Deep-Learning Color for Manga and Anime Sketches

Code at: https://github.com/DeanHazineh/Deep-Learning-Color-for-Manga.git

Dean Hazineh

May 13, 2020

Contents

1 Introduction/Abstract

My goal for this final project is to explore and learn about general adversarial networks (GANs) while also studying its application to a topic of great interest in my personal life–that is, a form of media called Manga. For those who are unaware, Manga can be summarized as a type of comic/graphic novel created in Japan and the focus of the stories told in this medium cover all genres including fantasy, comedy, horror, etc. A more thorough background can be found quickly online [Wikipedia, [1]], but the most important thing to recognize in order to follow this report is that this medium has

two very fundamental features characterizing it– Manga prints (1) have a very particular art style specifically in regards to the way in which male and female characters are drawn and (2) are all printed almost exclusively in black and white (minus the cover page). The later aspect can be primarily associated to time constraints given that most Manga, including those with immense global success, are typically hand drawn by a single artist and with only a few assistants. The question I want to explore in this project addresses these two ubiquitous features of the medium.

In this work, the goal was to explore if deep learning can be employed in order to enable Manga artists to create colorized Manga prints instead of the traditional black-and-white prints without requiring significant extra time or work for the creator and without requiring a compromise to the mediums style. While much of the results shown clearly display the results of the investigation into machine colorization, the question of sketch enhancement is also subtly probed as I seek to create an architecture which not only colorizes an image but starts with a unpolished, early stage sketch devoid of quality shading. By quick inspection, one can find that the black and white prints in any Manga look significantly better than the "sketchified" black and white drawings in this work and that is in fact intentional. It would be considered a success in my opinion if one can begin with a less final starting image (reducing drawing time for the artist) and still then achieve both style preservation, feature finalization and full colorization by implementation of our proposed architecture. Here, I highlight that I do not believe an un-supervised colorization approach makes sense as a real-world solution for this particular problem, and as a result, I have designed the code focusing on a proof-of-concept solution that allows the artist to give input via the "color highlight" used for the overall image. With that being said, I have also generated results and a study excluding the artist input entirely (setting color inputs to a zero-matrix and retraining the model) in order to investigate in an academic nature what the results would be. For this report, I think both are equally interesting and amusing investigations worth presenting.

In terms of impact, creating a solution to this challenge is actually an extremely exciting prospect which may be less obvious to those who are not fans of the genre. According to sources cited on two reports [Anime News Network [2], Exoclick [3]], Manga make up almost 40% of all books and magazines published in Japan and is estimated to generate the equivalent of 4 billion revenue yearly, not including ad sales and the growing potential of further digitization. In China, the market for Manhua which is similar to Japan's Manga, is estimated to be valued at 26 billion US dollars and has grown almost 14% since 2017. If that is not impressive enough, the famous Japanese Manga, One Piece, which is still in production, has currently sold over 460 million print copies globally almost

matching the highest selling book of all time, Don Quixote by Miguel De Cervantes. Lastly as a die hard Manga fan and life-time reader, I can attest that colored prints in Manga are a luxury that many in the community would love to see happen but have previously accepted to be an impossible prospect and one that could only come at the cost of significant delay to the standard weekly chapter publication time. At this moment in time, only the most famous Manga, e.g. One Piece, can be re-published in digital color which also occurs with a time-delay relative to the primary black-and-white releases and to my knowledge, this is only ever done within the industry manually by professional artists. My hope is that this work can generate new insight and inspiration into an automated and more general implementation to this light-hearted desire.

2 Architecture Motivation and Background

Examining the literature, one can find that there has actually been a number of academic investigations into the question of deep-learning color for black and white images. There has in fact even been some exploration by researchers directly into the problem of deep-learning color for Manga itself [cGAN-based Manga Colorization [4]]; however, I found this work and others to be largely incomplete/inconclusive or, in my opinion, flawed regarding a realistic extension to industry and use by artists. From my search, all previous works have explored the topic of unsupervised deep-learning for colorization, which as noted before is not the primary implementation I have in mind, but since they serve as a backdrop to my architecture and one of the side-explorations conducted here, I first briefly review the recent developments on this topic.

