Line Sketch Colorization with Hint

Bilin Shi
Department of Computer Science & Engineering
The State University of New York at Buffalo
bilinshi@buffalo.edu

Siwei Mai
Department of Computer Science & Engineering
The State University of New York at Buffalo
siweimai@buffalo.edu

Abstract

Sketch colorization is a more difficult task compares to black-white photo colorization since the line sketch do not contain much local texture. Such task is related to a huge amount of demand in animation industry, yet resent research haven't propose any acceptable model for real producing scenario. In this project we implement several models and tricks in the related field and make some improvement and combination to attempt to reach a better performance. Currently the results are not as our expectation, thus further improvement will be achieved in the future.

1 Introduction

Colorization is a popular topic in computer vision, different from other specific topics in computer vision, colorization is a more abstract area which usually contains the creation of human. Traditionally, colorization of computer needs hints from human, and it is only a tool for speeding up the production process, such as the old versions of Adobe Photoshop. However, with the exponential development of deep learning, it is possible for computer to color a black and white picture automatically and creatively. Zhang el. [19] propose a colorization method using deep convolutional neural network, which can automatically paint the black and white photos. Then with the proposed of generative adversarial network (GANs) [6], many solutions for colorization adapt this powerful strategy for training the model. Pixel2Pixel [8] is a generative model which can transform an image from one type(like sketch) to another type(like coloful image).

However, colorization from sketch is a harder topic since the sketch do not have local texture information, the input sketch only contains the edge information, thus such colorization needs more creation. Anime sketch colorization is a subtopic of colorization from sketch, which faces a more specific scenario. Usually, we can figure out much details from a black and white photo and easily imagine how the colorful photo looks like, but for most of us, it's hard to imagine what a anime image looks like only given a sketch, and different painters would paint totally different colorful image from the same sketch. Thus it is a very creative process, and few researchers are working on such topic. There are some interesting application for anime sketch colorization. PaintChainer [1] is the first application only focusing on this specific topic, and the coloring process is fully automatic. Style2paints [18] is a more powerful application for anime sketch colorization, it gives the user more flexible space for adjust the output result via user hints.

Currently, most of those application face the same problems: the watercolor problem and color consistency problem. The watercolor problem is the mix phenomenon in the colored image, it is usually ignored for single image colorization but quite important in anime sketch colorization because the structure of anime image is simple. And the color consistency problem only exist in those contains

the user hint, the problem is that, colored image usually do not have exactly the same color in the specific position. This problem is vital in real producing life since the high-level painter want the output image be exactly what they want to paint, otherwise such coloring process does no help for the whole painting process.
In this project our goal is to estimate the painting process of high-level painters. Specifically, we are focusing on basic colorization of anime sketch and make some improvement on the training strategy, deeper explore for the problems abouvementioned will be done in the next step.
In chapter 2 we will talk about come preprocessing work before training the model, which is also much importance for an acceptable performance. Chapter 3 contains the details of our model and training strategies. Experiments are shown in detail in chapter 4 and the conclusion and future improvement is shown in chapter 5.

2 Data Pre-processing

Data pre-processing is a quite vital part in real engineering project since the data is usually not ideal for specific tasks and also not easy to collect, so does in this task. The data pre-processing contains the collecting and cleaning, sketch extraction and color hint extraction which is specially designed for this task.

2.1 Data Collecting and Cleaning

Though our knowledge, there is no dataset available for anime sketch colorization but only some collected anime images dataset. We go though Danbooru 2018 dataset [4] and found the dataset contains many junk data and low quality, so we decide to collect our training data directly from internet. The website where we collected data is zerochan [2] which only contains anime images, and a multi-thread python crawler is used for collecting. After collecting the raw images, we need to clean it because what we need is colorful images which can extract sketch, not the sketch itself, and small images should also be filtered. Such filtering can be done though the image shape and variance in color domain. Finally we collect 20000 images for training and testing.

