Automatic Manga Colorization with Hint

Honghao Wei
Stanford University
weihh16
weihh16@stanford.edu

Yiwei Zhao
Stanford University
ywzhao
ywzhao@stanford.edu

Junjie Ke
Stanford University
junjiek
junjiek@stanford.edu

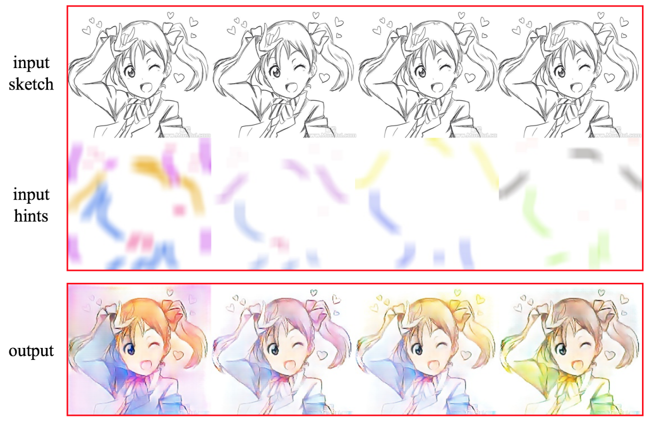


Figure 1: Our Colorization Result (from line sketch)

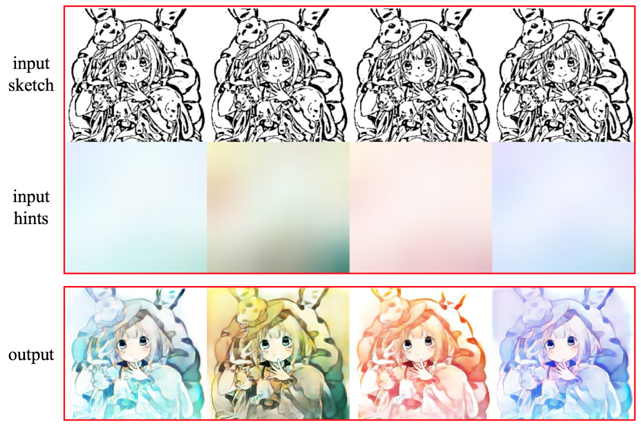


Figure 2: Our Colorization Result (from detected edge)

Abstract

Learning to generate colorful manga pictures from line-art image is not only interesting, but also a practical application in industry to save manual work and increase animation quality. In this paper, we investigate the sketch-to-image problem by using uNet [16] and conditional generative adversarial networks (cGAN) [12, 4]. After combine GAN's classification loss with l_1 loss, high level feature loss and TV loss, by using improved WGAN training method, our model successfully generate colored mangas from a sketch, which is compatible with users preference as color hint. Experiments show that our model can produce high-quality images both visually and numerically in terms of l_2 distance and KL distance evaluation.

1. Introduction

In this paper, we tackle the issue of automatic colorization with users' preference as color hint. The problem of current convolutional network model is that it tends to average the colorization with unsharp boundary and blur the background. Moreover, current models mainly focus on colorization for grayscale images, but line-art images contains less information than grayscale pictures and is thus more challenging. Producing colored image staightly from line-art image also satisfies the need of designers and painters. The users may have preference for the color and style of the image and the model should be able to learn to adapt to the color hints as additional information provided by the user. For instance, when user brushes the eyes of cartoon figure to be red, we infer white-color hair would be favored enlightened by the pattern learned from large volumes of manga image data. We use uNet, a practical application of ResNet, to train an end-to-end model. We also use conditional generative adversarial networks (cGAN) to train generative models. Both the generator and discriminator will be conditioned on the input of both the line-art image and the provided color hint. To maximize the advantage of color hint and to mitigate the negative effects of bad color hints, we reduce the network's dependency on the color hint with different approaches. Last but not the least, we introduce various kinds of loss to get clean and unblurred background and colorization results. For instance, the color of different areas spit by line should not be mixed together.

2. Problem statement

Problem Given color hint and line art image, we colorize the sketch.