Although outside the context of Manga, a great and relevant introduction to the topic of unsupervised, deep-learning colorization is given by Richard Zhang et al at Adobe Research [Group Github, [5]]. By training a feed-forward pass CNN on over one-million images from the imagenet database, their group published in 2016 the then state-of-the-art architecture for converting gray-scale photographs to vibrant colored images. He has aptly described this problem of general, unsupervised colorization as “the problem of hallucinating a plausible color version of the photograph”, but this is immediately seen to be a severely under-constrained inverse problem necessitating some form of support [Colorful Image Colorization Paper [6]]. By modifying the standard CNN algorithm (with some changes outside the context of this work) and with enough training data, the group demonstrated that highly realistic colorization can be achieved under the context of re-framing the problem as a large classification task. The performance is then found to be constrained largely by the usual “data-set feature bias”. This and previous works therein cited provide guidance that a CNN can be

an effective architecture for learning color thus encouraging its use as a starting point in our task.

With that said, however, the previously cited architecture by the Zhang team in its current form would be insufficient for this task without changes for two key reasons. First, the Manga training data set which is here created through a careful but automated process feasibly results in a set (at this current time) on the order of tens of thousands of images instead of millions of images like in the cited work. A full re-framing of our problem as a classification task where shapes are associated to colors, even in the supervised color hint scenario, appears unlikely to be effective[1]. Second and more importantly, their work and similar work in the field at that time (around 2016) implemented the Euclidean L2 loss between the ground truth and predicted color as the objective function. Their work has shown that without some complex randomization, this loss function favors grayish colors that leads to desaturated results in total contrast to the typical color palletes in Manga/Anime art-style. Furthermore, it has been well shown in recent years that this MSE type loss is largely ineffective for image translation tasks since it leads to blurry outputs and poor contrast due to an effective averaging of all possible outputs [Summary and note of papers [7]]. Without using an L2 Euclidean loss, I do not a priori know what is the best loss function is for this task and for this reason, I turn (with ample motivation in literature) to implementation of an adversarial loss. In theory, a GAN should not suffer from the mean image problem at all by virtue of optimizing an entirely different type of mathematical divergence.

Here, I find that the best solution/starting point for this task is to implement a conditional adversarial network with a CNN architecture and to focus predominantly on framing the problem as an image-to-image translation task. The motivation and guidance for which our implementation is closely built from is the famous, so-called Pix2Pix architecture released in late 2016 by A. Efros's team at the Berkley AI Research Lab [Pix2Pix Github [8]]. To summarize, an image-to-image translation task is the problem of translating one possible representation of a scene to another given sufficient training data. In the past, this has required customized and hand-engineered objective functions as seen in the paper from Richard Zhang's group; however, the exploration underlying the Pix2Pix implementation was to remove that step by creating a general image translation architecture which could work across different problems and to demonstrate that a conditional generative adversarial network (cGAN) can effectively self-learn the required loss function for many different image tasks [Image-to-Image Translation Paper [9]]. Again, coming up with the loss-function analytically is not only an open-research problem but it is a non-trivial task generally requiring expert knowledge since

[1] At this time, I am uninterested in exploring transfer learning from the imagenet set given a better alternative.

the standard L1 and L2 Euclidean distance minimization leads to blur. The particular details of the model used in this work is left to the Methodology section with rigorous theoretical treatment deferred to the original Pix2Pix paper [Image-to-Image Translation Paper [9]]; however, here I present a high-level summary of the architecture's unique and key features.

The two fundamental components of the GAN-based architecture are the generator and the discriminator–two separate neural networks that are each individually optimized in a feedback loop. The two networks play a role analogous to that of an art forger and an art critic and are pitted against each other in a zero-sum game where the generator is tasked with creating a fake painting to fool the critic (the machine generated colorized Manga) and the discriminator is tasked with identifying the fake art from the original art (the artist colorized Manga in the training set). The training process for the GAN entails that the generator actively learns what group-features comprise the original art and gets better at reproducing it while the discriminator simultaneously learns new tricks to distinguish and identify the forged art. In summary, the GAN learns a numeric loss function that classifies an output image as real or fake while simultaneously training a generative model that minimizes this loss. In the cited work and in my work here, I utilize an objective function that is actually a combination of the GAN loss with a L1 loss along with slight modifications to a conventional CNN generator and discriminator as discussed in the following section.

