ZING! POW! – COMIC COLORIZATION WITH CGAN

A Thesis

Presented to the

Faculty of

California State Polytechnic University, Pomona

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science

In

Computer Science

By

Yiyang Yan

2018

SIGNATURE PAGE

THESIS:	ZING! POW! – COMIC COLORIZATION WITH CGAN
AUTHOR:	Yiyang Yan
DATE SUBMITTED:	Fall 2018
	Department of Computer Science

Dr. Tingting Chen
Thesis Committee Chair
Computer Science

Dr. Hao Ji
Computer Science

Dr. Yu Sun
Computer Science

ACKNOWLEDGEMENTS

I'd like to acknowledge all the faculty and students whom I've encountered at Cal Poly Pomona. Thank you for creating such a pleasant campus environment. I particularly like to thank my Thesis Committee Chair and advisor, Dr. Tingting Chen for guiding me throughout the thesis process.

I'd also like to thank the MENTORES Scholarship Program for sponsoring me at CPP.

I'd also like to thank my late grandmother for putting in the idea about going to graduate school in my mind – inception!

ABSTRACT

The industry standard for East Asian serialized comic artists and animators is to work extremely long hours and expect very little pay. While this custom seems archaic and unfair towards these hard-working artists, the nature of their line of work is understandably demanding. Serialized comics and animations must be finished quickly to meet weekly deadlines. The quality of these artworks must reach a certain standard to maintain consumer interest. Finally, these artworks must be done cheaply to avoid financial issues. As a result, these comic and animation artists are overworked and underpaid. I propose to ameliorate the nature of the work and to relieve these artists from some of the burden from the nature of their work. Using Conditional Generative Adversarial Networks, I have improved existing methods to automate coloring on black-and-white comic pages, as well as investigated weaknesses in current automated colorization methods available. By automating the coloring portion of an artist's work, I aim to relieve at least this part of an artist's busy schedule.

TABLE OF CONTENTS

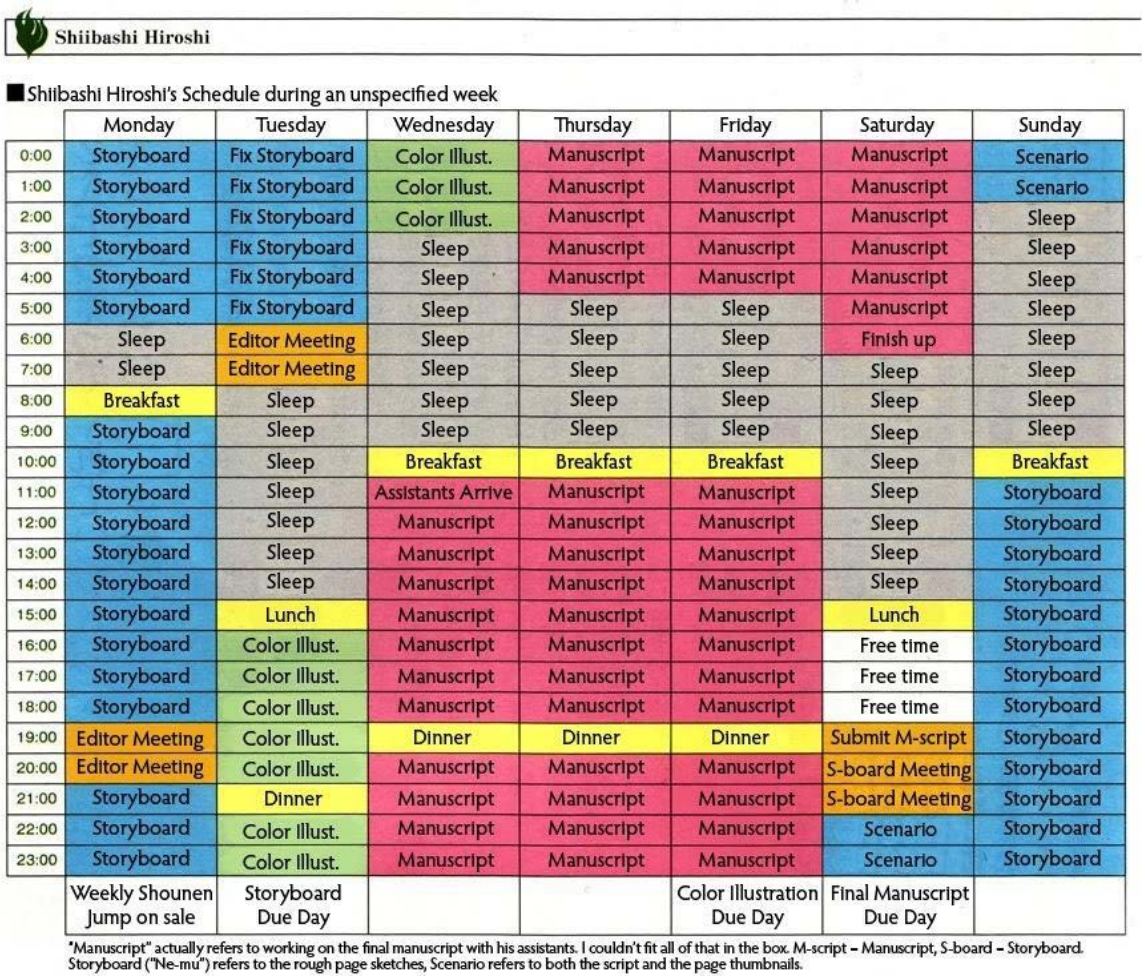
LIST OF FIGURES

CHAPTER 1 – INTRODUCTION

A Cruel Industry

The starving artist - a common trope that describes the well-accepted state-of-being of the common artist. Mainstream and media depict a typical artist as talented but struggling to maintain his or her well-being through talent alone. Independent artists may have trouble finding a steady stream of paying commissions. Industry artists may be underpaid and underappreciated for their work.

Serialized comics artist and cartoon animators fall into the latter category. Particularly in the East Asian regions, comics artists and animators are severely overworked and underpaid. The below figure charts the renown Japanese comic artist, or mangaka, Hiroshi Shiibashi's typical week's schedule.[3] Of particular note, Shiibashi only has three hours out of a total 168 hours of free time every week. The rest of his time is spent split between drawing for his weekly serialized Japanese comic, or manga, meeting with his editors, or eating and sleeping. In addition, Shiibashi only sleeps two hours on Mondays! Such a busy schedule comes from the nature of Shiibashi's work. This mangaka publishes his manga in a weekly magazine. My focus is with Shiibashi's coloring schedule. According to the chart, Shiibashi spends ten hours a week coloring. I aim to reduce that time and free up Shiibashi and other artists' schedules.



Shiibashi Hiroshi

■Shiibashi Hiroshi's Schedule during an unspecified week

	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
0:00	Storyboard	Fix Storyboard	Color Illust.	Manuscript	Manuscript	Manuscript	Scenario
1:00	Storyboard	Fix Storyboard	Color Illust.	Manuscript	Manuscript	Manuscript	Scenario
2:00	Storyboard	Fix Storyboard	Color Illust.	Manuscript	Manuscript	Manuscript	Sleep
3:00	Storyboard	Fix Storyboard	Sleep	Manuscript	Manuscript	Manuscript	Sleep
4:00	Storyboard	Fix Storyboard	Sleep	Manuscript	Manuscript	Manuscript	Sleep
5:00	Storyboard	Fix Storyboard	Sleep	Sleep	Sleep	Manuscript	Sleep
6:00	Sleep	Editor Meeting	Sleep	Sleep	Sleep	Finish up	Sleep
7:00	Sleep	Editor Meeting	Sleep	Sleep	Sleep	Sleep	Sleep
8:00	Breakfast	Sleep	Sleep	Sleep	Sleep	Sleep	Sleep
9:00	Storyboard	Sleep	Sleep	Sleep	Sleep	Sleep	Sleep
10:00	Storyboard	Sleep	Breakfast	Breakfast	Breakfast	Sleep	Breakfast
11:00	Storyboard	Sleep	Assistants Arrive	Manuscript	Manuscript	Sleep	Storyboard
12:00	Storyboard	Sleep	Manuscript	Manuscript	Manuscript	Sleep	Storyboard
13:00	Storyboard	Sleep	Manuscript	Manuscript	Manuscript	Sleep	Storyboard
14:00	Storyboard	Sleep	Manuscript	Manuscript	Manuscript	Sleep	Storyboard
15:00	Storyboard	Lunch	Manuscript	Manuscript	Manuscript	Lunch	Storyboard
16:00	Storyboard	Color Illust.	Manuscript	Manuscript	Manuscript	Free time	Storyboard
17:00	Storyboard	Color Illust.	Manuscript	Manuscript	Manuscript	Free time	Storyboard
18:00	Storyboard	Color Illust.	Manuscript	Manuscript	Manuscript	Free time	Storyboard
19:00	Editor Meeting	Color Illust.	Dinner	Dinner	Dinner	Submit M-script	Storyboard
20:00	Editor Meeting	Color Illust.	Manuscript	Manuscript	Manuscript	S-board Meeting	Storyboard
21:00	Storyboard	Dinner	Manuscript	Manuscript	Manuscript	S-board Meeting	Storyboard
22:00	Storyboard	Color Illust.	Manuscript	Manuscript	Manuscript	Scenario	Storyboard
23:00	Storyboard	Color Illust.	Manuscript	Manuscript	Manuscript	Scenario	Storyboard
	Weekly Shounen Jump on sale	Storyboard Due Day			Color Illustration Due Day	Final Manuscript Due Day	

"Manuscript" actually refers to working on the final manuscript with his assistants. I couldn't fit all of that in the box. M-script = Manuscript, S-board = Storyboard. Storyboard ("Ne-mu") refers to the rough page sketches, Scenario refers to both the script and the page thumbnails.