Dataset 20000 colored manga images from `safebooru.org`

Expected results Generated colorful images from cGAN trained with WGAN strategy

Evaluation In addition to visually compare the generated results with original images, we also compute their l_2 distance and Kullback-Leibler divergence (KL distance) which are numerical measurements of how closely of the generated results and the original ones.

Input line art image (not greyscale), and color hint

Methods U-net, GAN, cGAN trained with WGAN strategies

Output colorized manga image in correspondence to users' color hint

3. Related Works

Colorization problem, especially generating colorful image from gray images, have been researched for a long time. Levin et al.[9] proposed an effective and straight-forward method that incorporates colorization hints from the user in a quadratic cost function, knowing that the spacial distribution of color should be smooth. However, this method is not data driven and can not take advantage of big data we have.

On the other hand, convolution Neural Network has become successful in different high level understanding tasks such as classification, segmentation and video caption. Encourage by the success of the CNN, people also apply this method to image processing task such as getting super resolution version of the image [2]. Using CNN, Zezhou et al.[1] provide a CNN based, complex methods that can generate colorful images based on the input gray images.

However, generate colorful images from line sketch is much harder. Line contains less information and the network need to be creative to generate something they don't know, which seems impossible for end to end CNN model. Generative adversarial networks (GANs) were recently regarded as a breakthrough method in machine learning, which uses two adversarial network trained together to generate creative result [4] [15]. Using GAN conditional on some specific input[13], Phillip et al. created Pix2Pix network which translation network that can map one kind of images to another style, including producing city images from map, transforming the image of daylight to night, and create real shoes and handbags images from sketches[8]. PaintsChainer [14] is a project that can transform line sketch to colorful manga using unconditional GAN. But it is trained on a special training set where there are line sketch and their related colorful images in pair and this data is relatively hard to get. Kevin Frans[3] and Liu et al.[10] also tried method that firstly do edge detection to get the line sketch and then using conditional GAN to get the colorful image with hints. But their results are not as clear as that of PaintsChainer and seem more blurred.

4. Dataset

We collected 20000 colored manga images from `safebooru.org`, an hourly-updated anime and manga picture search engine. We collect data with our own python code and the dataset is original. We choose it as datasource because it provides high-quality manga images with different painting styles and therefore our model can learn from different painting patterns of images.

We separate 14000 images for training set, 3000 images for validation set and 3000 images for test set. For preprocessing, we would resize each image to 256 * 256 pixels, remove the alpha channel and convert it to a 3 channel BGR color image. We also use normalization to transform the original intensity values into desirable range of (0,255). We do not adopt augmentation techniques in this work for the following two reasons. First, the previous work trains on relatively small dataset, such as Minions, which contains 1100 pictures of different colored minions. Second, we want our model see more painting and colorization style in manga, simply do rotation and translation would not help.

During training, we use image itself as feature and the only other feature we extract is the edge information. Please refer Section 4.2 for details of edge detection and Section 5 for example of dataset.

5. Technical Approach

5.1. U-Net

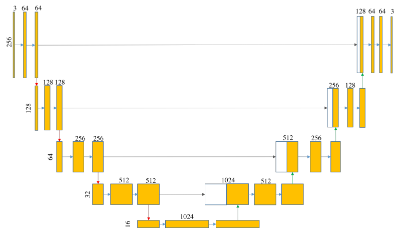


Figure 3: U-Net architecture

We use the U-Net3 encoder-decoder architecture as our network, which is illustrated in Fig 3. There are mainly two different parts of the 'U'-shaped network: the encoding part on the left of downsampling and the decoding part on the right. In the encoding part, we use 5 repeated units including two 3 * 3 convolutional networks with stride of 1 and one 2 * 2 maxPooling layer with stride of 2. Each convolutional networks is followed by a rectied linear unit

(ReLU) and Batch Normalization (BN) layer. The decoding part is made up of one convolutional transpose networks and two convolutional networks in each step and this is repeated for five times. Each convolutional transpose network has a 2 * 2 deconv kernel size and half number of channels. The two convolutional networks after convolutional transpose has the same size, activation and normalization as those in contracting path. In order to allow the informaion flow straightly from the encoding phase to the decoding phase, we also add a direct path from the preceding encoding convolution to the decoding convolution transpose. This is done by concatenating the corresponding convoluted results in encoding part with the upsampled results in decoding part. At the end, we use an additional convolutional transpose layer to map the hidden layers to outputs with same shape of inputs, which is our generated colored image. In total, there is 18 convolutional networks and 5 convolutional transpose networks.

