Automatic Colorization of Anime Sketch Image with Enhanced Residual U-net and Auxiliary Classifier GAN

Chen Pinghao, Huang Xuanmao, Zhang Yikai, Wang Zizhuo, Jiang Changhao

December 30, 2019

0 Introduction

Our Sketch Colorization project is a semi-automatic colorization system for sketchs. Given a sketch image and a reference image as inputs, our system generates a plausible color version of the sketch. To colorize the target sketch image by the color from the reference image, we extract a series of color features from the reference image and feed it to the colorization network to help the process.
Due to the difficulty of labelling the generated image, we chose unsupervised learning and used one of the most famous deep learning technique in Computer Vision, Generative Adversarial Network. It generally has two model, generator and discriminator. Generator generates realistic fake outputs by deceiving discriminator, which makes a decision of whether a given image is real or fake.
In our project, we use a UNet as generator and PatchGAN as discriminator. The reference image is pre-processed into a color histogram and is concatenated with the sketch image as input of the network.

1 Method

Our generator is a UNet with pooling and un-pooling layers. The input of the network is the original sketch image with color histogram of the reference image. Thus, the input size of the generator is 512 x 512 x 15, and the output size is 512 x 512 x 3. And the innermost feature maps are fed into attention gate block.
Our discriminator is 70x70 PatchGAN. Discriminator gets both real image and generated fake image from the generator. After that, it trys to determine which is real and which is not. In the purpose of stable training, we use image pooling, which feeds fake image into discriminator among past 50 fake images.

Denote generator as G, discriminator as D, color histogram extractor as C, sketch image as x, and original colorized image as y. Then training objective becomes

$$\arg\min_G \max_D \mathcal{L}_{GAN}(G, D) + \lambda \mathcal{L}_1(G)$$

$$\mathcal{L}_{GAN} = E_{x,y}\left[\log(1 - D(x, G(x, C(y)))\right] + E_{x,y}[\log(D(x, y)]$$
$$\mathcal{L}_1 = E_{x,y}\left[\|y - G(x, C(y))\|_1\right]$$

where λ is weight hyperparameter

2.1 Color Histogram

For reference images, we crop image into 4 regions vertically. After that, we extract top-4 colors for each region. Using 16 extracted colors, we make 4 RGB images. The i-th image contains top-i color of each cropped region. When the model is trained with color histogram information, we expect that it learns how to merge given regional color information properly that makes natural and realistic colorization.

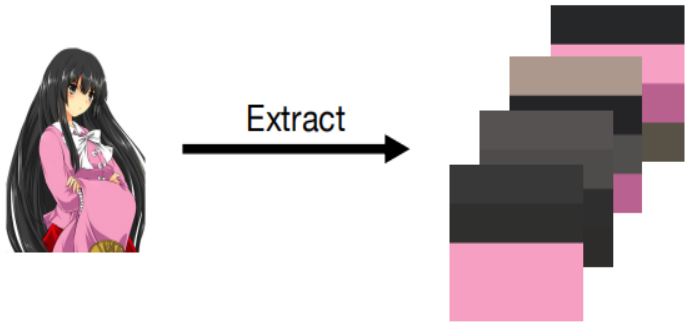

Figure 1: color histogram graph

2.2 Attention Mechanism

In our experiment, we apply attention in skip connection. When concatenating skip connection in decoder, attention gate calculates weight of activation in skip connection and scales skip connection

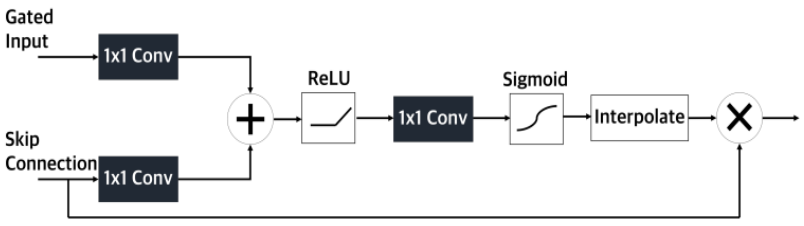

Figure 2: attention mechanism

2.3 Unet

Unet is hard to train .That is to say if the U-net find itself able to handle a problem in low-level layers, the high-level layers will not bother to learn anything.For each layer in U-net's decoder, features can be acquired from higher layers or from skip connected layers.
Thus, for a newly initialized U-net, the output of the mid-level layers can be extremely noisy if we add the vector of 4096 directly to the layer. As mentioned above, the noisy mid-level layer is given up by U-net and these layers cannot receive any gradient as a consequence.