Figure 1. Renown manga artist Shiibashi Hiroshi's busy work schedule includes only 3 hours of free time every week![6]

Cartoon animators face similar tough working conditions. For an example in Japan, the industry standard pay rate for new talent artists is only ¥200 per drawing.[8] They create twenty illustrations per day. This totals a mere ¥100,000 or US $911 per month. The Japan Animation Creators Association (JAniCA) reports that, in 2015, animators average eleven working hours per day and that they have just four days off per month.[8]

Animators facing long working hours but low wages have become commonplace globally. Like the serialized comics artists, the demanding nature of these artists' works creates these slave-like working conditions. Serialized comics and animations must be finished quickly to meet weekly deadlines. The quality of these artworks must reach a certain standard to maintain consumer interest. Finally, these artworks must be done cheaply to avoid financial issues. Comics and animations are considered cheap forms of entertainment. Comic publishers do not pay their artists very much. The high-profile allure of being published in a popular comic magazine has allowed comic publishers to negotiate a low salary for their comic artists. Animations are comparatively low budget for studios to produce. By custom, animation studios do not receive much of the profits from a cartoon. Thus, animators do not get paid much either.

In the face of such laborious long work hours for these artists demanded by their respective industry, I aim to find a way to automate the coloring portion within these artist's work schedule through artificial intelligence machine learning. Through my research, I have found that a Conditional Generative Adversarial Network (cGAN) best suits my aim. In this paper, I explore the possibility of using cGANs to generate a colored

version of my uncolored input drawings. Because conditional GANs learn from a conditional generative model, this makes cGANs suitable for image-coloring translation tasks, where I condition on an input image and generate a corresponding output image.

Conditional GANs has commonly been used for machine learning image processing. GANs have been vigorously studied in the last two years and many of the techniques we explore in this paper have been previously proposed. Nonetheless, earlier papers have focused on specific applications, and it has remained unclear how effective image-conditional GANs can be as a general-purpose solution for image-toimage translation. My code is available at https://github.com/tried42long/Manga_Colorization.

CHAPTER 2 – LITERATURE SURVEY

Like Minds

***Related Work**: cGAN-based Manga Colorization Using a Single Training Image*

Paulina Hensman has proposed a comic colorization method based on conditional Generative Adversarial Networks (cGAN).[3] Unlike previous cGAN approaches that use many hundreds or thousands of training images, her method requires only a single colorized reference image for training, avoiding the need of a large dataset. Because colorizing manga using cGANs can produce blurry results with artifacts, and the resolution is limited, she has therefore also proposed a method of segmentation and color-correction to mitigate these issues. The results are sharp, clear, and in high resolution, and stay true to the character's original color scheme.

As further expanded upon later in Chapter 5, this thesis's work is based on Hensman's colorization method as a starting point. I have found weaknesses in her method and strive to improve upon them.

Figure 2. Hensman's thesis: cGAN-based Manga Colorization Using a Single Training Image[3]

***Related Work**: pix2pix* - Image-to-image translation with conditional adversarial nets

Isola et al created pix2pix, which features the automated coloring process baseline for all other researches. Pix2pix features a conditional adversarial network as a general-purpose solution to image-to-image translation problems.[5] However, while pix2pix has many functions, I am concerned only with its colorization capabilities. A pix2pix network could be trained on a training set of such corresponding pairs to learn how to make full-color from black & white images. Once a pix2pix network has been trained on such a dataset, it could then be used to color arbitrary black & white images.

Related Work: Consistent Comic Colorization with Pixel-wise Background Classification

Sungmin Kang's research comes in that fills a weakness in Hensman's research. In Hensman, colors "bled" from characters to background. Kang also developed a model that automates colorization in comics. However, Kang used a more traditional method of training using a large amount of comics. But notably, Kang was able to more accurately colorize his comics by introducing a step that separates background from foreground. By using Pix2Pix to distinguish between foreground and background within a comic page, Kang can more accurately colorize his results.[9]

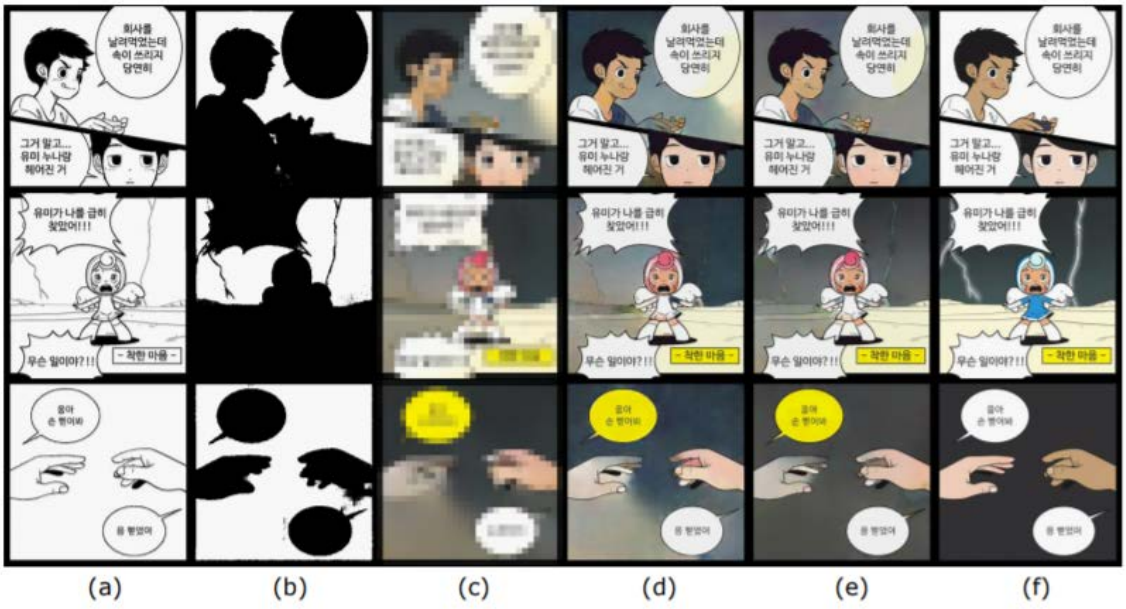


Figure 3. Kang's Consistent Comic Colorization, where background detection distinguishes foreground comic characters from the comic background[9]

CHAPTER 3 – RESEARCH GOALS

A Target

My literature survey arrives at my problem statement. Currently, the comic artist and animator industries severely overwork their staff. These artists draw for long hours at low pay. To relieve some of the burden and to improve production quality, I aim to automate the coloring part of their work. Hensman's comic colorization uses a conditional Generative Adversarial Network to train an automatic colorization process based on only single images. For my research, I will be improving upon Hensman's overall single image training automated colorization process.

CHAPTER 4 – METHODOLOGY

How I Did It

A diagram of my design model is shown below. My Conditional Generative Adversarial Network (cGAN) takes advantage of Keras, a high-level neural network API written in Python.[10] Keras is built on top of TensorFlow, an open source machine learning framework. In the diagram, my target images that are to be colored and my test images which contain the color references are used as input into the cGAN model. Within the model, Python and Keras break down the images into their pixel value data. The cGAN takes these pixel values and applies a series of convolutions and deconvolutions (more on that later)

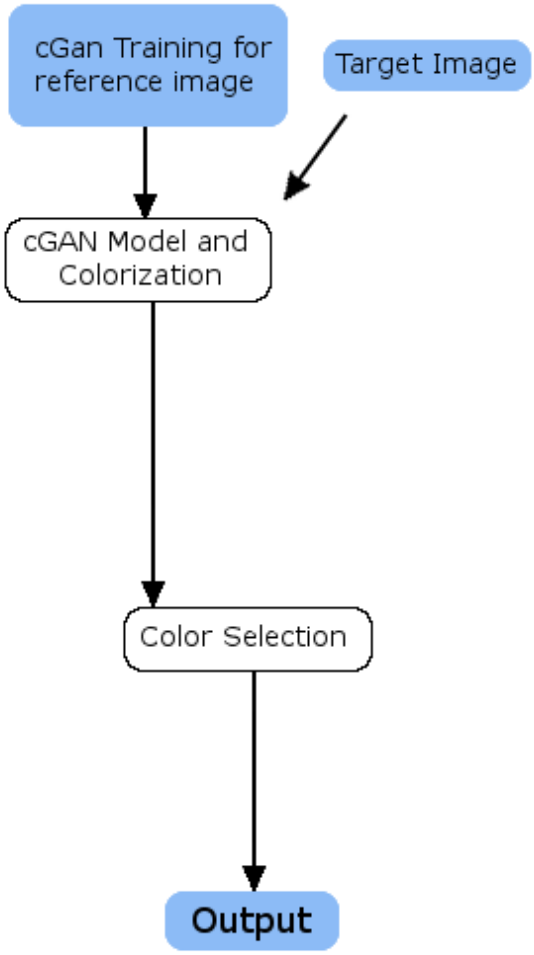


Figure 4. My design diagram

My Conditional Generative Adversarial Network (cGAN) is a composite network made up of two networks along with a conditional acceptance requirement. GAN features a generator, which is a deep neural network that generates my images, and a discriminator, which distinguishes estimated correct colors from decidedly false colors that the generator produces. Consider the generator and discriminator as having the roles of a counterfeiter and identifier. Colors are labeled has accurate or false by the

discriminator to further train the bank in identifying the fake color schemes. The two neural nets compete with each other.[7] The discriminator strives to distinguish accurate from false. Meanwhile, the generator continues to challenge the discriminator with convincely false colors. The discriminator is a classification problem where one set of data comes from my actual color inputs while the other set of data comes from the generator. This network relationship's cost function J^D that determines true colors from false can be model by this function:

$$J^D(\theta^D, \theta^G) = -\frac{1}{2}\mathbb{E}_{x\sim p_{data}} logD(x) - \frac{1}{2}\mathbb{E}_z log(1 - D(G(z)))$$