The decoding part is symmetric to the encoding part in this architecture. There is no fully connected layer and thus we could use parameter sharing to reduce the number of parameters to learn. Moreover, U-Net is a practical application to ResNet[7], since we concatenate the upsampled results with corresponding 'inputs' before each maxPooling. In this way, we can take advantage of ResNet to alleviate difficulties in approximating identity mappings by multiple nonlinear layers.

5.2. Edge Detection

The ideal way to train our model is to use pairs of the original line-art image and the colored image. However, it is very hard to find a large data set with both the colored manga and their original line-art image. So we need to generate the sketch image from the colored manga.

A straight forward way to do this is using edge detection, which is widely used to find the boundary of objects. It works by detecting discontinuities in brightness. In our work, we use the edge detection techniques to turn a colorful image into line-art image, and use the edge detection results as inputs to U-Net in conditional adversarial learning.

Due to time limit, we only try OpenCV for edge-detection but we believe a good edge detection methods could significantly improve the model performance since the outlines and boundaries of objects would become clearer and more accurate. Example of edge detection results is showed as below. It indicates the edge detection result is very similar to the original line-art image.

5.3. Color Hint Generation in Training

In conditional adversarial learning, we input the users' color hint as the condition of our generator and discriminator. During training, the color hint is generated by blurring and sampling color blocks from the original image.

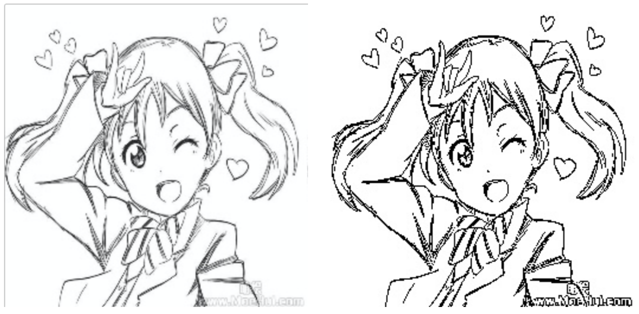

(a) Original line-art (b) Edge detection result

Figure 4: Comparison between line art and edge detection results

We notice that bad quality color hint would significantly undermine the final colorization results. Therefore, how we generate color hint during training phase is very important.

There are two problems we need to tackle. Firstly, since generated images depend on the color hints, we need to figure out a solution when there is no color hint. A bad color hint processing method would result in whiten images with generator doing almost nothing. Moreover, it is a practical concern that when user give color hint in real life, they may not colorize their intended areas perfectly along the boundary. For example, if the user chooses the hair of cartoon figures to be yellow, it is possible that some part of hair is left white while in some areas near boundary, the background would be mistakenly brushed yellow. In this way, a bad color hint would make both the hair and background messy and blurry.

To remedy this negative impact of bad quality color hints, we propose three methods.

Random white-out To tackle the issue, we need to reduce the dependency of model over color hint. One possible methods is to reduce the information color hint contains and we do this by randomly whiten a small area of image before blurring. Currently, we choose to whiten thirty randomly-selected 10 * 10 px areas.

Gaussian filter We initialize a convolutional kernel with Gaussian distribution and apply it over original colorful image. We keep the original size by padding and blur the image with convolution operation.

Random color block selection In order to give some solid color hint in addition to the blurry information, we directly select some random color blocks of the original images and left all the other areas as the blurred color hint map mentioned above. In this way, we would get accurate color hint in some specific location. This color hint is similar to the way users give color hints by using color blush to paint some zone in the image to be a certain color (such as brushing the hair to be yellow, the bow tie to be red etc.)

The final version of our model use a combination of the method described above to 1) preserve the general color information of the original images. 2) reduce the dependency of model over color hint by whitening. 3) preserve some detail information of the original images.