3 Methodology

In this section, I outline in detail the experiment carried out in this work. In summary, the goal of the project is to create a mapping that takes an artist's sketch and any color specifications and turns it into a final colorized image with the intended style preserved. Specifically, the input passed in to the model is a 256 x 256 x 1 binary image encoding the black-and-white initial sketch in addition to a 256 x 256 x 3 RGB image encoding a hint/instruction regarding what general color scheme the artist wants the AI model to use when colorizing the image (this is referred to throughout as the color cue). The output would then be a 256 x 256 x 3 image that is both a colorized and feature enhanced rendition of the black and white sketch originally given to the generator. Here, I study the quality of the output image in two cases– (A) when no color cue is provided such that the color cue matrix is initialized to an array of ones for all sketches and (B) when color cues are given. These two experiments are pictorially diagrammed in Figure. 1. For each experiment, the discriminator and generator are trained for 30 epochs on an NVIDIA GeForce RTX 2070 Mobile GPU unit (7 GB

memory) which takes approximately 2.7 hours for either case. The set of three images shown in in Figure 1 for Experiment A and B accurately depict the type of images used in the training set for each experiment respectively. For both cases, the training data images are the same excluding the obviously different color cue matrices. The training data used for this report consists of approximately 9000 unique sets (i.e. 9000 triplets of sketch + color cues + artist colorized images). The testing data set includes images hand-picked by myself which stress-tests the trained model in various ways and has never been previously shown to the generator during training.

3.1 Training Data Generation

A single data-point for fitting/training the model requires three components: (1) a high-quality, digitally colorized anime/manga-esque image, (2) a corresponding binary sketch version of that image, and (3) (in the case of the color cues experiment) a corresponding color cue that I can imagine an artist would provide the hypothetical generator in order to get out the real, digitally colorized image. In order to make this project realizable, each of these three components must be quickly obtained and/or generated in a fully automated process and here I discuss how this was achieved. For the related code, see the scripts "download_images_2p7.py" and "process_images_and_batch3p5.py" on the linked Github Repository.

The first step in the training data generation process is to obtain a large set of digitally colorized images that fit the style of Japanese manga and anime (see the "model output" image shown in Figure 1. This is actually easily done by writing a web-scraping python script that automatically searches and downloads tagged images from the imaging hosting website, Danbooru [Danbooru URL[11]]. This site is ideal for this task since it is essentially a large-scale crowdsource and tagged anime dataset with nearly 4 million anime images (and reportedly over 108 million image tags total allowing quick filtering and searching [Danbooru Machine Learning Guide [12]]). Fair Warining: many of the images on this site are arguably not "safe-for-work" content. Using the code released on my github, I downloaded approximately 9000 images (at about 1 second per image) with fantasy themed tags like "holding staff" and "magic solo". After, these images are further processed in python via resizing and cropping such that all images are converted to a 256 x 256 square. This then satisfies component 1 making up the high-quality, digitally colorized manga-esque art.

The corresponding sketches and color-cues are then obtained by automatic image processing on these downloaded, colored images. In order to obtain the black-and-white sketches (again devoid of effects like shading since I want to reduce drawing time for artists) from the digitally colorized

images, I apply an edge-detection algorithm via OpenCV which implements Gaussian-weighted adaptive thresholding to binarize the image. Here, there is freedom of choice to decide what blocksize to use in the algorithm, i.e. the number of neighboring pixels to sum over when thresholding, and different values produce a slightly different appearance to the "sketchified" image. An example of a colored image and the corresponding sketches resulting from this method with different blocksize parameterizations is displayed in Figure 2. Throughout this work, I utilize a blocksize of 7 since it provides some distinction between primary edges and finer details as if the artist had utilized two different pencil sizes which seems realistic. Above these images, there is also displayed a "complexity value" for the sketch which is utilized in a system to automatically identify images for which this sketch process does not work well by virtue of the colored images having too many edges or abnormal global color gradients. The complexity value is simply computed by calculating the percent of all pixels that has been set to 0 (the black edges). From inspection of samples in the data set, I discard all images with a complexity value greater than 40 at blocksize characterization 7 (i.e greater than 40% of the pixels depicting an edge). For reference, the complexity distribution of one tagged imageset after the sketchify process was completed (approximately 2000 images) is displayed in Figure 3 and shows that most images satisfy the criteria of having a complexity value less than 40. For clarity, in Figure 4, I display three examples of the colored images and their corresponding sketches.