GAN is a zero-sum game. While the generator maximizes the log-probability of determining accurate and false colors, the generator minimizes this probability. This relationship can be modeled as such:

$$\theta^{(G)*} = \arg\min_{\theta^{(G)}} \max_{\theta^{(D)}} V(\theta^{(D)}, \theta^{(G)})$$

Conditional GAN differs from GAN in that the generator learns to generator the fake samples with a specific condition rather than a generic noise distribution sample. This extra condition, vector y, is fed into both the discriminator D(X,y) and generator G(z,y). The function of cGAN can be described as:

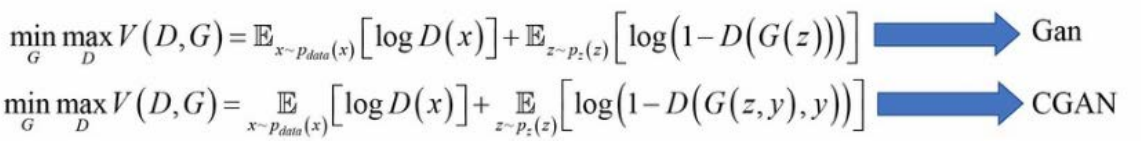


Figure 5. Minmax function differences between GAN and cGAN[7]

A glance into my code reveals how I have designed my cGAN in Keras:

```
# function to train the cGAN model
def train(gen, disc, cGAN, gray, rgb, gray_val, rgb_val, batch):
    samples = len(rgb)
    gen_image = gen.predict(gray, batch_size=16)
    gen_image_val = gen.predict(gray_val, batch_size=8)
    inputs = np.concatenate([gray, gray])
    outputs = np.concatenate([rgb, gen_image])
    y = np.concatenate([np.ones((samples, 1)), np.zeros((samples, 1))])
    disc.fit([inputs, outputs], y, epochs=1, batch_size=4)
    disc.trainable = False
    cGAN.fit(gray, [np.ones((samples, 1)), rgb], epochs=1, batch_size=batch, \
        validation_data=[gray_val,[np.ones((val_samples,1)),rgb_val]])
    disc.trainable = True
```

Figure 6. cGAN training function using Keras

Much of this function is like Hensman's cGAN used in her own research. I have found little need to change it because model initialization is the same. Of particular note is Keras's fit() function, where the cGAN specification is designated.

- **gray** is the numpy array of training data. It is an array containing two-dimensional matrices of my color input images. These inputs images are used to train the model so that my target images may be colored in.
- **np.ones** contains the array of target data.

- **epoches** is set to 1. However, there is a later loop that runs the training multiple times, so my cGAN does go through multiple iterations.
- **batch_size** is the number of samples per gradient update. It is set to 1.
- **validation_data** is the tuple on which to evaluate the loss.[10]

The following is a sample of a set of convolutions within the cGAN model:

```
conv5_512 = Conv2D(256,kernel_size=(3,3),padding='same',activation='elu',kernel_regularizer=l2(0.001))(conv4_256)
conv5_512 = Conv2D(256,kernel_size=(3,3),padding='same',activation='elu',kernel_regularizer=l2(0.001))(conv5_512)
conv5_512 = MaxPooling2D(pool_size=(2,2),padding="same")(conv5_512)
conv5_512 = BatchNormalization()(conv5_512)
```

Figure 7. A line of code for cGAN convolution

Conv2D is a two-dimensional convolution function from the Keras API.

- The first argument, **256**, is a tensor variable that changes based on the convolution sequence.
- **Kernel_size**, the number of pixels to take in at a time, is fixed at 3x3.
- **Padding** is also fixed to same, so that extra pixels outside of the image border is taken if necessary
- My **activation** function that I have selected is Exponential Linear Unit (ELU). The possible negative values push the mean of the activations closer to zero. This causes faster learning and convergence.
- **kernel_regularizer** allows penalties on layer parameters or layer activity during optimization. The default is 12(0.01), so I chose to stick with the default.[10]

CHAPTER 5 – PROJECT TEST AND FINDINGS

Colorful Contributions

My comic colorization process strives to improve upon Hensman's colorization method. In my investigation, I use Hensman's neural network as the control while I make improvements. The following are findings and suggestions I have found to improve upon the method.

Overall, I find Hensman's colorization process lacking for my needs of coloring comic pages. However, by altering input, weights, and convolutions, the coloring produced becomes more accurate.

Control

First, I start with my control: what Hensman's colorization can actually do. Below is produced from Hensman's own input images and target images. With observation from the naked eye, it is clear that Hensman's cGAN produces coloring that is reasonably logically correct for this set of target images, at least.

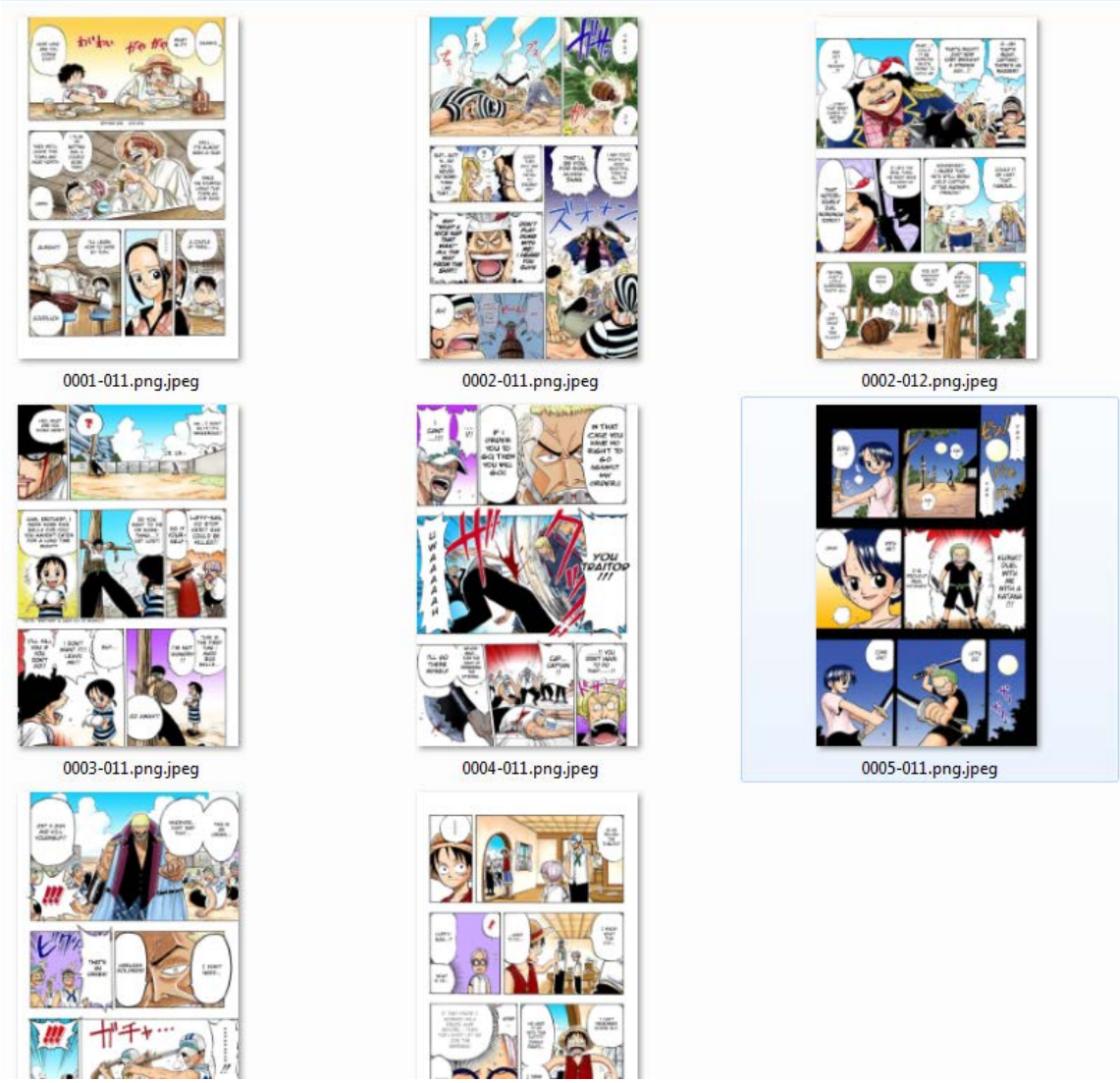


Figure 8. Control set colorization

Figure 9. Control set results

I have found similar success when using my own input images and target images that are like Hensman's as shown below. Figure 10 displays a portion of my training set.

These are the comic pages that become the training data for my colorization method. It is intended that the schemes within these pages are used as the basis for coloring my target images. Training and coloring are based on the hue of different parts of the images. For an example, the hue of the sky within my training sets is marked. Hue of areas in my target images are expected to be sky as well and are colored in the sky's color. This method of coloring is useful for comics because my colorization process can figure out a reoccurring character and color that character the same way repeatedly across new pages.