If we attach two additional loss in two decoders to layer is possible hard to train, no matter how noisy the output of a mid-level layer is, the layer will never be given up by the U-net and all layers will get stable gradient in the whole training process.

The training objective can be expressed as:

$$G^* = arg \min_G \max_D L_{GAN}(G, D) + \lambda L_1(G) \tag{1}$$

$$L_{GAN} = E_{x,y}[log(1 - D(x, G(x, C(y))))] + E_{x,y}[log(D(x, y))] \tag{2}$$

$$L_1 = E_{x.y}|||y - G(x, C(y))|||_1 \tag{3}$$

G represents generator, D represents discriminator, C represents color histogram extractor, λ represents weight hyperparameter, x represents sketch image, y represents the original colorized image.
The detailed structure of the Down-Sample Block and Up-Sample Block are as follows:

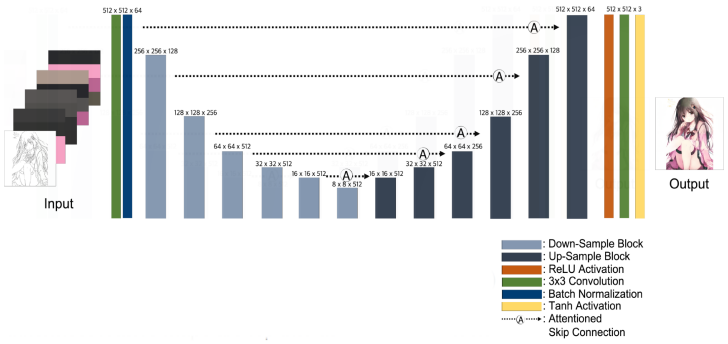

Figure 3: UNet structure

Both blocks are residual block, and Skip-Connections are concatenated in Up-Sample Block.

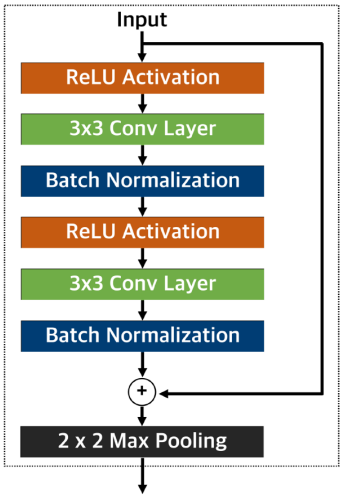

Figure 4: Up-Sample Block

3 Dataset

512 x 512 animation images in Danbooru 2017 dataset. Among them, we select image of single person whose background color is white. Based on these colorized image, we extract sketch using sketchKeras model. Finally, we make sketch to color image paired dataset and corresponding color histogram information in json format. The following is link:
https://www.kaggle.com/ktaebum/anime-sketch-colorization-pair

4 Standard Output

The following result is our reference output. If UNet is well trained by a train set of 14,000 and validate set of 4,000. It should gain a complete model. Using this model, the following line drawing can be colorized.

Figure 5: reference output

Figure 6: reference output

5 Test Result

Below is our hyperparameter for input:

Table 1: Hyperparameter Settings

Hyperparameter	Value
learning rate	0.0002
beta1(optimizer)	0.5
batch size	4
lambda	100
epochs	15

However, we do not have time and resources to fully train the network. We use 20 pictures for train ing and 5 pictures for validation. This is what we have got so far:

Figure 7: test output

6 Discussion and Problems

In our experiment, we discover these potential problems:
(1). Cannot get style for dark image
This is a common problem among feature extraction algorithms. Since a color histogram of a dark picture is hard to obtain, it will require further insight. Therefore, the temporary model cannot solve this problem yet.
(2). Cannot colorize well for noisy background image

If the input painting is incomplete or contain much noise, it will greatly reduce the effectiveness of the model. However, if the input sketch is sharpened before feeding it into the network, the performance is better than expected.

7 Conclusion

Though our test result is not satisfying, we believe that given enough time to train the network, it can fully complete the colorization. Besides, though this method costs a lot of time and resources to gain a model, it can be applied to significantly reduce the workload of human creators.