For the last step in the training data generation process, I establish a method to automatically derive color cues corresponding to the colored images. I imagine an effective system would be for an artist to very quickly create color cues by drawing over the sketch with large highlighters of different colors. In the imagined process, they would also need not worry about coloring within the lines or specifying variations in color or shade. To approximate this in a way that could be sufficient for a proof-of-concept implementation and without requiring any manual dataset labeling, I create the color cues for each image by spatially blurring the colored images with a large Gaussian kernel. From testing, I found that a Gaussian filter with a standard deviation of 20 pixels qualitatively produces the desired effect. For reference, this corresponds to a FWHM of approximately 50 pixels which is nearly 1/5 the width of the image. Three examples of the color cues derived from this method and their corresponding colored images are shown in Figure 5.

3.2 Generator and Discriminator Model

The model for the generator and discriminator used in this work is the same as that reported in the Pix2Pix paper [9] with a review of the technical details summarized here. For a quick visualization, a schematic of the generator is shown in Figure 6 and that of the discriminator is shown in Figure

7. Both the generator and discriminator utilizes modules of the form convolution-BatchNorm-ReLu. For the related code assembling the generator and discriminator, see the scripts "mainv3.py" on the linked Github repository.

A key design consideration for the generator is based on recognizing that the input and output image (the black-and-white sketch and the colorized image) share similar underlying structures via the edges of characters and objects. The standard approach for an image generator is to employ an encoder-decoder network, but this then requires that information for which I wish to preserve in the input successfully travels through the full length of the net, as well as through the bottleneck. To better account for this low-level information sharing between the input and the output, I add skip connections between layers in the generator thus allowing some information to be shuttled across the net more directly. This is referred to as a "U-Net" implementation and each skip connection implies a concatenation of all channels at the two layers.

To provide context for the discrimator design choices, I first begin by commenting on the objective function used in this work. Motivated by previous studies in the literature, it is beneficial for this task to use a total loss that is a weighted combination of the GAN objective and the more traditional L1 Euclidean distance. Specifically, while the GAN loss forces the generator to create a result which fools the discriminator, the added L1 loss serves to further anchor the output by enforcing the constraint that the colors in the generated image appear similar to the ground truth image. Furthermore, while there may be many plausible colorization outcomes for a given black-and-white image, the L1 loss can encourage a particular color instantiation over other options especially for a character the network may have had limited exposure to previously. The final objective can be written as,

$$G = \arg\min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G), \tag{1}$$

where λ is the weighting factor and has been set to $\lambda = 100.0$ in this work. A numerical study of different values for the weighting factor was not conducted here and may be a future line of exploration.

While the L1 objective (like the L2 distance) is known to cause a blurry output image thus implying a loss of information in transferring high-frequency spatial components, it does, however, still accurately transfers and captures information in the low spatial frequency channels. As a result, by using the joint objective, one then only really require the GAN objective to enforce correctness on high-frequency structure. Because of this fact, one can achieve high quality results with less compu-

tational resources by implementing a unique discriminator architecture referred to as a PatchGAN [Pix2Pix paper [9]]. In this approach, the generator image is subsection into patches of pixel size NxN and the discriminator then examines the structure within each patch (now taken to be independent of any long range correlations outside the NxN subsection). Each patch is then classified as real or fake and the discriminator is applied convolutionally across the image. For this work, each logical output from the discriminator here refers to a 70x70 pixel receptive field in its input [PatchGan Refernece [14]]. Variations to this value were not probed or tested for this study.