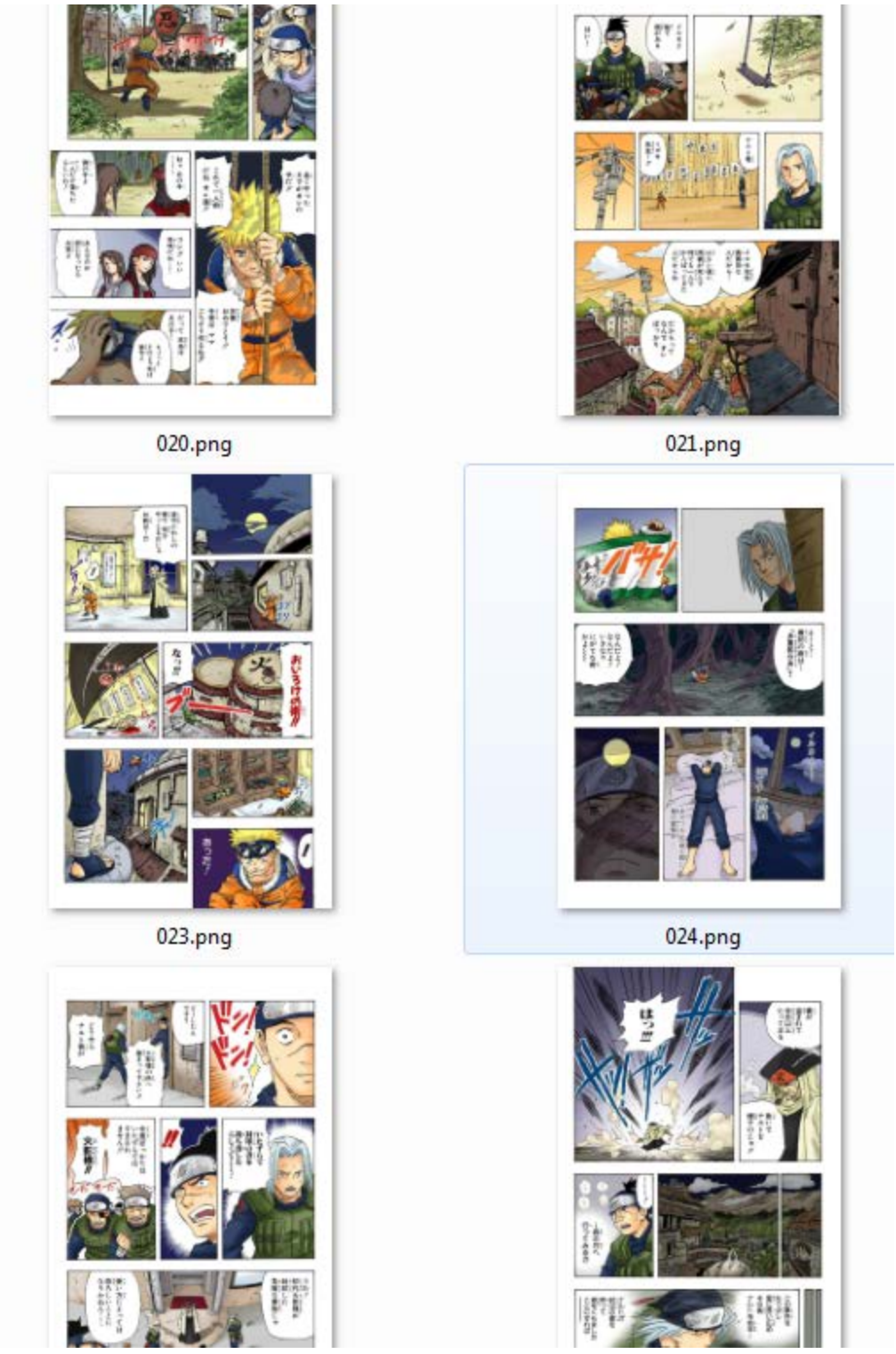


Figure 10. my generated control set variation

Below is Figure 11. Figure 11 displays a portion of the target images – images that I want to color in. In this case, I have selected training images and target images from different chapters within the same comic. This ensures my process's automated coloring colors in the same hues consistently and logically according to the already established color schemes of my training set. For an example, the target images' skies will be colored according the skies of the training images. The characters of the target images' characters will be colored according to the characters in the training images.

Figure 11. my target control set

Figure 12 displays the results of my colorization process. While the coloring is not perfectly correct, the automated coloring is present and makes logical sense. This coloring is considered a success.

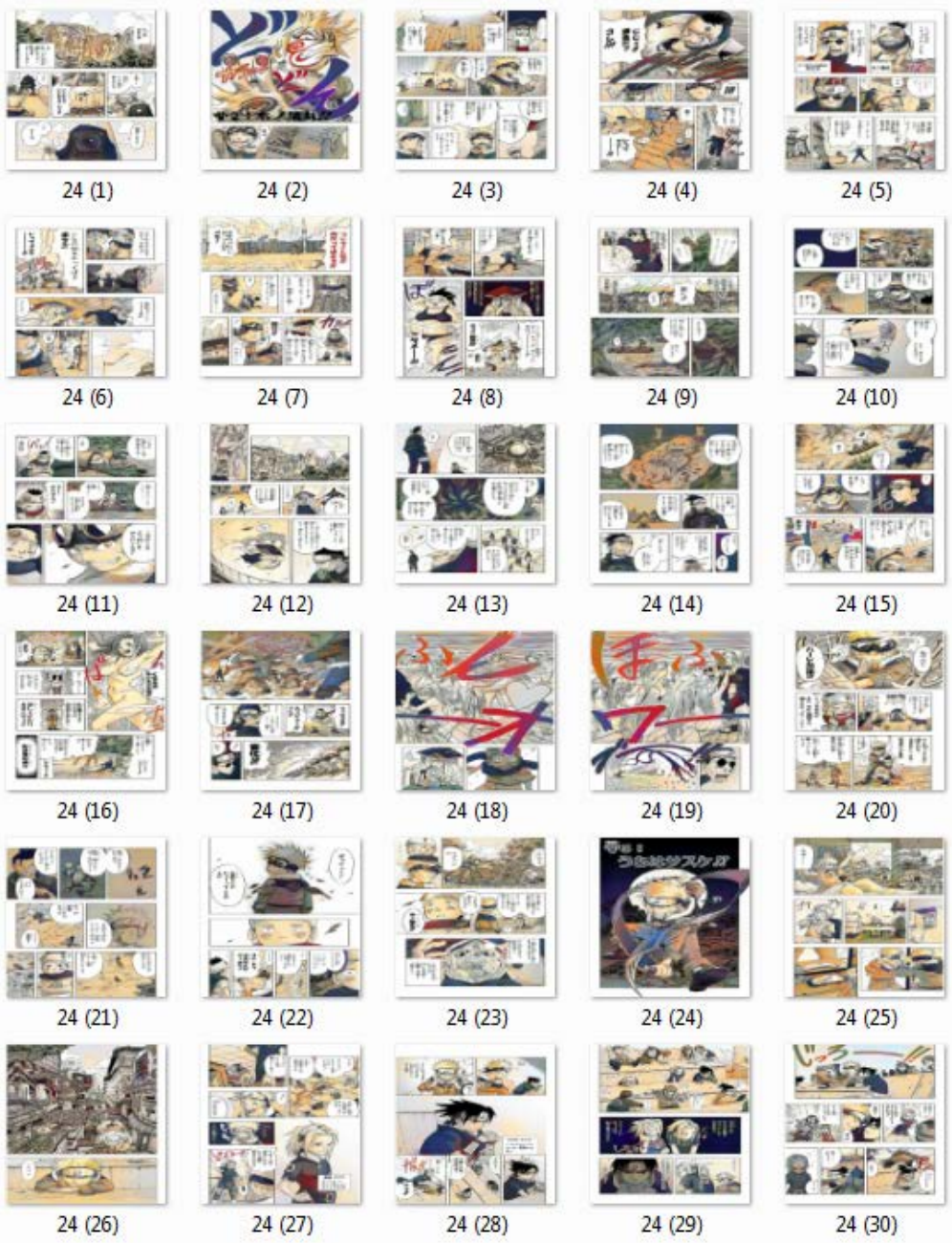


Figure 12. my generated automated colorized comics. A success!