2.2 Sketch Extraction

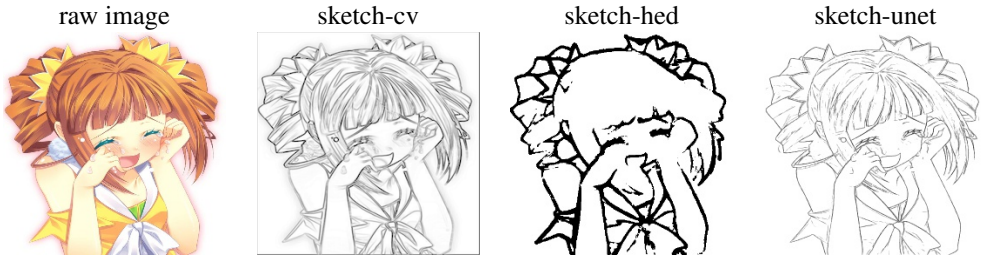
This is the most difficult part for image processing, for this task and also in computer vision. Currently there is no optimal solution for sketch extraction, even human can not extract the sketch from a colorful image whatever the paint-level he or she reached. Expect traditional edge detection algorithm, the deep learning method can also extract the sketch, and there is no guarantee which one can reach a better performance. Thus we try both traditional computer vision method which is a improved canny algorithm [5], and two deep learning methods, one is holistically-nested edge detection (HED) model [17] and a unet-based open-source model called sketchKeras [11]. We expect that the extracted sketch should be as consistent as possible with the artist's painting.

Canny Based Sketch Extraction. The edge detection part uses the improved Canny idea which mainly from the following points: first, we use a filter based on Euclidean distance to filter from the vertical horizontal direction, plus or minus 45 degrees, which replaces the traditional Gaussian filter. This will filter out more noise interference;second, we calculate the amplitude and direction of the image gradient with non-maxima suppression, which is based on Frei-chen algorithm. These make up the deficiency of the traditional gradient amplitude algorithm; finally and most importantly, this detector uses Otsu algorithm to automatically determine the low and high threshold which is manually entered before, along with the connected edges. The detector replace the global thresholds with the optimal thresholds of the regions from Otsu algorithm, which reaches a better anti-noise ability and improves the accuracy of edge detection.

Deep Learning Method for Sketch Extraction. HED is a edge detection algorithm which combines the potential edges over multiple levels and multiple scales. The original HED is training on BSD400, which is quite different from our anime images, and thus it output the unexpected results which we will show later. Another deep learning model we use is called sketchKeras, this

model is open on github proposed by the team of Style2Paints. SketchKeras is a unet-like model designed specially for anime sketch extraction.

The results of each method is shown below. As we can see the output of HED is worse than the others, it miss many details of hairs and draw the edge too thick. The reason is that the original HED use a quite different dataset from this task, and the desired output is the strong shape of the object, which do not contains high frequency details. So this is not a suitable model for this task. Compare the results of canny-based method and unet-based method, there exist double line for one single edge in the cv's output. Since our goal is extract the sketch which is simillar to the artist's painting, not the exactly edges, the double line problem should not exist in the expected sketch. Although for the third method, the model and training details are hidden by the author, it's quite a powerful model for extract anime sketch, just as our expectation, so finally we choose this model for sketch extraction.



2.3 Color Hint

During painting process, an artist usually fist draw the sketch of the expected objects and senses, then paint the color to the sketch. So, to estimate such process, we need to feed some 'hints' to the 'painter'(computer). Simplest way to generate a color hint is generating a highly blurred image using gaussian filter. But such hint it not very suitable for this task since human can not color the image like a gaussian blur, so we proposed two constrains: whiteout and color blocks [16].

Whiteout Nowadays artists usually use some painting tools to speed up the painting process like Adobe Photoshop and SAI. When using these tools, the artists do not need to paint everywhere, instead they can color some key area and the tools will automatically paint the around-related areas like hairs, arms, etc. So to estimate this process, we randomly add whiteout to the blurred image. And there is another explanation for adding such whiteout: since we do not add dropout layers in the generator, these white blocks in the color hint image can be treated as the replacement of dropout, which can force the generator to learn exactly how to paint a sketch but not how to directly transfer the color hint to the colorful image.

Color Block To keep the color consistency in the generated images, we randomly pick up some blocks which has the same color as the original image. By adding these color blocks, we expect that the generator can keep the consistency of the pointed areas.

Now the procedure of generating the color hints is shown in Figure 1, after collecting and cleaning the raw images, we first generate highly blurred images via guassian filter, then we randomly choose some areas to replace them with white, and also some blocks to keep exactly the same color as raw images. Then we get our color hints as the condition in training process.

