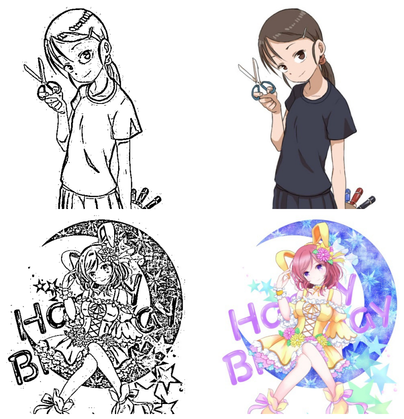

Fig. 1: The left images are line arts that we want to color; the right images original images.

Therefore, we are trying to create a tool that can automatically colorize the pictures. Suppose we want to colorize an image with skeleton provided by an artist using similar coloring technique from that artist, we may first train the model of colorization using the images colored by that artist. These images can be either previous animation films with similar style or the preceding frames from current animation film.

Using deep learning strategies, researchers are trying to find methods to color an image from black and white images. [2] tries to use DGAN (deep convolutional generative adversearial nets) to tackle this problem. [2] approaches this problem by treating it as a classification task, which also uses a CNN to evaluate the coloring images. [8] applies multimodal color distributions and uses a histogram to recover colored images from black and white images. There are also other work that focuses on coloring black and white images which will not be addressed here.., yet we have one more constraint which is to make the style of coloring as consistent as possible throughout the film.

Previous approaches have used some hints to help coloring, for example, if one wants to choose a specific style, he or she may need to give the system a strategy for completing the job [5]. In our implementation, we hope to avoid any hint, that the system is running totally voluntarily. We try to learn these hints automatically.

A. Contribution of this paper

In this proposal, we will present our modified algorithm for CNN, GAN and WGAN. We will also present our color histogram which is later used to generate hints during coloring process. Together, we proposed a automated coloring system.

B. Origination of this paper

In section 2, we will represent previous algorithms and models. We will also define problem we are trying to solve. In section 3, we will propose our algorithm for automated colorization. In section 4, we will list our dataset and present our result for training. In section 5, we will discuss our result, conclude this paper and present future work.

II. Previous Approach

A. Problem definition

Our problem can be formulated as follows. Given a skeleton (line art), i.e., a set of edges, for a particular graphs, we needs to find the best possible color for each sector. Mathematically, for a sector S, we try to find the best color c under circumstance given by $\mu(S)$.

$$\begin{aligned} c &= \arg\max_c P(c|S;\mu(S)) \\ &= \arg\max_c \frac{P(S;\mu|c)P(c)}{P(S;\mu)} \end{aligned} \tag{1}$$

Here, the environment variable $\mu(S)$ gives the information about sector S. For example, we need to know the location χ, size ζ, shape η and hint ρ. μ is related to the three variables by some function f.

$$\mu(S) = f(\chi(S), \zeta(S), \eta(S), \rho) \tag{2}$$

Here, we cannot directly use L2 loss because there is no specific color for a specific pixel as we can see in real life where an image can be colored with different colors and still be beautiful. In such way, we may transform the learning problem into a unsupervised learning problem that we are not trying to fully recover the original color but instead try to find the best color according to the information provided by χ, ζ, η and ρ.

B. Multimodal model

The first model we can think of is multimodal model [10]. Previous work has already shown the ability of coloring grayscale image [3, 12]. In our problem, this method cannot be directly applied. In those work, a multimodal is used with cross entropy to detect scheme. However, in the multimodal, all images are grayscale images, which provides some sorts of information about the possible colors. In our expression, it can be included in the hint rho. The intensity, which is provided in grayscale images, can infer the color of the objects. Since we do not have that detailed texture which exists in grayscale images, we have to search another way. In their work, it is clearly shown that if the intensities of some pixels are low, the corresponding color for those pixels will be white. In other words, if one pixel in the original image is white, the output image after colonization will also be white. When coming to coloring animation images, most parts of the line arts are white; therefore, the output of the corresponding image will be mostly white, which disobeys our goal for solving the problem.