Figure 13. Close-up of a generated automated colorized page

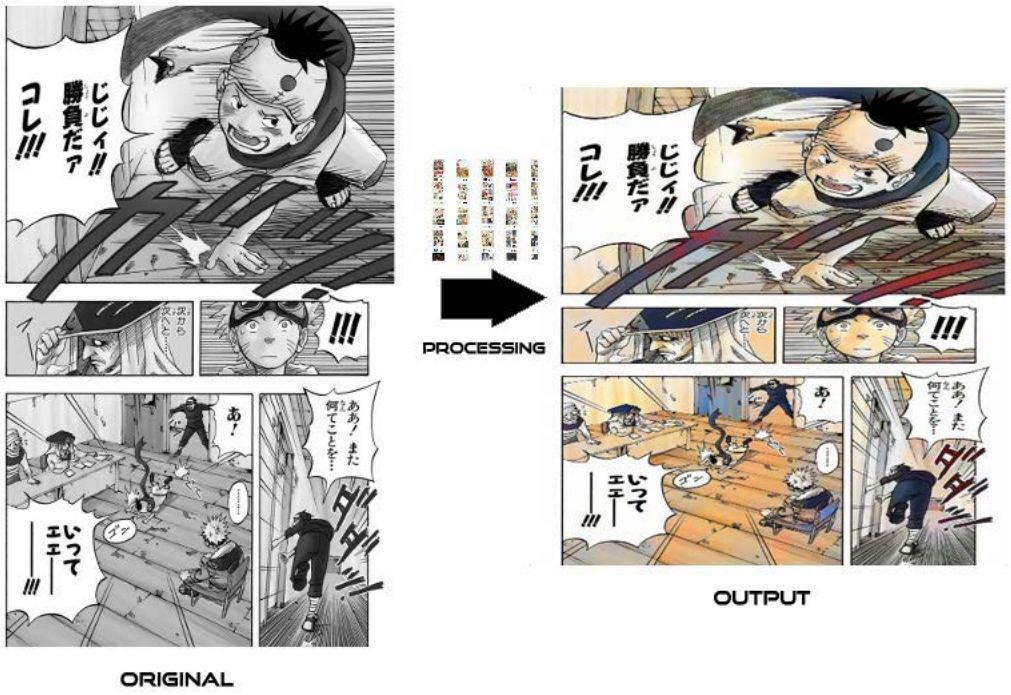


Figure 14. Overall process: Black-and-white pages are automatically colored according to a set a training colored pages

However, this is where the usefulness Hensman's process ends. In my target images above, I had obtained the above target black-and-white images by converting colored images into grayscale. This ensures that the target images have the best hue for logical automated coloring, and this was Hensman's method for obtaining target images to be colored. Needing colored images to be converted into grayscale for colorization is not practical because there is no need to automate colorizing pages that have already been colored. I need to automate coloring pages that have never been colored before. I need to find actual comic pages to color in.

Applying Hensman's process to comic pages closer to actual comic pages produces these results:

Figure 15. Hensman's colorization method applied to common comic pages

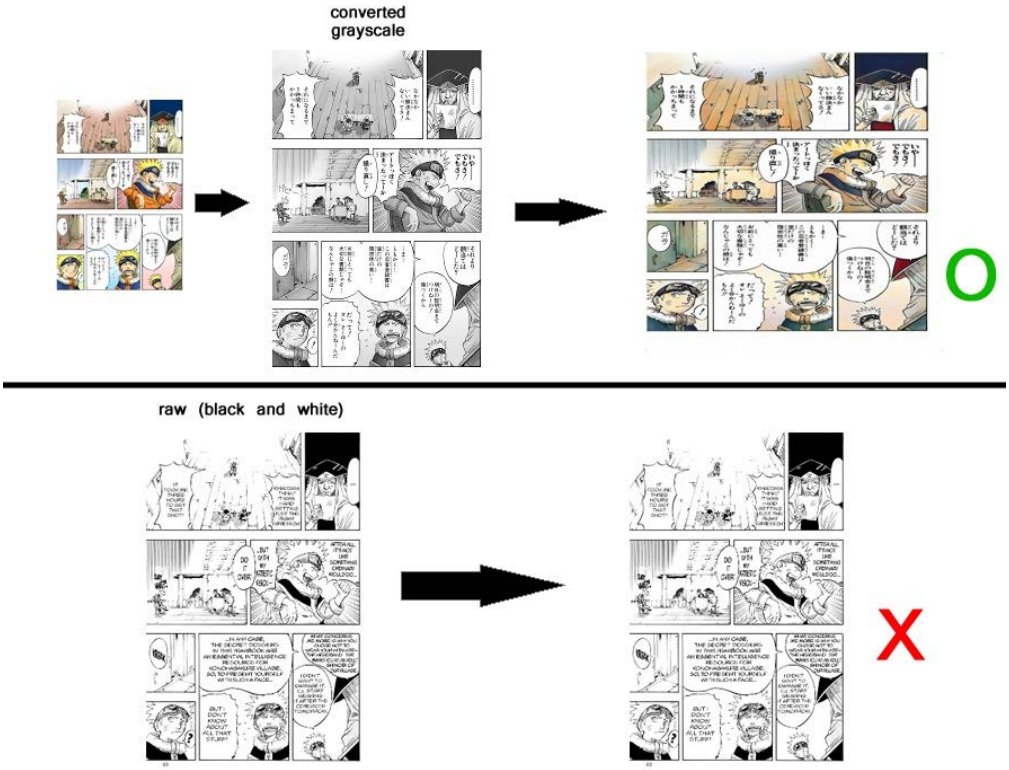


Figure 16. Hensman's method performs best if the target images have been converted to grayscale from already colored pages. This is not practical.

The colorization process produces little to no coloring! The reason is because Hensman's colorization process uses pix2pix as its backbone. A function of pix2pix is to restore in black-and-white historical photos into color. The processes use hue similarity to estimate what color a patch of grey ought to be. Due to the nature the busy lifestyles of comic artists, there is little shading, or tone, drawn in for most comics. Hensman's process cannot be realistically applied to most comics and black-and-white artworks.

To overcome this newfound weakness, I have investigated and offer three improvements: accept only highly shaded and detailed pages, build the cGAN model from certain weights, and increase the number of convolutions performed.

Solution 1: Photorealistic Detail

I gathered a mass of black-and-white comic pages that are clearly more photorealistic than the average comic page and applied my colorization method on this sample group.

The key difference that I am qualifying between this sample group and the ones I used for the control group is that this sample group contains more photorealistic artwork. But what does that mean? What I am looking for is a clear level of detail for these photorealistic artworks. The artwork is more detailed because the artists have inserted gradient, shading, or tone. And because the comic artists inserted shading, there is an imitated level of grayscale in these images. The shading and tone within these black-and-white pages produce a black-and-white-like realistic photograph – exactly the type of image that pix2pix and other colorization methods are trained for! Figure 17 displays the sample group that I need: a set of photorealistic comic pages that I predict will be more compatible with the colorization.

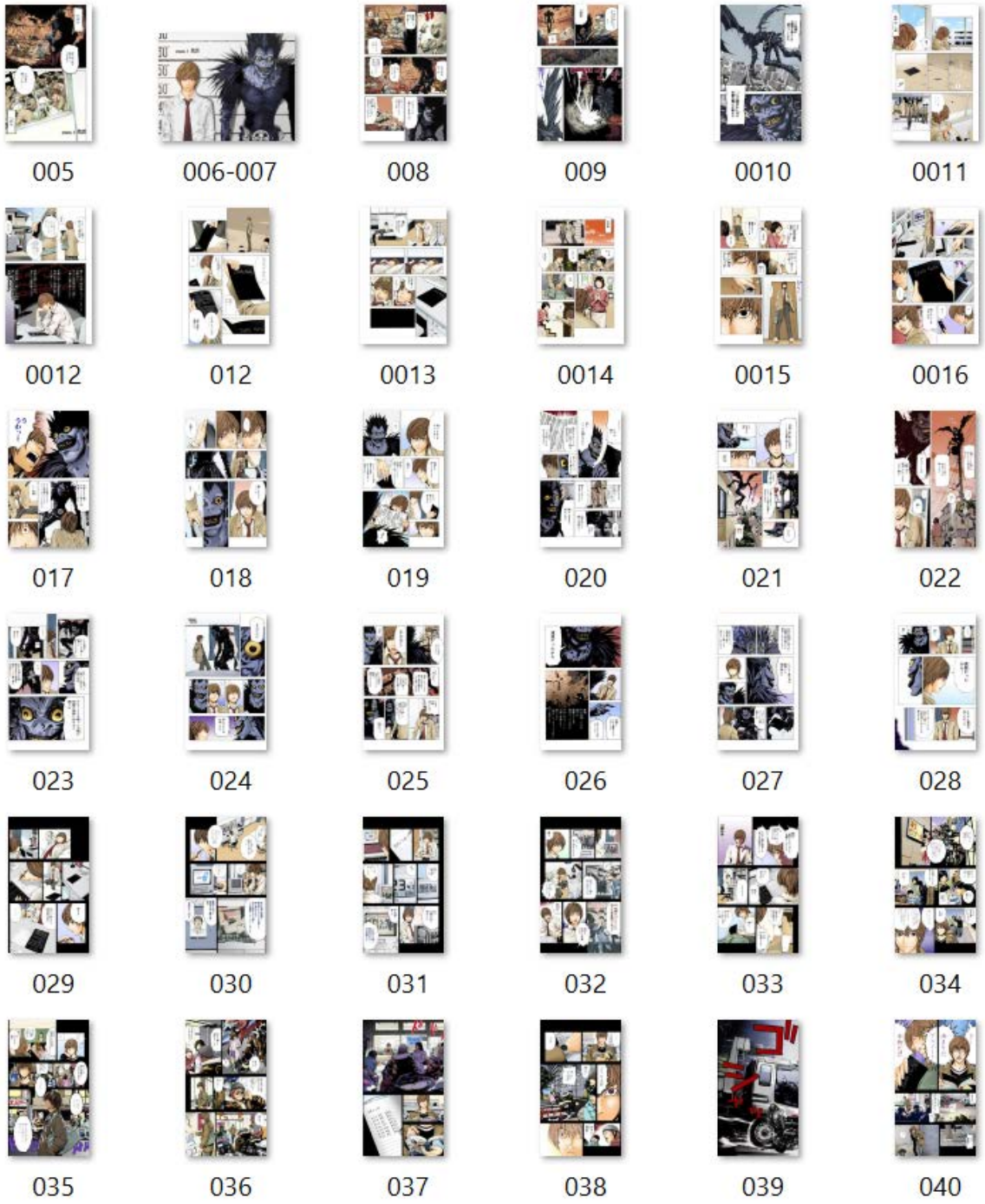


Figure 17. Training set for my photorealistic comic pages

This training set will be coloring the images in Figure 18: equally photorealistic pages that need to be colored.