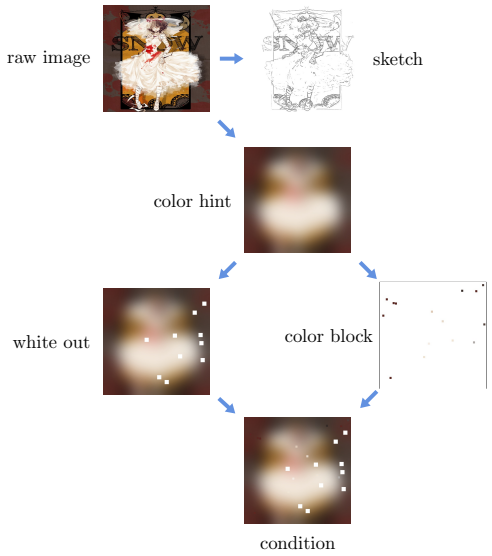


Figure 1: Color Hint Generation

3 Model Design

3.1 Basic Model Structure and Improvement

Our model is designed based on U-net [13], which is originally proposed for biomedical image segmentation and then widely used in colorization. The structure of Unet is similar to an autoencoder but adding the skip connection between the encoder blocks and decoder blocks which have the same scale. As shown in Figure 2, the model contains 5 blocks for encoder with channels from 64 to 1024 exponentially, each block contains 2 convolution layers with 3x3 filter size and 1 maxpooling layer used for reduce the size of feature maps. The decoder block is just the flip of encoder blocks and replace the maxpooling layer with a deconvolution layer with share the same parameter setting with the convolution ones.
The model we use is almost the same as unet, except that we replace the maxpooling layer in encoder part with a convolution layer with the same stride. The intuition of such replacement is that, our input sketch only have very little information like shape of objects and edges. These information are vital to the final colored image, however the maxpooling layer reduce the information extracted from last layer in a quite simple way, and the reduced information can not be recovered. By replace the maxpooling layer with a convolution layer [15], the model can now learn how to reduce the redundant information from the feature maps, and such process now becomes recoverable, which means we do not loss any important information during front-forward process.

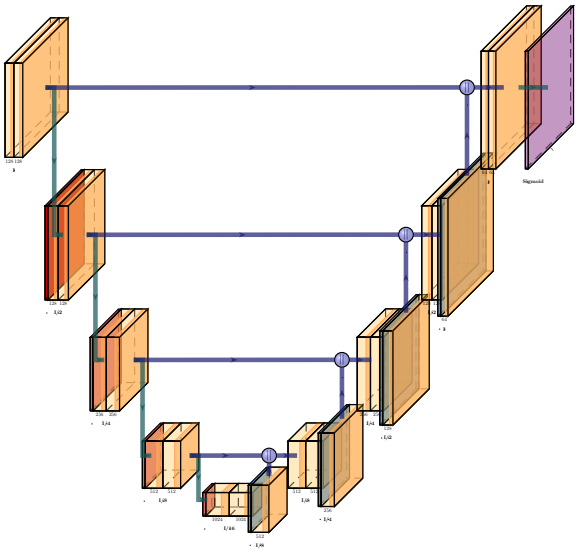


Figure 2: U-Net Structure

Currently we do not pay much attention on the structure of discriminator, so we simply design the discriminator with 4 convolution layers with stride equals 2 and 1 dense layer to make the prediction, which is shown in Figure 3.