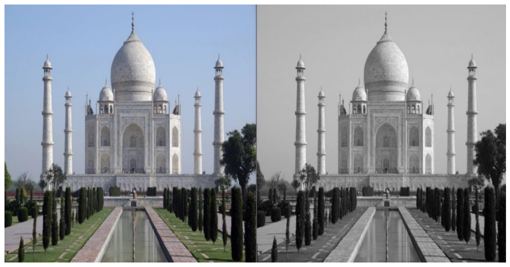
Fig. 2: The left image is the colored image while the right is the grey-scale image

C. Generative Adversarial Networks

In Generative Adversarial Networks (GAN), we are trying to find the optimal point in a minimax setting as in equation (3), where the discriminator aims to distinguish between the real images and the fake images. And the generator aims to generate realistic images that hopefully could fool the discriminator.

$$\begin{aligned} \min_G \max_D V(D,G) =& \mathbb{E}_{x\sim p_{data}(x)}[\log(D(x))] \\ &+ \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))] \end{aligned} \tag{3}$$

Ian Goodfellow et al. proposed the algorithm 1 to train such a loss function [6]. In practice, the paper [6] suggests that, instead of minimizing $\mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))]$, we maximize $\mathbb{E}_{z\sim p_z(z)}[\log(D(G(z)))]$, which provides stronger gradients early in the training. From previous research, researchers found that adding $L1$ regularization to the generative loss function could improve the performance of the GAN [9]. Here, we can derive the gradient of both discriminator and generator in and separately.

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} [\log D(\vec{x}^{(i)}) + \log(1-D(G(\vec{z}^{(i)})))] \tag{4}$$

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(1-D(G(\vec{z}^{(i)}))) \tag{5}$$

Algorithm 1 Minibatch stochastic gradient descent training of GAN

```
1: procedure TRAIN GAN()
2:     for i=1:number of training iterations do
3:         for j=1:k steps do
4:             Sample minibatch of m noise samples
5: $\vec{z}^{(1)}, ..., \vec{z}^{(m)}$ from noise prior $p_g(\vec{z})$
6:             Sample minibatch of m examples $\vec{x}^{(1)}, ..., \vec{x}^{(m)}$
   from data distribution $p_{data}(\vec{x})$
7:             Update the discriminator using gradient accent
   by the equation (4)
8:         Sample minibatch of m noise samples $\vec{z}^{(1)}, ..., \vec{z}^{(m)}$
   from noise prior $p_g(\vec{z})$
9:         Update the generator using gradient decent by the
   equation (5)
```

In GAN, we want to find an equilibrium point of a minimax game. It uses a gradient descent method which is designed to find lower value of a loss function, thus the convergence may fail in finding the equilibrium point. Many efforts have been made to stabilize the training of GAN. For example, Saliman proposed adding Gaussian noise to each output layers[11]. In our experiments, we found that adding Gaussian noise to the discriminator significantly stabilized the training for both generator and discriminator.

There are few common pitfalls one may encounter during training GAN. For example, mode collapse, the generator may only learn a specific narrow distribution, which ends up being lack of diversity. There are also some more sophisticated improvements such as WGAN[1] which aims to improve this. Researchers claim that it will remedy the problem of mode collapse and help stabilize the training.

III. Our algorithms

A. Hints

As discussed before, directly inferring color from white-black edges is not nearly impossible since we do not have enough information as we did in grayscale images. Alternative method has been brought up in coloring line arts. Such as using hints[4]. These hints can either be intensity from grey-scale images or manually input to the system. Here, we will adopt the same techniques that use hint to help, yet in a different way. Here we will introduce the concept of color histogram. We went through all the images in the dataset and study the distribution of colors. In this study, we try to find the colors that were used most. To simplify the color histogram, we evenly shrink the rgb space which ranges from 0 to 255 for each channel into 512 bins. Here, we use 8 levels to distribute each channel, therefore creating 512 bins. When counting the number of each bin, we go through each pixel and throw it into corresponding bin. When reconstructing the hints, we may use the average color for each channel in current bin when projecting back onto the original rgb color space.

After constructing the color histogram, we found the white (bin 512) to be the most frequent color and holding a frequency much higher than other colors. This makes sense since most of the backgrounds are white. During training, when acquiring hints through blur, we first randomly white out some small patches of the image and then fill colors according to the distribution of the color histogram.