Figure 18. One of my target sets for realistic comic coloring

And here are the results in Figure 19.

Figure 19. Results for realistic comic coloring. Much better coloring results than trying to color common comic pages

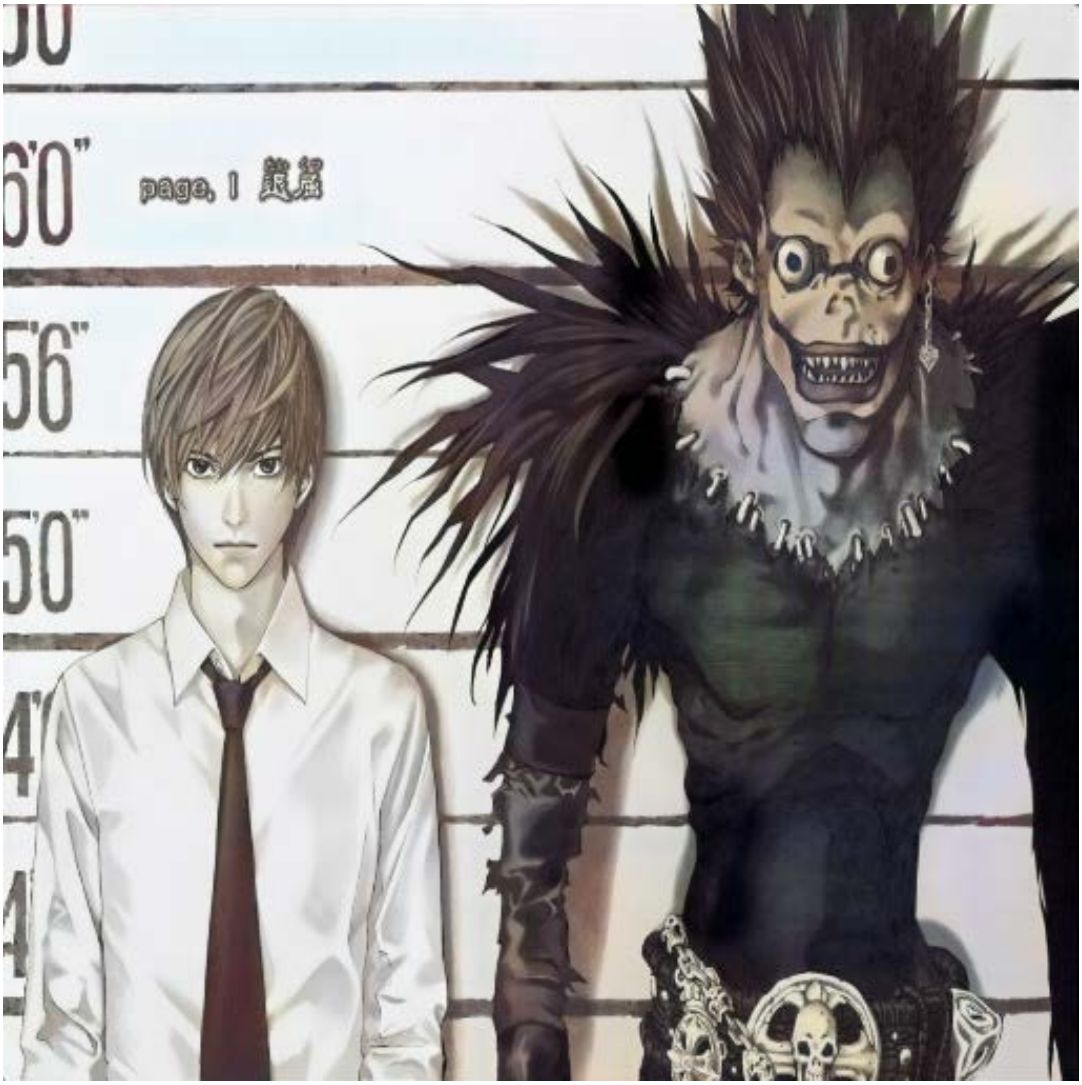


Figure 20. Close-up of an automated colorized page, from a more photorealistic artwork

Using comic artwork more realistic and closer to photography generates better coloring from my automated colorization process, as expected from my prediction and reasoning! A comparison can be seen in the figure below.

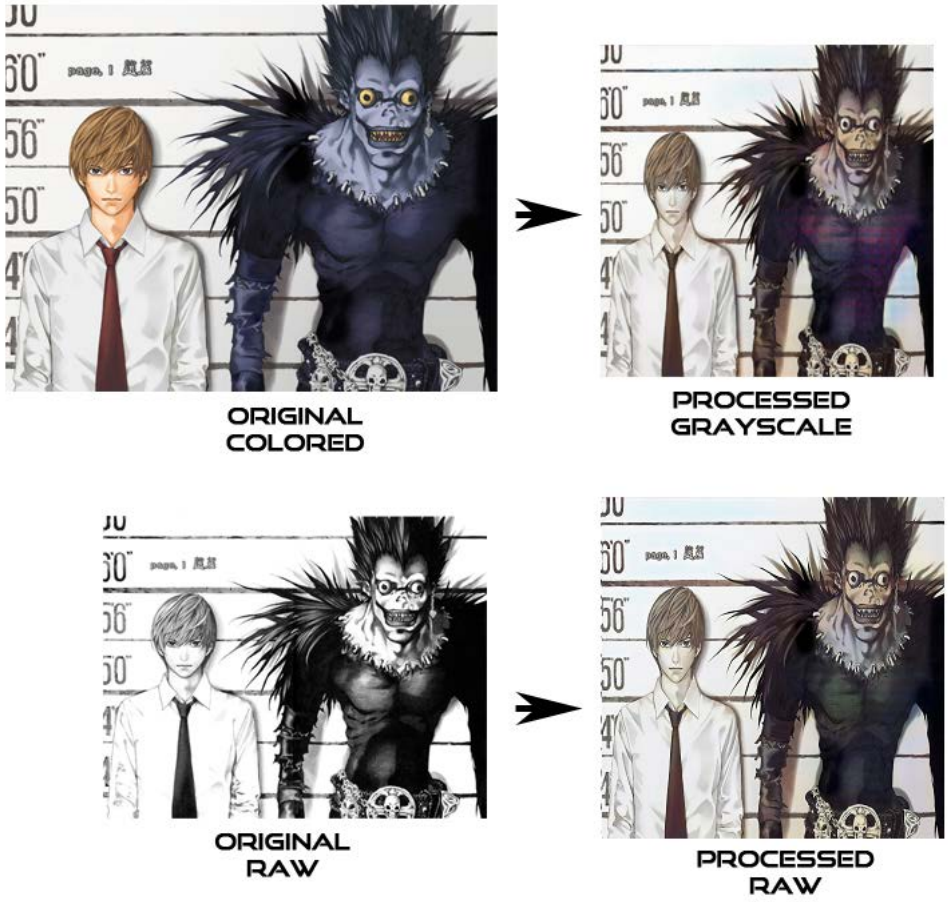


Figure 21. A comparison between different methods of colorization.

To recap, Hensman's method of using grayscale comic pages produces very accurate colors but requiring an already colored image for colorization makes its use limited. Finding detailed black-and-white images with enough detail that mimics grayscale is the next best option for colorful results.

Solution 2: Model Weight

From frequent testing of my photorealism theory, I arrive at my next conclusion. Input weights for my cGAN affects the output images as shown below. The more I run the colorization sequence, even with the same input data, the better the images are colorized.

Figure 22. difference colorizations generated from different weights

Although minute, the later colorizations have more accurate coloring. The differences may not be obvious to the naked eye. I layer the images on top of each other and apply pixel difference to see the difference. The difference of images that are completely that same would appear all black.

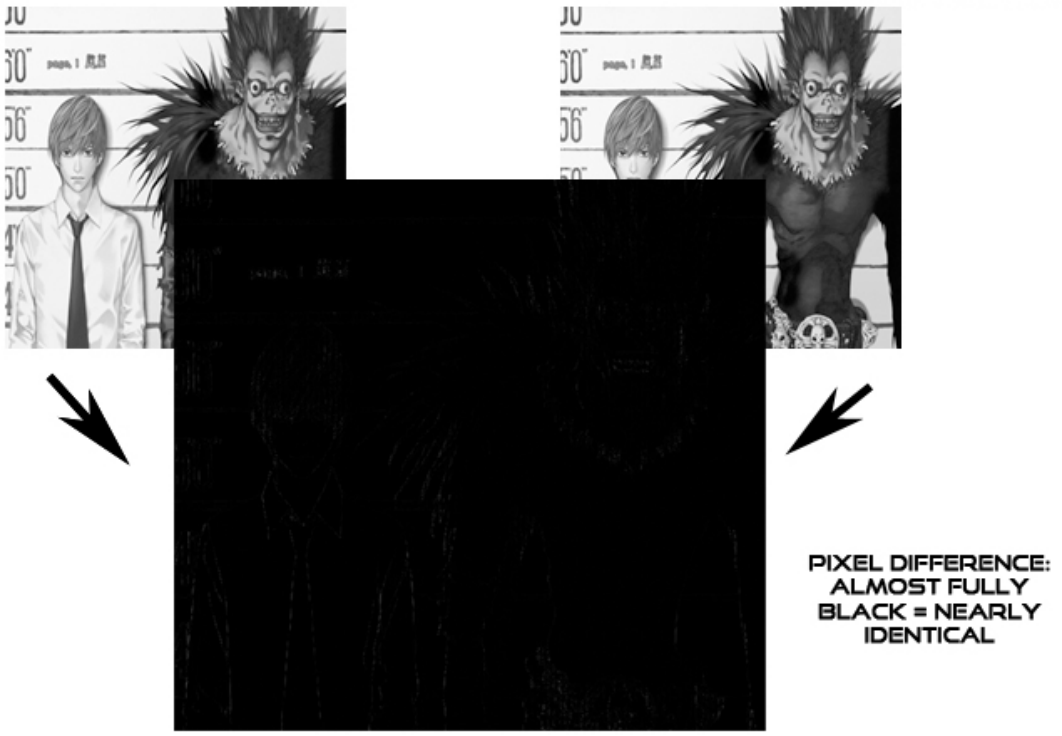


Figure 23. Using Pixel difference as a tool to find differences between images

Even with all the input data constant, the colorization changes slightly each run. This is because my weights are saved, updated, and used as input in the next iteration every time.

```
if e%5 == 0:
    cGAN.save_weights(store+str(e)+'.h5')
```

This would suggest that running my colorization process many times with many different training data sets would improve the colorization process. However, it is difficult to measure which training data sets improve coloring, since the differences are so small.