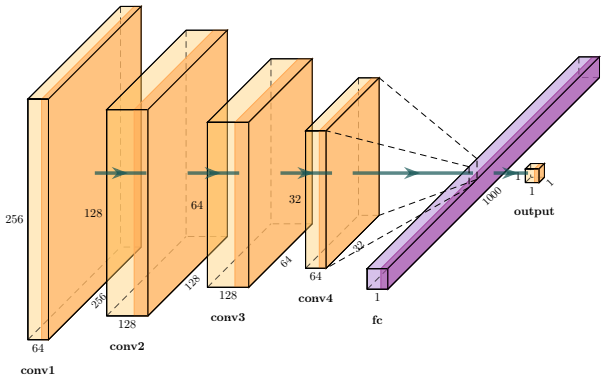


Figure 3: Discriminator

3.2 Adversarial Training with Condition

In generative adversarial networks (GANs) play a minmax game, the discriminator try it best for perfectly distinguish the real image and fake image, and the generator is going to fool the discriminator to output the wrong prediction. Such adversarial process usually can increase the quality of generated images compare with only using the generator.
For our project, it's not totally the same as GANs since we need to map a sketch to a colorful image, while for GANs it generate image directly from a random variable. Thus we apply the idea of conditional adversarial networks(cGANs) [12], where we take the color hint as the condition in this special case, which is shown in Figure 4. The random variable is added into sketch, and when training the discriminator, we only feed the color hint to the discriminator but no sketch. It is reasonable for this task since our goal is to learn the exact mapping but not randomly generate a 'real' image.

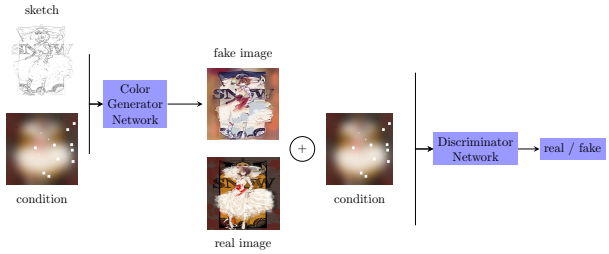


Figure 4: Training with Conditions

3.3 Improvement of Training Strategy

Formally, the target of original GANs is to find optimal parameters' setting $(\hat{\theta}, \hat{\phi})$ that minimax a value function $\mathcal{V}(\mathbb{P}, \mathbb{Q}_\theta, D_\phi)$:

$$\mathcal{V}(\mathbb{P}, \mathbb{Q}_\theta, D_\phi) = \mathbb{E}_{x\sim\mathbb{P}}[\log D_\phi(x)] + \mathbb{E}_{z\sim h(z)}[\log(1 - D_\phi(G_\theta(z)))], \tag{1}$$

$$(\hat{\theta}, \hat{\phi}) = \arg\min_\theta \arg\max_\phi \mathcal{V}(\mathbb{P}, \mathbb{Q}_\theta, D_\phi). \tag{2}$$

Here we can prove that the value function $\mathcal{V}$ is a shifted form of Jensen-Shannon divergence. Although GANs have already achieve much better performance than before, there are still some common problems existed during training GANs like mode collapse problems and stop convergence of generator. Arjovsky et al. [3] prove that Jensen-Shannon divergence along with KullbackLeibler divergence and Total-Variance divergence are facing the same problem, that is, when two distribution are completely separable in lower dimensional manifold, those divergence will always backpropagate 0 gradients to generator, thus the generator cannot improve at all. Thus the author propose the Wasserstein GANs which use Earth-Mover distance to measure the difference of two distribution, and with some transformation and restriction, the target of the model becomes to

$$\min_G \max_D \mathbf{E}_{x\sim P_{real}}[\log D(x|c)] + \mathbf{E}_{x\sim G(z|c)}[\log 1 - D(x|c)]. \tag{3}$$

Here the discriminator D should be a K-Lipschitz function, and the author achieve this contrain through clipping the weights of parameters of discriminator. However, such constrain is hard to achieve simply through clipping, and Gulrajani et al. [7] point out that such clipping would cause the mapping of discriminator becomes too simple. To address this problem, they add a penalty to constrain the discriminator to be an 1-Lipschitz function. The new target is:

$$\min_G \max_D \mathbf{E}_{x\sim P_{real}}[D(x)] - \mathbf{E}_{x\sim G(z|c)}[D(x|c)] + \lambda \mathbf{E}_{\tilde{x}\sim p_{\hat{x}}}[(||\nabla_{\hat{x}} D(\hat{x})||_2 - 1)^2] \tag{4}$$

where $\hat{x}$ is the weighted average of real image x and fake image $\tilde{x}$ with a random scalar ϵ.

3.4 Extension of Generator Loss

Trough experience, it's really hard to solve a colorful-image-related task only through adversarial training. In our viewpoint, the GANs is a improving part of the whole task, the generator should first learn some mapping function from one type of image to another. If we train the model though GANs from the beginning, there is no guarantee the discriminator will distinguish two images in reasonable way and the way generator fool the discriminator. And also, such training process is quite long.
To handle this problem, it is very common to add other form of loss into the generator loss, here we introduce two of them.