B. Loss Function

We have used two loss functions in our approach. The first one is L2 loss which is used in CNN. The expression can be found as in equation (6). In this equation, we are trying to learn how to recover the original image.

$$L = \sum_{i=1}^{m} \|\vec{c}_i - \vec{c}_i^{0}\|^2 \tag{6}$$

When implementing GAN, we add a pixel by pixel L1 regularization to the original GAN loss as shown in equation (3)

$$\begin{aligned}\min_G \max_D V(D,G) =& \mathbb{E}_{x\sim p_{data}(x)}[\log(D(x))] \\ &+ \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))] \\ &+ \lambda|(x - \bar{G}(z))|\end{aligned} \tag{7}$$

C. CNN

We trained a baseline model using convolutional neural network following a similar network architecture design as mentioned in [4] with multiple convolution and RELU layers. However, we want to make our task more difficult and get a stronger baseline. So we also provide the model with color hint images, which is consistent with the following GAN models for them to be more comparable. In addition, we added two residual connection layers between the convolution and deconvolution layers. We used L2 loss and Adam optimizer to train the neural network. We run the model for 70 epochs for the model to fully overfit the training data.

D. GAN

First we directly implemented GAN using the loss function (7). The structure for our GAN is shown in figure 3.

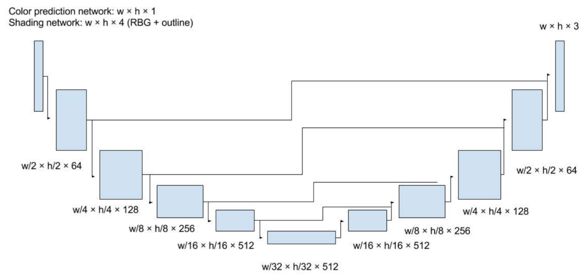

Fig. 3: Structure of GAN

E. WGAN

We then implemented WGAN[1] as proposed in the paper, except that we trained the discriminator and generator equally frequently. This is tue to the fact that the generator relies on the training data through edges and blurred hints. We clipped the weights of discriminator after every gradient descent update between [-0.01, 0.01].

The results of WGAN is not as good as our original GAN approach since it is a baseline WGAN model. The color of results from WGAN are not as bright as our previous GAN results and it is more blurry. In the future, we may improve the training of WGAN with gradient penalty as proposed in this paper[7] and other techniques discussed in this paper[11].

F. Overall Algorithm

The overall flowchart for the algorithm can be shown in figure 4. In general, the input is a line art image and it will go through color hint predictor which is a GAN and which generates a hint according to color histogram or an automatically generated color hint. The hint will be passed to color predictor network, which is also a GAN. Final image will be generated using color predictor.

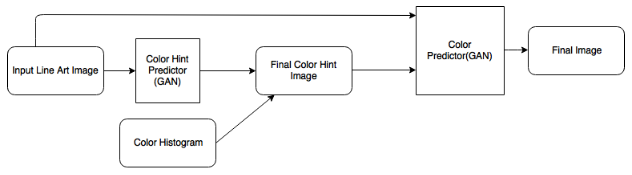

Fig. 4: Flowchart for overall algorithm using GAN

IV. Experiment Result

A. Data Set

We have download 10,000 images from ”https://safebooru.org”. We first went through the color histogram to find the distribution for each color or pixel-color. Then we used canny edge detector with different variance to detect the edges. In such way, we create line-art images from original datasets. Normally, we will choose a variance $\sigma = 2.5$.

B. Color Histogram

We first build the color histogram of all images. We found the following distribution as shown in figure 5. As suggested earlier, the most frequent color is white and therefore the highest peak is at bin 511, which is white. Also a list grid of top 100 most frequent colors are shown in figure 6.

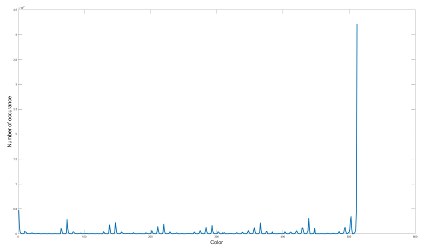

Fig. 5: Color Histogram

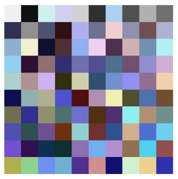

Fig. 6: Top 100 most frequent colors

C. Result for Colorization

We have presented results for coloring images using different methods in figure 7. The test images are randomly picked. We compared results using CNN baseline, GAN, and GAN using predicted color hints. When using CNN, we found the colors are not satisfactory where we have extremely low color saturation even if we give the model color hints. Using GAN methods, there are more colors generated, and the generated images have similar colors with the original image. We also have results in the second last row, where we applied automatically generated hints, by which we generated different kinds of colors but still looks quite realistic. In some cases, it generates better images than using hints from original images. For example, in the last column, the face in the last third picture are too white to recognize while the last second row has better generated images. In the last row, we represented our results of WGAN, it is not as bright as the pervious GAN, but has a kind soft effects that looks generally good.