3.3 Testing Approach

In order to evaluate the capabilities of the trained models for both experiments in a fair and insightful way, I review the performance of each on the same, carefully selected set of testing images. Six of these images can be seen in Figures 8 and 9. These six images were chosen as they collectively present three different levels of difficulty. As such, I have classified the six into three groups referred to in this work as evaluation "tasks". A summary of the anticipated challenge each task presents is reviewed following, listed in increasing order of difficulty:

- The Color Task:
  For images in this class, the model has trained on several other images directly containing these characters although presented in a different pose or drawn by a different artist. This will test the performance for cases where the model is well-conditioned to formulate the problem as a classification task and colorize with the aid of object/character recognition.

- The Transfer Task:
  No direct variation of the characters present in these test images have been previously shown to the model during training; however, the overall theme and complexity level of images in this task closely match that of the training set. This test will evaluate how well the model can perform true image translation with limited reliance on character memorization.

- The Interpolation Task:
  These images have complexity values well over 20% higher than the cutoff threshold defined for the training data set. The model has also never seen any variation of these characters, this type of drawing style (it no longer has features similar to the downloaded Danbooru anime set), or images with a similar density of edges.

4 Results

The results of the three aforementioned evaluation tasks are here displayed in Figure 8 for the colorization experiment where no artist color cues are provided to the generator and in Figure 9 for the colorization experiment where color cues are provided. For emphasis, the former can be classified as the unsupervised deep-learning approach while the latter is the supervised.

From qualitatively inspecting the generator output, I find that the unsupervised colorization model (no artist input) can effectively accomplish the basic "Color Task" meaning that the proposed architecture can, in this case, both color and shade the sketches with reasonable fidelity to the true colored image. While some colors in the generated image are off in shade, the generator output is largely believable and the color choices made by the model are consistently applied across the image with high contrast–this is the most important part in my opinion. Although the architecture can easily match the true color without hints in the "Color Task" by virtue of having seen the characters during training, it no longer has such support in the "Transfer Task" if no color cues are provided. Again, in this case, the model has never seen these characters and thus has no direct support to encourage one possible color hallucination over another. As a result, the model has no way to know that the girls hair in Figure 8 should be pink instead of brown or that the cat should be colored red instead of black. While the colors are different from the true image, I would argue that the generated colors in both cases are very realistic and plausible. Although the model internally classifies the stratified parts of the clouds to look like magic (which is consistent to the shapes in the training data) and thus colors it as so, all the color choices therein made appear well justified. The model has learned to associate witches robes with the color black along with skin as tan and without artist color cues, it aptly chooses that colorization to be the most realistic guess. While the output for the "Transfer Task" is not perfect, I would consider this output to be a success for the proof-of-concept and an impressive feet for deep learning. If the artist does not care to enforce a particular color pallete, then this output could still be promising. Finally, the last task is the "Interpolation Task" which is again the hardest since the generator must colorize an image outside the scope of the task it was trained for. Here, I find that without color cues provided, the generator is unable to make sense of the high density of features and edges and so fails to produce a realistic or believable colorization.

When color cues are now included as an input to the generator as displayed in Figure 9, nearly all the challenges in the previous case are mitigated. With color cues given by the artist, the model can, by qualitative metric, successfully accomplish both the "Color Task" and the "Transfer Task"

with impressive accuracy and fidelity. While I can still visually discriminate which of the images displayed in the Figures for the "Transfer Task" are the original vs the deep-learned image, I find that style is preserved and the generator output is a major enhancement of the crude sketch. If this can be further improved, it would be an immense time-saver for manga artists. Lastly regarding the "Interpolation Task" which is to serve as a stress-test of the generator, I find the output to be modestly clear and passable although of insufficient quality to encourage real-world use. This fact, however, is acceptable since if images of this class were desired, the automatic training generation process can be extended to include images of this nature.

5 Concluding Remarks

To summarize this work, the ultimate goal was to create a deep-learned model where an artist could pass in a sketch and some color cues to then get out a colorized and enhanced version of the image. In this report, I have explored this avenue along with a side exploration into an unsupervised approach with no color cues provided. Our finding is that if the training data is made to include instantiations of the character to be used in the artist's tasks, color cues may be unnecessary. Alternatively, if color cues can be given as will be expected in the real-world application scenario, the model performs the task at hand beautifully. Additional examples of the Manga colorization from color-cues is shown in Figure 10. I find this architecture and the state of technology at present to be promising and likely sufficient in order to build and deploy a realizable and smart solution for the Manga industry. With that being said, this remains an open-problem and the work shown here exists only as a rudimentary proof-of-concept. In order to really demonstrate feasibility, more care must be taken in the generation of the training data.