L1 Loss of Pixel Level In this task we use L1 loss instead of L2 loss because L2 loss would lead the generator to generate blurry images [8] which is not our expectation. Instead, L1 loss can usually sparse the parameters, then the more important information is remained, which can generate

a clearer colorful image.

L2 Loss of High Level Features We also adapt a L2 loss of high level features generated by VGG19 [14], which we assume contains high level information hidden in the raw images. The form of L2 loss is:

$$\mathcal{L}_2 = \sum ||\Phi_{VGG}(x) - \Phi_{VGG}(G(z|c))||_2. \tag{5}$$

4 Experiment

4.1 Model Pre-Training

As mentioned in section 3.4, directly train the model using GAN loss is inefficient and time consuming, with the inspire of two-stage training idea of Style2Paints [18], we first do pre-training only for generator using the combination of L1 and L2 loss. The total loss is:

$$\mathcal{L} = \mathbb{E}[\lambda_1||x - G(z|c)||_1 + \lambda_2||\Phi_{VGG}(x) - \Phi_{VGG}(G(z|c))||_2] \tag{6}$$

where λ_1 and λ_2 are the ratio of two kind of losses. The training results are shown in Figure 5 and Figure 6. The figures show that the curves of validation do not have a tendency to increase, which guarantees that our pre-trained model haven't overfitted yet, even not converge after 100k batches. It is suitable to feed such pre-trained model into next steps.

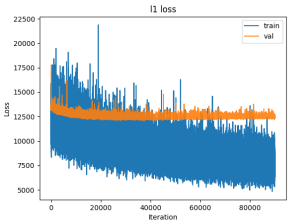


Figure 5: L1 Loss

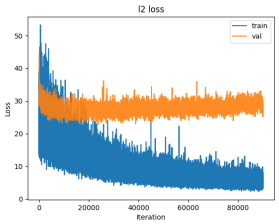


Figure 6: L2 Loss

4.2 Results of Training with GANs

Before go into WGAN-GP training, we first use original GANs to train the model given the pre-trained parameters. The loss is defined as the same as Equation 1. The loss curves are shown in Figure]7 and Figure 8. The training process is quite unstable and the losses for validation data is continuously increasing, which indicates that the generated results would be worse, and so it is as shown in section 4.4.

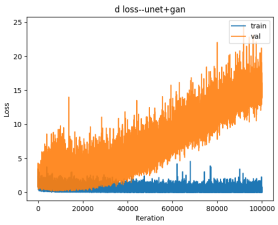


Figure 7: D Loss of Training with GANs

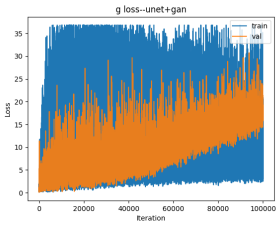


Figure 8: G Loss of Training with GANs

4.3 Results of Training with WGAN-GP

For training with WGAN-GP, we extend the generator loss with L1 loss to constrain the model attempting to converge from the beginning. The results are shown in Figure 9 and Figure 9. These figures shows that the training process is still unstable, but the validation curve is very close to

training curve. Such phenomenon indicates that the generated results are not getting worse, probably with the increase of iteration times, the model would be more likely to converge.