5.4. GAN

It's very hard to directly use neural network to learn how to colorize the picture because it's hard to define a good loss function that represents the quality of the image. Inspired by Goodfellow et al. [5], we use the generative adversarial networks framework to let the network learn colorization through the battle of generator and discriminator. The overall training process can be shown in Fig 5

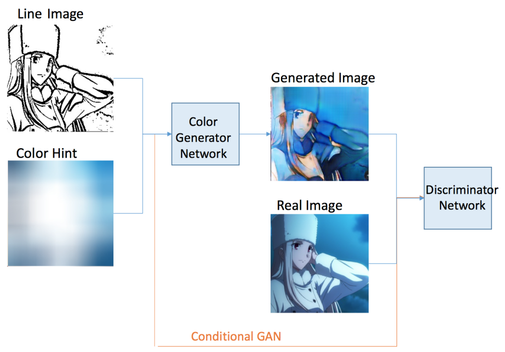


Figure 5: Overall process of cGAN method

The framework of GAN contains two neural networks, the generator network $G(z;\theta_g)$ and discriminator network $D(x;\theta_d)$. $G(z;\theta_g)$ is the network that estimates the generative distribution $p_g(x)$ over the input data x, where z is noise add to the input sketches. Meanwhile, the network $D(x;\theta_d)$ is used to make a judgment of whether the image is from the real dataset or the one generated by $G(z;\theta_g)$. It maps from point x in data space to a probability.

We use a similar U-net network as $G(z;\theta_g)$ which directly generates a colored image from sketch and color hint. The discriminator $D(x;\theta_d)$ network is realized by a 2-classes classification convolutional neural network.

To train two neural networks, we define the loss to be the cross entropy of the classification:

$$\nabla_{\theta_d}\frac{1}{m}\sum_{i=1}^{m}(\log D(x^{(i)};\theta_d)+\log(1-D(G(z^{(i)};\theta_g));\theta_d))$$

Since the generator $D(x;\theta_d)$ is trying to foo the discriminator, it would want to increase the loss in each training step. On the other hand, the discriminator's job is to distinguish real from fake and it would want to decrease the loss. This adversarial process is realized by updating the discriminator by gradient ascend while updating the generator by gradient descend.

There are several ways to add color hints to the framework of GAN. The most straight forward way is to concat the color hints with the sketch, making the generator to be $G(z, z_{hint};\theta_g)$. We can also try a conditional GAN [13]. Intuitively, this allows the discriminator $D(x;\theta_d)$ to make judgement conditioned on the color hint, and therefore becomes $D(x, z_{hint};\theta_d)$.

The random input z to our GAN is intergreated on our U-Net model by adding a 1×100 size placeholder as a random input channel and linearly map to a feature map of the size of the origial image. This random matrix is then concatenated with the original input and feed together to the UNet. As for the discriminator, we use a simple multi-layered CNN, the structure of which can be shown in Fig 6

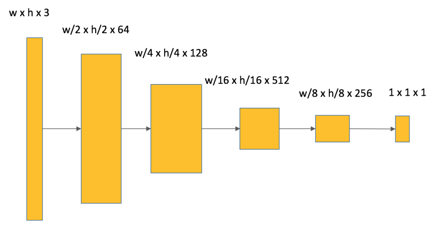


Figure 6: discriminator network architecture

5.5. Improved-WGAN training strategy

The original version of GAN is hard to train in practice because it is hard to balance the training of generator and discriminator. When the discriminator is too strong, its loss will soon become zero while the generator keeps failing but doesn't know how to improve. To solve this problem, we apply the training strategy named Wasserstein-GAN (WGAN) method [11]. In WGAN, the discriminator aims at minimizing the Wasserstein distance (W-distance) between real and generated data, which provides a useful metric of convergence and image quality.

The final generator loss function and the discriminator loss function are defined as

$$L(G) = -\mathbf{E}_{x\sim P_g}[D(x)]$$
$$L(D) = \mathbf{E}_{x\sim P_g}[D(x)] - \mathbf{E}_{x\sim P_r}[D(x)]$$

W-distance can be approximated by the negative value of the discriminator loss. Since lower W-distance would correspond to higher quality images, we use this as one of our evaluation metrics.