Specifically, it remains unclear how the model will respond with more realistic creations for the color cue. In actuality, one would not expect the color cues to look much like that generated for this work and shown in Figure 5. The color cues shown here inherently imply that every pixel is provided with an associated color hint while in reality, the colorization matrix would be sparse and incomplete if created by a human. Furthermore, the nature of the convolution approach implies that all variations in color shade from the true image are numerically encapsulated in the color cue matrix creating an inverse problem that is in fact better condition for inversion. This is entirely different than in reality. The silver-lining to this as presented in this report however, is that the unsupervised demonstration shows that results can be produced without any dependence on the color cues at all. This logic implies that even if the color cue matrix provided by the artist are of poorer conditioning

in terms of colorization support than that automatically generated in this work, the conclusions may be weakened but to no degree worse than that found in the no color cue case.

6 References

References

[1] Manga, https://en.wikipedia.org/wiki/Manga

[2] Anime News Network Posting, https://www.animenewsnetwork.com/news/2019-04-08/japan-manga-market-grows-1.9-percent-in-2018/.145512

[3] ExoClick Report, https://www.exoclick.com/the-manga-and-anime-market-is-big-business/

[4] P. Hensman and K Aizawa, "cGAN-based Manga Colorization Using a Single Training Image", Jun 21 2017, arXiv:1706.06918 [cs.GR]

[5] Colorful Image Colorization Github, Abstract, Richard Zhang et al., http://richzhang.github.io/colorization/

[6] Zhang R., Isola P., Efros A.A. (2016) Colorful Image Colorization. In: Leibe B., Matas J., Sebe N., Welling M. (eds) Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science, vol 9907. Springer, Cham

[7] MC.AI Review Post, The reasonable ineffectiveness of MSE pixel loss for future prediction, https://mc.ai/the-reasonable-ineffectiveness-of-mse-pixel-loss-for-future-prediction-and-

[8] Image-to-Image Translation with Conditional Adversarial Nets Github, Image-to-ImageTranslationwithConditionalAdversarialNets

[9] Image-to-Image Translation with Conditional Adversarial Networks, P. Isola et al. https://arxiv.org/abs/1611.07004

[10] TF Pix2Pix Tutorial, https://www.tensorflow.org/tutorials/generative/pix2pix

[11] Danbooru Image Hosting Site, https://danbooru.donmai.us/

[12] Danbooru Site Machine Learning Guide, https://www.gwern.net/Danbooru2019

[13] OpenCV Edge Detection, https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_thresholding/py_thresholding.html

[14] Article on Pix2Pix, https://machinelearningmastery.com/how-to-implement-pix2pix-gan-models-from-scratch-with-

List of Figures

Figure 1: Schematic diagram displaying an overview of the two experiments studied in this report