My weights are stored in .h5 files. Upon examining the .h5 files in a specialized file reader, I still have a hard time distinguishing one weight from another. If these weights can be kept better track of, using specific weights for specific types of colorization would improve the colorization accuracy.

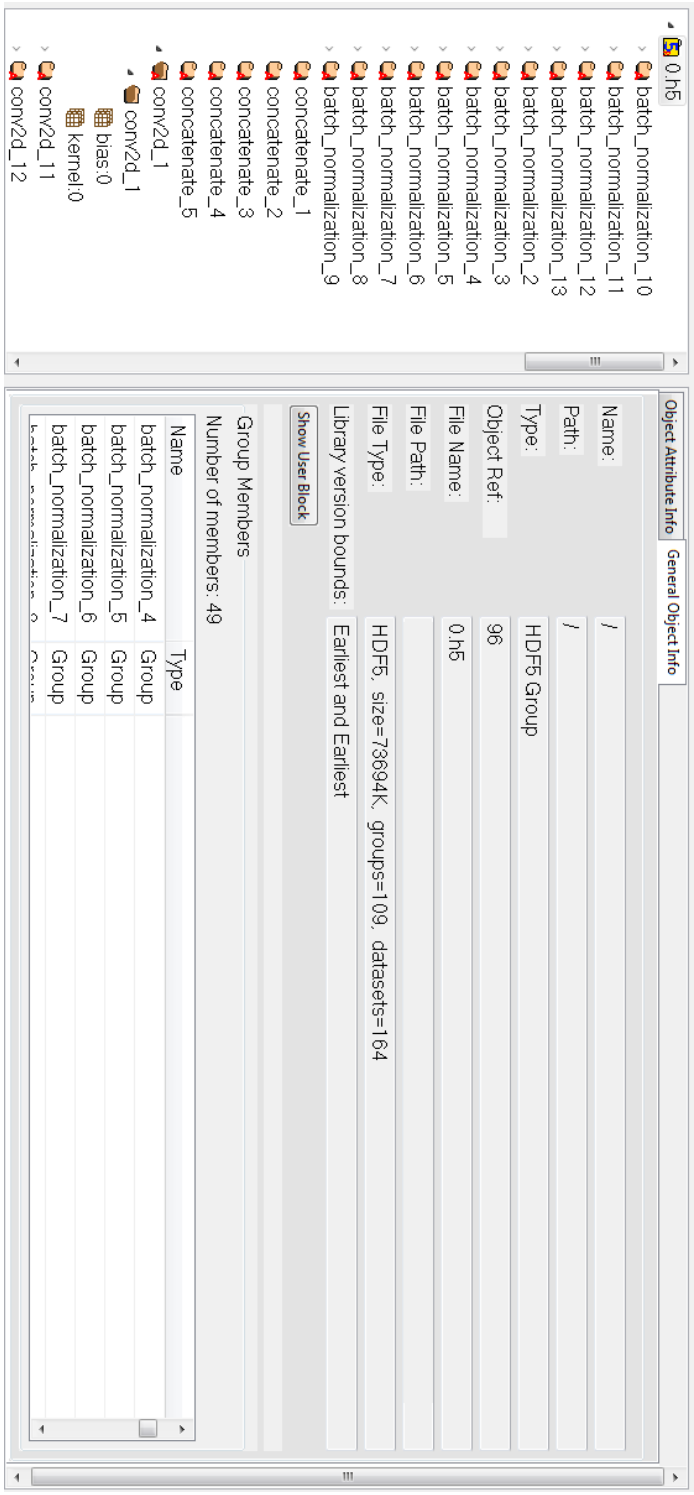


Figure 24. Weights are stored in H5 files and can be read via a H5 reader

Solution 3: More Convolutions

One more approach is to add more convolution layers in the cGAN model. Commonly, image convolutions are performed on 512px x 512px images. The square shapes from these images make for simple convolution calculations. Adding an extra layer of convolution and using 1024px x 1024px images does not produce much noticeable results, as shown below.

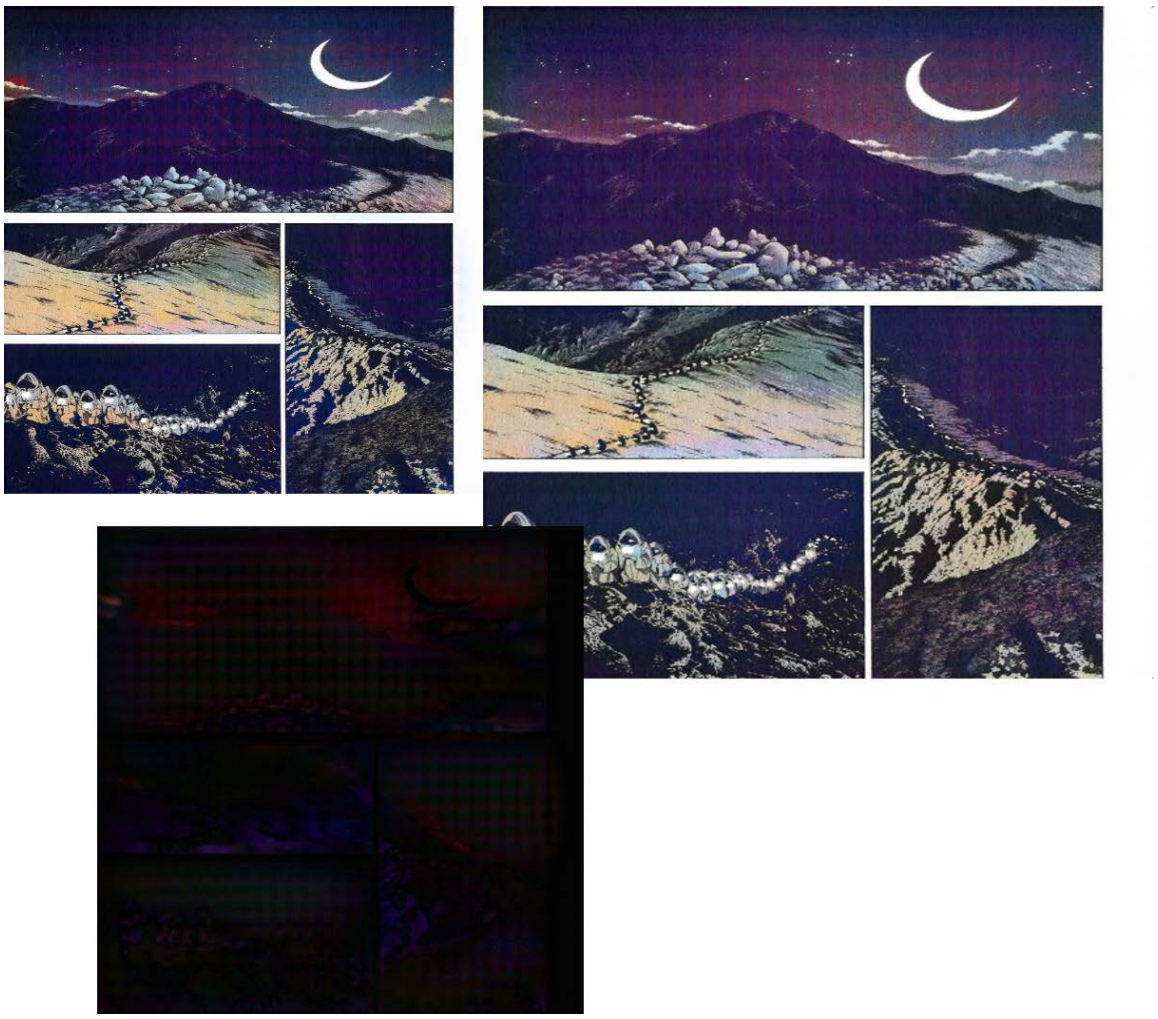


Figure 25. An automated colored page, along with its 1024x1024 convolution version, and their difference index, showing that there is little difference between the two

Extras

During my investigation, I found some interesting bits that are not directly related to my primary investigate but should still be reported for their significance. One such discovery is the automated coloring of characters. My colorization process will color reoccurring characters the same way each time. However if new characters are introduced, they will be colored the same way as a reoccurring character until the coloring process has been trained to color the new characters according to their proper color schemes.