We follow the improved WGAN training strategy [6] to use gradient penalty to enforce the Lipschitz constraint. The gradient penalty samples from the interpolation area between the real and generated distribution and then enforces the gradient norm of the discriminator around 1. The corresponding discriminator loss function can be computed as

$$L(D) = \mathbf{E}_{x\sim P_g}[D(x)] - \mathbf{E}_{x\sim P_r}[D(x)] \\ + \lambda \mathbf{E}_{x\sim P_\xi}[||\nabla D(x)||_p - 1]^2$$

5.6. Axillary Generation Loss

In conditional generative adversarial learning, the loss is extended to be value of two-player min-max game conditioned on additional inputs, sketch in our case. However, we believe it is not sufficient to evaluate our model with standard GAN loss only. Previous approaches of conditional GANs have found it beneficial to mix the GAN objective with a more traditional loss functions [10]. As compensation, we introduce three additional losses.

L1 pixel-to-pixel loss We introduce L1 loss instead of L2 loss over pixel-wise level because L2 loss tend to ”average” the image by penalizing peaky weight vectors while L1 loss encourages sparseness and is invariant to the ”noisy” inputs. It is practically useful when we tackle the case of bad color hints.

High level L2 pixel-to-pixel loss after VGGnet To encourage the generated image to be visually similar to the real image, we employ a pre-trained Visual Geometry Group net (VGGnet) to extract high-level information of the image. The features are extracted from the highest convolution layer before the linear prediction layer. We compute the L2 distance over this feature map from the real image to the generated image.

Total Variation Loss To enhance the smoothness of the generated image and encourage the neighboring pixels to be similar, we also add a Total Variance loss to the image we generated. We calculate total variation loss as follow, where y is image. In this way, we aim to remove unwanted details while preserving important details.

$$V(y) = \sum_{i,j} \sqrt{|y_{i+1,j} - y_{i,j}|^2 + |y_{i,j+1} - y_{i,j}|^2}$$

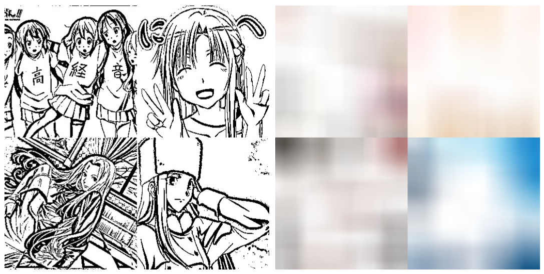
6. Experiment and results

6.1. Baseline: End-to-end U-Net qualitative Results

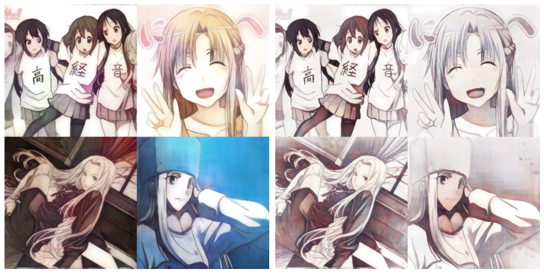
The baseline model is an end-to-end U-Net which optimizes the L2 distance directly. Figure 7 shows the output of U-Net on validation images after training on large volume of training data. Notice that the end-to-end model largely relies on the color hint input. The areas without proper color hint will be pale and not vivid enough. As shown in Figure 7d, when there’s no color hint, the image is hardly colorful.

For quantitative results, please refer to subsection 6.3 for details.

In training phase, we set mini-batch size to 4. There are two reasons why we set such small batch size. First, we do



(a) Line-art (b) color hint



(c) U-Net output(with color hint) (d) U-Net output(No color hint)

Figure 7: End to End Unet results

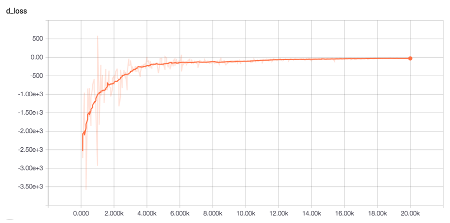
grid searching over different batch size and find out larger batch size does not help significantly on reducing noise and smoothing training process. One of the possible explanations is the painting and colorization style for mange about different authors and cartoon figures are very different from each other. Another reason is due to time limitation and complexity of parameter tuning of WGAN. Noticing increasing batch size leads to significant increase of gradient computation in each step, and we need time to adjust architecture and hyperparameters of model, we finally decide to choose small batch size.