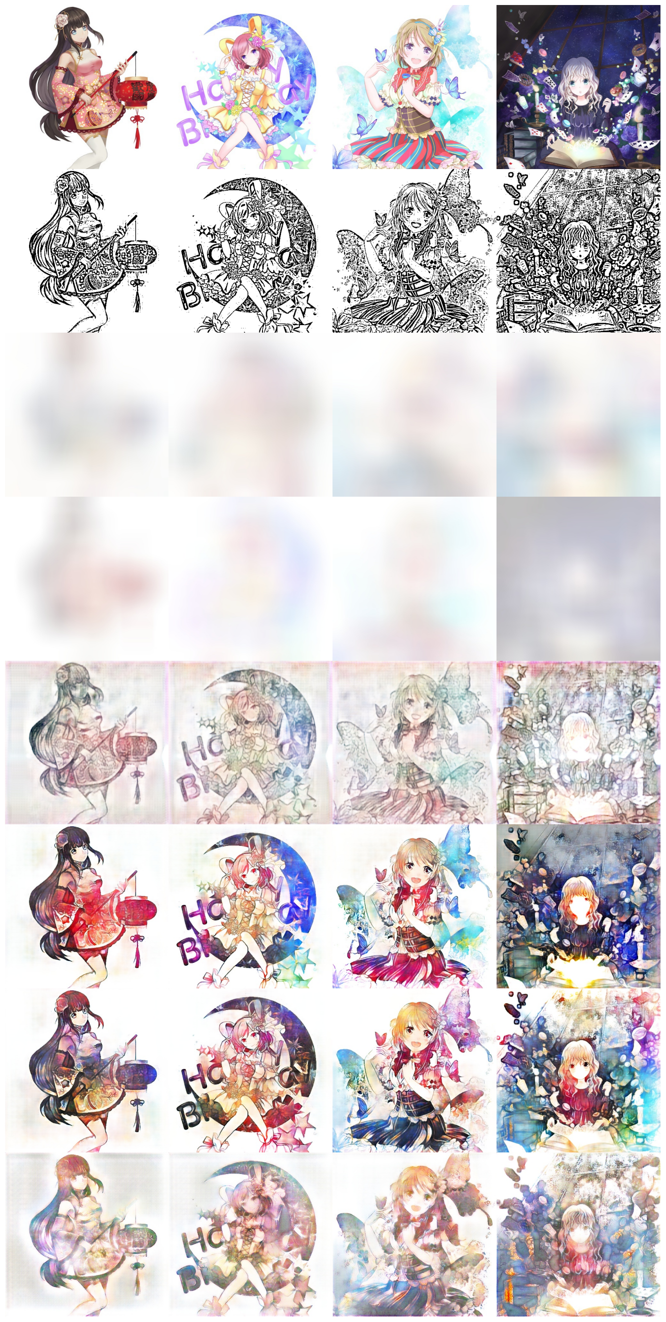

Fig. 7: Images from top to bottoms: original image, line art image, guessed color hint image, color hint image(orginal image blur), CNN base line colorization image, GAN colorization image, GAN with guessed hint colorization image, WGAN with guessed hint

D. Training loss

Here we only have training loss. Note that since we do not have a ground truth for each image as images may have different colors that may still look brilliant, it makes not much sense to record validation loss. The performance can only be evaluated by user preference. After adding Gaussian noises to the discriminator, we are able to stabilize the discriminator and generator loss. The original generator loss will increase after certain epoch. This is due to the fact that as discriminator gets very stronger, the generator does not know how to improve, thus the loss starting to get bigger. The corresponding results are shown in figure 8