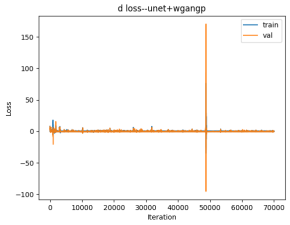


Figure 9: D Loss of Training with WGAN-GP

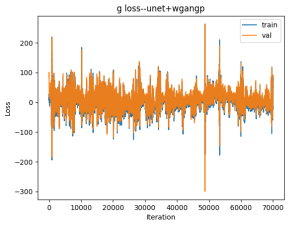
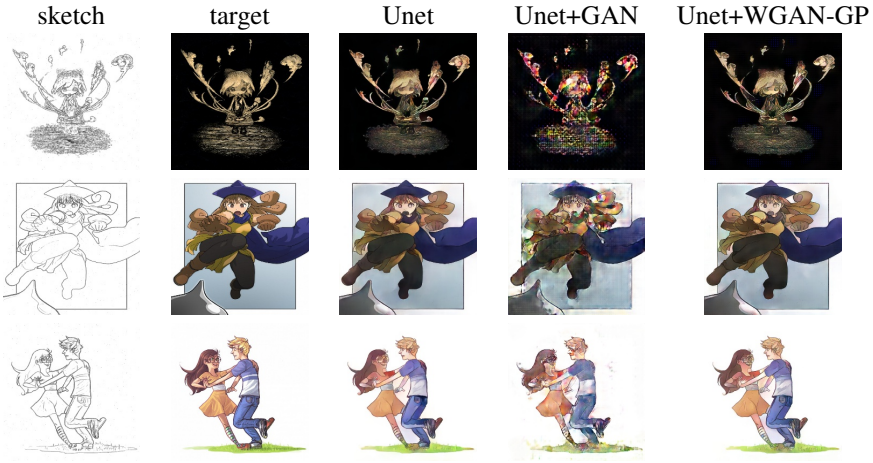


Figure 10: G Loss of Training with WGAN-GP

4.4 Comparison among each step

The generated images of three abovementioned models are shown below. We can see that the model trained with GANs reach the worst results just the same as our analysis. And compare with the pre-trained model, the model trained with WGAN-GP generate more details in the colorful images. These results prove that our improvement of using pre-trained model for initialization and change of adversarial training strategy have positive effect for anime sketch colorization, and it's also a good start for our future exploration.



5 Conclusion and Future Improvement

In this project we work on anime sketch colorization, which is a hard topic in colorization field and few previous work is focusing on this specific topic. Our work contains data collecting and datat cleaning, sketch extraction, color hint extraction, refinement of U-net model and attempts for different adversarial training strategies. Our results are not as good as our expectation but enough to prove the positive effect of our improvement.

There are still several issues need deeper exploration in the future: **sketch extraction** is still a challenge since current method can not extract correct sketch when the edges are not very obvious. Through our knowledge we did not find any advanced research on extracting sketch from colorful anime images. There are several sketch extraction algorithm whose inputs are black and white manga images [10] or blurred sketch images [14] which may be helpful for extracting sketch from colorful images; **the form of prediction** is the colorful image itself in our project, but it can be simpler like residual images [9], and simpler mapping can reduce the training time and increase the quality of generated images; **color consistency problem** is not solved in this project, and there is no existed solution for this problem, though our knowledge. This is a more challenging topic and need more exploration on how to define a reasonable and feasible penalty for color inconsistency. These challenges will be our next exploration for anime sketch colorization task.

References

[1] PaintsChainer. https://paintschainer.preferred.tech/index_en.html.

[2] Zerochan. https://www.zerochan.net.

[3] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[4] Gwern Branwen. Danbooru2018: A large-scale crowdsourced and tagged anime illustration dataset. https://www.gwern.net/Danbooru2018#kaggle, 2015.

[5] Yingke Feng, Jinmin Zhang, and Siming Wang. A new edge detection algorithm based on canny idea. In *AIP Conference Proceedings*, volume 1890, page 040011. AIP Publishing, 2017.

[6] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[7] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5767–5777, 2017.

[8] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[9] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1646–1654, 2016.

[10] Chengze Li, Xueting Liu, and Tien-Tsin Wong. Deep extraction of manga structural lines. *ACM Transactions on Graphics (TOG)*, 36(4):117, 2017.

[11] lllyasviel. sketchkeras. https://github.com/lllyasviel/sketchKeras, 2017.

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

[13] Olaf Ronneberger, Philipp Fischer, and Thomas 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.

[14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[15] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

[16] Honghao Wei, Yiwei Zhao, and Junjie Ke. Automatic manga colorization with hint.

[17] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *Proceedings of the IEEE international conference on computer vision*, pages 1395–1403, 2015.

[18] Lvmin Zhang, Chengze Li, Tien-Tsin Wong, Yi Ji, and Chunping Liu. Two-stage sketch colorization. In *SIGGRAPH Asia 2018 Technical Papers*, page 261. ACM, 2018.

[19] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *European conference on computer vision*, pages 649–666. Springer, 2016.