In addition, we use Adam for optimization after we compare the performance with the one of SGD. Moreover, we use grid search to find the most appropriate learning rate, ranging from 1e-1 to 1e-7. Finally, we set learning rate to be 1e-4 as the best after selection.

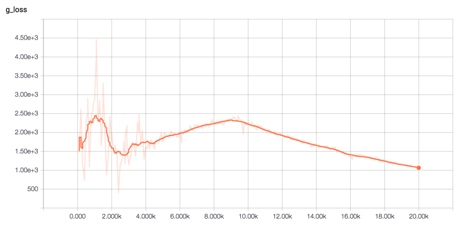
6.2. cGAN method qualitative results

The discriminator and generator loss of training cGAN using WGAN strategies are shown in Fig 8. Notice that the discriminator loss keeps raising which represents the steady decrease in W-distance.

The generated images obtained from cGAN approach can be shown in Fig 1 and Fig 2. Fig 1 shows the result when we input a real line sketch and an arbitrary user-defined color hints. Fig 2, on the other hand, shows the result when we input a sketch image obtained from edge-detection and an arbitrary user-specified color hints. The results show that our model can successfully output a color-



(a) Improved-WGAN discriminator loss curve. Its negative value is approximated W-distance. Lower W-distance (higher discriminator loss) would correspond to higher quality images.



(b) Improved-WGAN generator loss curve.

Figure 8: Loss curve of our cGAN method trained with improved WGAN training strategies

ful manga based on the sketch and color hints.

Given different color hint, WGAN model is capable of generating colorization in different style. For example, the contours of green one on in Figure 2 seems to be sketched and there are sufficient choices of colors in final colorization, which is in correspondence to the feature of manga in oil painting. Comparatively, the red one on figure 2 use simple and single color without outline drawn. It is similar to the inbetweening in Japanese light novel.

Moreover, the generated image is less blur and colorized area is more accurate. For instance, the background would not be falsely colored unless we use really strong color hint (see comparison between the first and second one of figure 1). To be specific, our model learns to color different area with different choice. For example, the figures' face would never be painted colorful. It is because in Japanese manga, cartoon figures' face tend to be flesh color and our recreations clearly convey it by learning the boundary from edge detection and colorization pattern in training set.

6.3. Quantitative Evaluation

In order to measure the similarity between the generated image and the ground truth image, we use four methods to evaluate our result: The W-distance approximated by the negative of discrimination loss, the KullbackLeibler divergence measure, l_2 distance and visual evaluation.

Wasserstein Distance The W-distance can be approximated by the negative value of discriminator loss [11]. Smaller the distance represents higher similarity between the generated images and real images. From Fig 8a we can see that the discriminator's loss steadily go up until convergence.

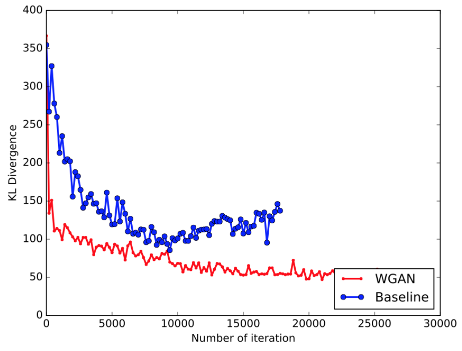


Figure 9: KL distance evaluation during training

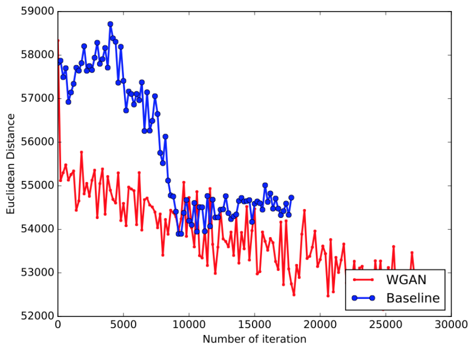


Figure 10: L2 distance evaluation during training

Kullback-Leibler Divergence The Kullback-Leibler distance is a measure of how one probability distribution diverges from a second expected probability distribution. This is a widely used measure in image generation problem. We compute the average KL divergence measure on validation set during the training and the results can be seen in Fig 9. It can be seen for end-to-end method (Baseline), KL divergence decreases at the beginning and then increase. For our cGAN method, the KL distance has a big drop at the beginning and keeps decreasing along the whole training process. The KL distance for cGAN is always smaller than that of baseline method.

l_2 **Distance** We also measure the average l_2 distance on validation set during training and the results are shown in Fig 10. It seems much more noise but the trend of descending for both two methods is clear and also the GAN method outperforms the baseline method.