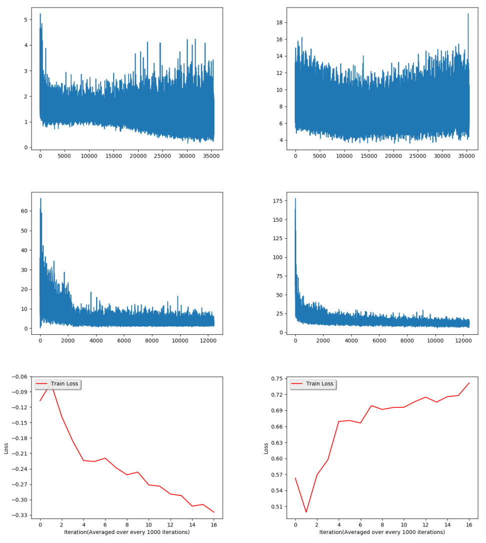
Fig. 8: From left right are discriminator loss and generator loss. The first line are the original training loss plots, the second line were loss after adding Gaussian noise .The third line is for WGAN, from left to right is corresponding to discriminator loss and generator loss. We took average for the WGAN loss for every 1000 steps, since the original loss are too noisy.

V. Conclusion and Future work

In our implementation, we have successfully implemented the automatic generator. In previous sections, we presented our implementation using CNN and GAN separately and the result are quite different. In our work, we do not require the hint in our methods. We designed different schemes for generating hints. First, we used color histograms to generate a color which is most likely to be used by artists. We also use CNN used blurred image as hints. GAN uses either blurred image as hints or uses another networks to learn hints automatically. We also added some noise to the GAN model which achieves a better result. We achieved our goal of generating a colorful images automatically.

In the future, we will try to improve our baseline model of WGAN as a short goal. For long time goal, we aim to color subsequent images for animation movies. In animation films, the color of each image will be strongly related to previous images which put a high requirements for hints. We will build a more robust image colorization in the future. Meanwhile, currently, our method is a pixel based colorization. In the future, we may build a sector based colorization where the change in color in one section will be less than it is now.

VI. Contributions

Kaiyi implemented the color hint histogram method and helped set up the experiment environments. He also contributed a lot in the final write-up. Baige implemented the baseline model and helped on the sampling phase of all the models. Yang proposed new ideas of training our GAN, implemented the various GAN models and recorded the experiment data. For other parts such as discussions, the works are divided equally.

References

[1] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein GAN. *ArXiv e-prints*, January 2017.

[2] Yun Cao, Zhiming Zhou, Weinan Zhang, and Yong Yu. Unsupervised diverse colorization via generative adversarial networks. *CoRR*, abs/1702.06674, 2017. URL http://arxiv.org/abs/1702.06674.

[3] Guillaume Charpiat, Matthias Hofmann, and Bernhard Schölkopf. *Automatic Image Colorization Via Multimodal Predictions*, pages 126–139. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-88690-7. doi: 10.1007/978-3-540-88690-7_10. URL https://doi.org/10.1007/978-3-540-88690-7_10.

[4] K. Frans. Outline Colorization through Tandem Adversarial Networks. *ArXiv e-prints*, April 2017.

[5] Kevin Frans. Outline colorization through tandem adversarial networks. *CoRR*, abs/1704.08834, 2017. URL http://arxiv.org/abs/1704.08834.

[6] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Networks. *ArXiv e-prints*, June 2014.

[7] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville. Improved Training of Wasserstein GANs. *ArXiv e-prints*, March 2017.

[8] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Learning representations for automatic colorization. *CoRR*, abs/1603.06668, 2016. URL http://arxiv.org/abs/1603.06668.

[9] Y. Liu, Z. Qin, Z. Luo, and H. Wang. Auto-painter: Cartoon Image Generation from Sketch by Using Conditional Generative Adversarial Networks. *ArXiv e-prints*, May 2017.

[10] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y. Ng. Multimodal deep learning. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pages 689–696, USA, 2011. Omnipress. ISBN 978-1-4503-0619-5. URL http://dl.acm.org/citation.cfm?id=3104482.3104569.

[11] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved Techniques for Training GANs. *ArXiv e-prints*, June 2016.

[12] Richard Zhang, Phillip Isola, and Alexei A. Efros. Colorful image colorization. *CoRR*, abs/1603.08511, 2016. URL http://arxiv.org/abs/1603.08511.