Figure 26. A new character (bottom right) has been introduced but is colored the same as the other character. This can be fixed by adding colored images of the new character into training

CHAPTER 6 – CONCLUSION

The Last Chapter

Continuing from Hensman's cGAN-based Manga Colorization, I have created more streamlined colorization process, powered by artificial intelligence. Input images are a set of colored pages and another set of black-and-white images. The goal is to color the black-and-white images automatically. Building from Keras, Tensorflow, and Python, I take self-selected sets of samples pages of comics to use for training and validation. The comic pages used for training are in color. Comic pages used for validation are my target images and are in black-and-white or converted into black-and-white. My goal is to color my black-and-white target images with the color scheme of my training images. This is method is a form of a machine learning technique known as style transfer. My model's architecture composes of several convolution layers and deconvolution layers with validation feedback via conditional generative adversarial nets.

A limitation I have found is that while traditional image coloring convolutional neural networks exist, these processes are trained from black-and-white photo coloring. The gray hues within black-and-white photo hint and help the colorization process figure out the correct color. These hues are generally not present in the black-and-white comic pages that I am trying to automatically color. The gray shades in a comic page are much more distinct. Because the page is hand-drawn, the color details are understandably less complex than a photography. This presents a problem within the

neural network because without the many gray hues, the network cannot figure out what to color parts of a page. Put simply - large areas of a black-and-white comic page remain black-and-white because the neural network does not have enough hues near the pixels to figure out what color should be colored in. Black-and-white comic pages are too different from the black-and-white photographs that the original foundation coloring neural network is built for.

To overcome these limitations, I have come up with these solutions:

1. Improve the weights of my cGAN.

I have noticed through excessive trials that certain weights color my target images more accurately than others. By using weights that I have personally found effective, I can improve the colorization accuracy.

2. Limit the scope of target images to the highly detailed comic pages.

My neural net responds to more photography-like comic pages better. The more detailed and shaded a comic page is, the better colored the page turns out.

3. Expand the number of convolutions performed.

A third usage is restoring colored pages that have been lost, but somehow have their grayscale versions available. Perhaps a colored page was photocopied once into black-and-white, but the original colored page is lost. In which case, my neural net responds best

This thesis has been a journey of success and learning. There have been some areas in my investigation that did not go as planned. Some of my coloring methods are not as impactful as I had hoped. But, learning that these methods do not work is an

important aspect of a scientific study as well. Both my successes and not-so-successes are equally important from a contribution standpoint.

REFERENCES

1. Baseel, Casey. 2015. “One Piece Manga Creator's Work Schedule Is Absolutely Insane.” SoraNews24. October 9, 2015. https://soranews24.com/2015/10/09/one-piece-manga-creators-work-schedule-is-absolutely-insane/.
2. Collins, Hannah. “The Life Of A Manga Artist Is More Hellish Than You Think.” Ranker. https://www.ranker.com/list/why-being-a-manga-artist-is-terrible/hannah-collins.
3. Hensman, Paulina, and Kiyoharu Aizawa. 2017. “CGAN-Based Manga Colorization Using a Single Training Image.” Thesis. University of Tokyo. https://arxiv.org/pdf/1706.06918.pdf.
4. Hui, Jonathan. 2017. “Generative Adversarial Nets (GAN) , DCGAN, CGAN, InfoGAN.” March 5, 2017. https://jhui.github.io/2017/03/05/Generative-adversarial-models/.
5. Isola, Phillip, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. 2018. “Image-to-Image Translation with Conditional Adversarial Networks.” ArXiv. Thesis. UC Berkeley. https://arxiv.org/pdf/1611.07004.pdf.rg/bibliographies/219418746
6. froglarTBGE. “Manga Recon Show #009: Schedule of a Weekly MangaKa.” March 5, 2009. https://froglartbge.livejournal.com/2713.html.

7. “Introduction to Conditional GAN.” Mapt. https://www.packtpub.com/mapt/book/big_data_and_business_intelligence/9781788396417/3/ch03lvl1sec17/introduction-to-conditional-gan.
8. “Japan's Animation Industry Failing to Cultivate Next Generation of Talent.” 2018. Nippon. January 14, 2018. https://www.nippon.com/en/currents/d00337/.
9. Kang, Sungmin, Jaegul Choo, and Jaehyuk Chang. 2017. “Consistent Comic Colorization with Pixel-Wise Background Classification.” Thesis, NIPS. KAIST. https://nips2017creativity.github.io/doc/Consistent_Comic_Colorization.pdf.
10. “Keras: The Python Deep Learning Library.” n.d. Keras Documentation. https://keras.io/.
11. Krishna. 2018. “Introduction to Exponential Linear Unit – Krishna – Medium.” Medium. April 12, 2018. https://medium.com/@krishnakalyan3/introduction-to-exponential-linear-unit-d3e2904b366c.
12. Mehra, Ravish, Nikunj Raghuvanshi, Lauri Savioja, Ming C. Lin, and Dinesh Manocha. "An Efficient GPU-based Time Domain Solver for the Acoustic Wave Equation." Applied Acoustics, February 2012, 83-94. https://www.sciencedirect.com/science/article/abs/pii/S0003682X11001605
13. Okeda, Daisuke, and Aki Koike. 2014. “Working Conditions of Animators: The Real Face of the Japanese Animation Industry.” Creative Industries Journal. Thesis.
14. Patait, Abhijit. 2017. “VRWorks Audio SDK in-Depth.” NVIDIA Developer. June 13, 2017. https://developer.nvidia.com/vrworks-audio-sdk-depth.

15. Preferred Networks. “AI-Powered Automatic Colorization.” PaintsChainer. https://paintschainer.preferred.tech/index_en.html.
16. Rhiannone-10. 2016. “The Top 14 Series With The Best Manga Artwork Ever.” MyAnimeList. December 19, 2016. https://myanimelist.net/featured/2070/The_Top_14_Series_With_The_Best_Manga_Artwork_Ever.
17. Sherman, Jennifer. 2017. “NHK Program Discusses Anime Industry's Financial, Working-Condition Problems.” Anime News Network. June 7, 2017. https://www.animenewsnetwork.com/interest/2017-06-07/nhk-program-discusses-anime-industry-financial-working-condition-problems/.117144.
18. Sigtia, Siddarth, Adam M. Stark, Sacha Krstulovic, and Mark D. Plumbley. 2016. “Automatic Environmental Sound Recognition: Performance versus Computational Cost.” Thesis. IEEE. https://arxiv.org/pdf/1607.04589.pdf.
19. Simo-Serra, Edgar, Satoshi Iizuka, Kazuma Sasaki, and Hiroshi Ishikawa. 2016. “Learning to Simplify: Fully Convolutional Networks for Rough Sketch Cleanup.” ACM Transactions on Graphics. Thesis. Waseda University.
20. Wei, Honghao, Yiwei Zhao, and Junjie Ke. n.d. “Automatic Manga Colorization with Hint.” Semantic Scholar. Thesis. Stanford University. https://pdfs.semanticscholar.org/08cf/d4a6005f00b6affa8e69e23bd73c31ddb097.pdf.

21. Zhang, Lvmin, Yi Ji, and Xin Lin. 2017. “Style Transfer for Anime Sketches with Enhanced Residual U-Net and Auxiliary Classifier GAN.” ArXiv. Thesis. Soochow University. https://arxiv.org/pdf/1706.03319.pdf.

APPENDIX A – MACHINE LEARNING TERMS

cGAN	Conditional Generative Adversarial Network
GAN	Generative Adversarial Network
pix2pix	An Image-to-Image Translation with Conditional Adversarial Network created by Phillip Isola, Jun-Yan Zhu, Tinghui Zhou Alexei A. Efros pix2pix is referenced across many technical papers in the cGAN field
Target Image	The image intended to be altered by Machine Learning algorithms.
Ground truth	the accuracy of the training set's classification for supervised learning techniques
Image Segmentation	Pixel by pixel classification into any of a class of subjects within a given scene
Python	interpreted high-level programming language for general-purpose programming
Keras	high-level neural networks API, written in Python, for deep learning
TensorFlow	open source machine learning framework
generator	In a GANs, a network that generates new data instances
discriminator	In a GANs, a network that evaluates the data from the generator for authenticity

Cost function	In machine learning, a measure of how wrong the model is in terms of its ability to estimate the relationship between two inputs of a model
Log-probability	In computer science, the logarithm of a probability
numpy	fundamental package for scientific computing with Python
gradient	vector at every point pointing to the next local minimum of your function
tensor	mathematical object analogous to but more general than a vector, represented by an array of components that are functions of the coordinates of a space.
kernel	Filters by which convolutions are performed
Convolution	2D mathematical operation performed on a matrix small parts at a time
Deconvolution	Upsampling, or backward-strided convolution
Padding	Extra zeros added to the boundary of matrices during convolution
ELU	Exponential Linear Unit An activation function that is linear for positive input but near zero for negative input
Activation function	In artificial neural networks, the activation function of a node defines the output of that node
Style transfer	technique of recomposing images in the style of other images
Zero-sum game	In game theory and economic theory, a

zero-sum game is a mathematical representation of a situation in which each participant's gain or loss of utility is exactly balanced by the losses or gains of the utility of the other participants.

APPENDIX B – COMIC TERMS

Manga	comics created in Japan or by creators in the Japanese language
Mangaka	Japanese word for manga artist
Anime	hand-drawn and computer animation originating from or associated with Japan
Screen tone	Comic shading is inked via method that uses a sheet of ink pressed into the paper. This