Visual Evaluation In Fig 11, we list the output of GAN and end to end model. The results displayed in first and second Columns' color hints are completely user-defined

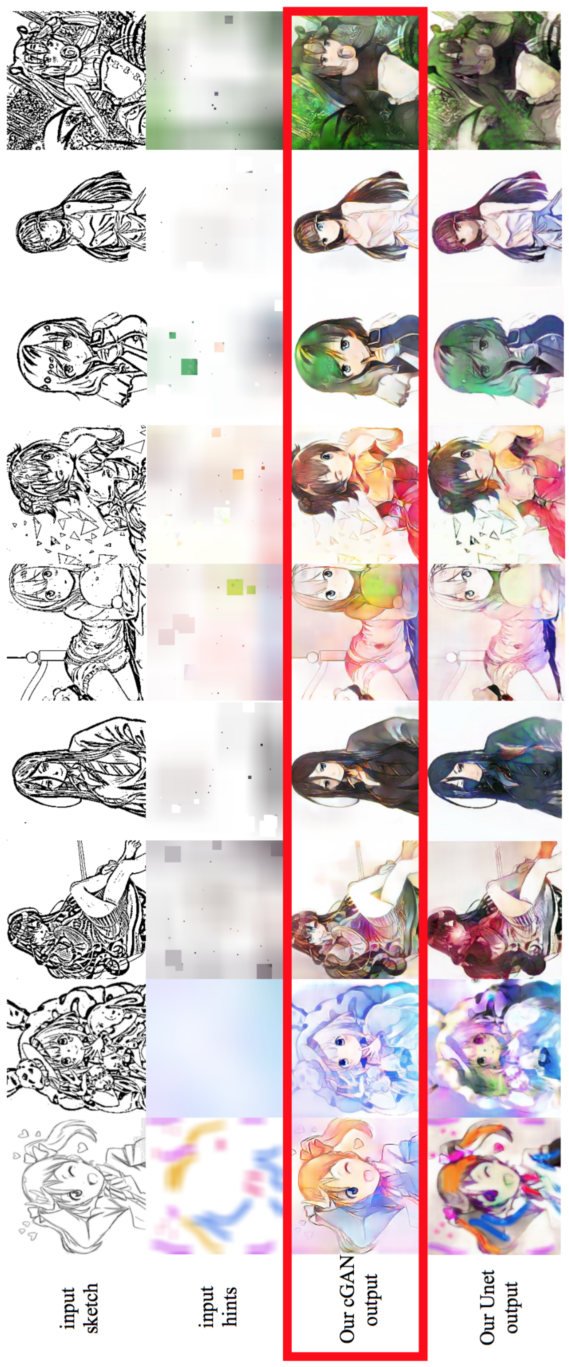


Figure 11: Comparison between cGAN and End to End method

(we use PhotoShop to make the color hint) while the other results' color hints are obtained from original images using the method we described above. It can be shown that for user-defined color hints, the quality of images get from GAN is realistic and much clearer than that of end to end model. Even though the color hint is in the shape of lines and not Gaussian filtered, the color is still evenly distributed in the output images. For color hint obtained from original images, the end to end performances good but the GAN method is still less blurred and the face color is more distinguishable from the background.

6.4. Result comparison and analysis

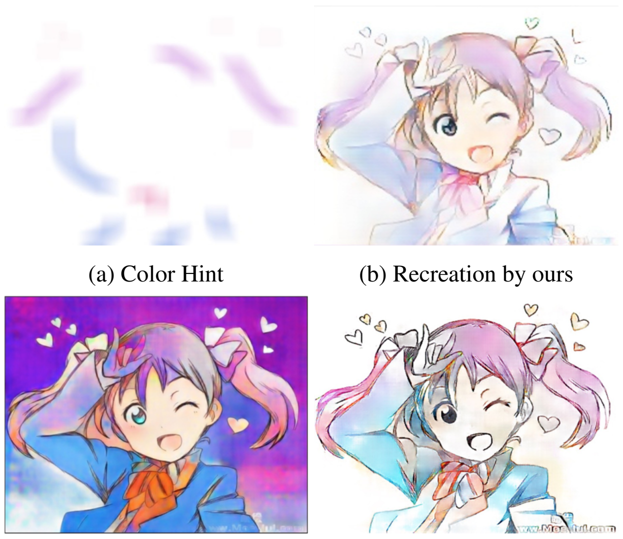
From the results above we can see that our cGAN model can generate much better colored image compared with end-to-end neural network model. Intuitively, the power of end-to-end model greatly depends on its optimization function, which in our case is the L2 distance between real and generated images. Generally speaking, L2 loss is not a good measurement of the similarity between the real image and the generated image. The generator may resort to average out the pixel RGB values which produces a pale generated image. Furthermore, it's very easy for the end-to-end model to overfit on the training set but it fails to generalize when provided with different color hint. We also notice that the L2 loss quickly stop improving on validation set with increased iterations. Given with the wrong incentive to optimize the L2 loss, the model may not learn how to generate images of higher quality.

On the other hand, adversarial models provides a direction for improvement by introducing the min-max game between the generator and discriminator. In this way, the model is able to keep reducing the distance between the real and generated distributions. Furthermore, by following the improved-WGAN method, the discriminator is directly optimizing the expected W-distance between two distributions which corresponds to the visual quality of the images. Lastly, the random z input gives the model resistance to noisy input and thus can generalize better.

6.5. Comparison with state of the art model

In this section, we compare out recreation with the one generated by PaintsChainer, a Japanese automatic colorization website for commercial usage, and DeepColor, another model based on conventional cGAN. We use the same line-art image and color hint as input and results are shown in Figure12.

We can see that our model color different areas more accurately and the colorization boundary is not blur. Comparatively, the one generated by PiaintsChanier blur colors with background and figure itself. We notice that figure's right hand and upper part of background are mistakenly painted purple, which should be the color of hair.



(a) Color Hint (b) Recreation by ours

(c) Recreation by PaintsChainer (d) Recreation by DeepColor

Figure 12: Comparison of recreations of state-of-the-art and ours with same line-art and color hint

The results of cGAN comparatively similar to our result. However, its background is still less clean than ours. Whats more, the face of figure turns to be black and outlines are overwhelming.

However, color saturates better in the recreation by PaintsChainer. We find the almost whole shirt painted blue and bowknot painted red, while part of them is left white in our generation results.

7. Future Work

The line sketch images we used for training comes from line detection results, which may be different from what the real line sketches are. In the future we may explore different ways to do the line detection and make it similar to the line sketch. We may also spend more time to find the data set which contains line sketches and colorful images pairs.

For the loss function design, we add add several terms together, but due to the time limit, we don't have time to test them individually to prove whether they can improve the final performance.

Since our manga colorization results are pretty good and realistic, another really excited thing we can think about is to make an APP based on the trained model we have. We believe it would be a good entertainment for people who love cartoon and even become a great tool for Animation industry.

References

[1] Z. Cheng, Q. Yang, and B. Sheng. Deep colorization. *CoRR*, abs/1605.00075, 2016.

[2] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):295–307, 2016.

[3] K. Frans. Outline colorization through tandem adversarial networks.

[4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[5] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[6] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. *CoRR*, abs/1704.00028, 2017.

[7] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[8] P. Isola, J. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *CoRR*, abs/1611.07004, 2016.

[9] A. Levin, D. Lischinski, and Y. Weiss.

[10] Y. Liu, Z. Qin, Z. Luo, and H. Wang. Auto-painter: Cartoon image generation from sketch by using conditional generative adversarial networks. *arXiv preprint arXiv:1705.01908*, 2017.

[11] A. Martin, C. Soumith, and B. Lon. Wasserstein gan.

[12] M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[13] M. Mirza and S. Osindero. Conditional generative adversarial nets. *CoRR*, abs/1411.1784, 2014.

[14] PaintsChainer.

[15] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434, 2015.

[16] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015.