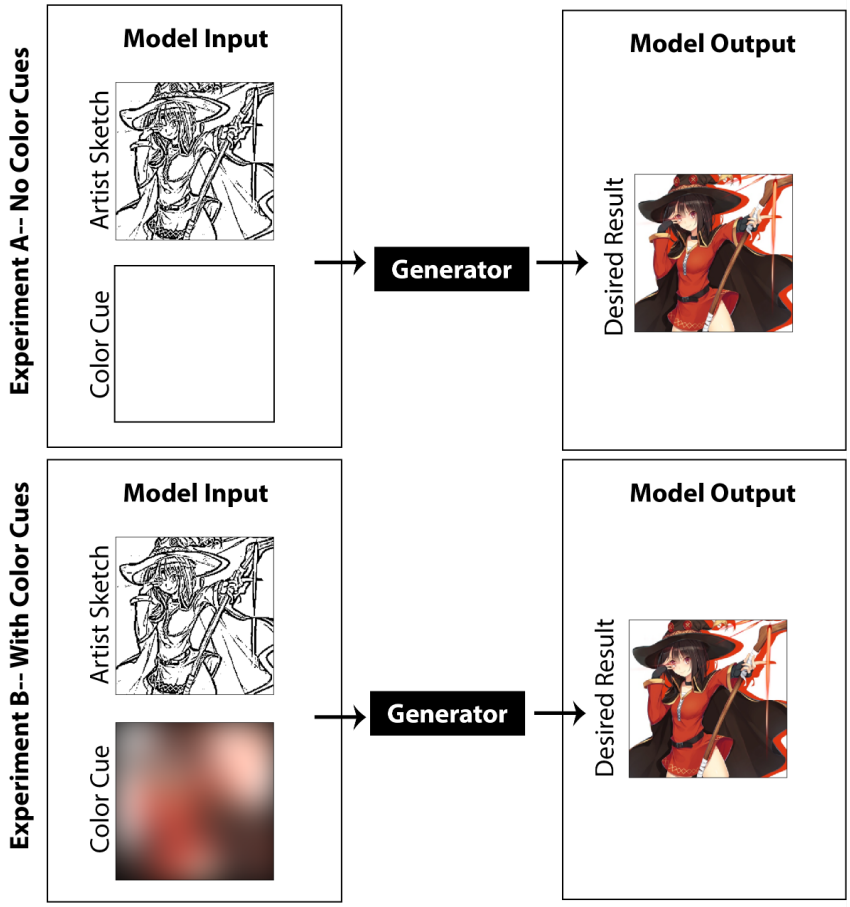

Figure 2: Example of sketchifying a colored image with various blocksize choices used in the thresholding algorithm. In this work, I choose to use a blocksize of 7 to balance edge emphasis and image detailing.

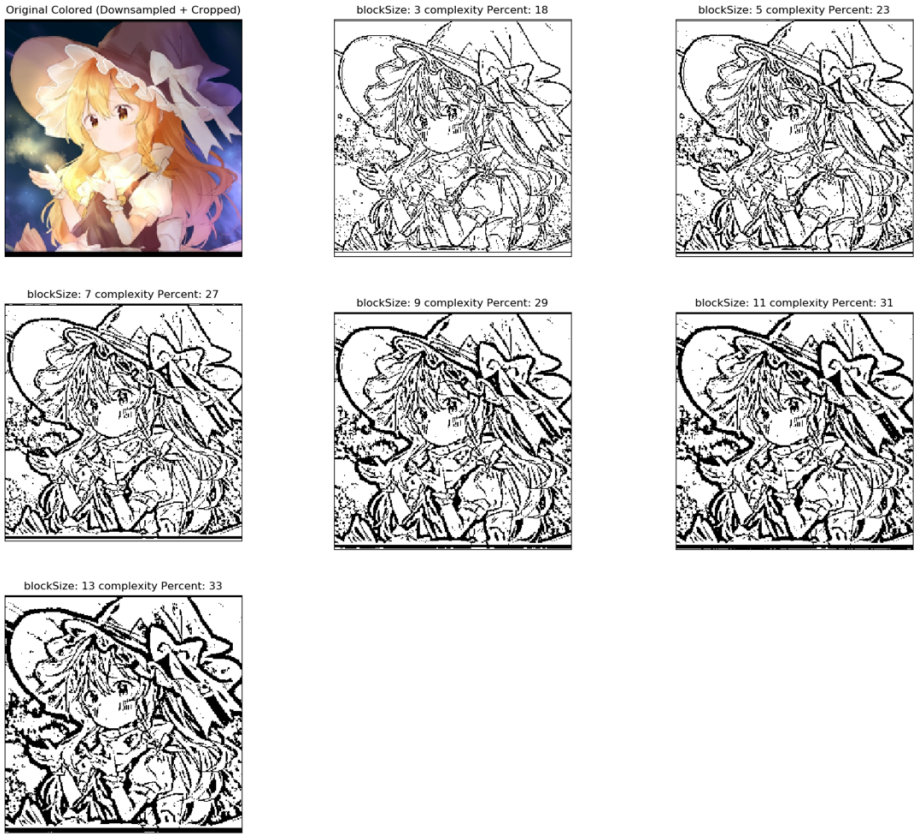

Figure 3: A histogram displaying the distribution of complexity values for sketchified images within one of the tagged image sets. All Images with a complexity value greater than 40 was discarded.

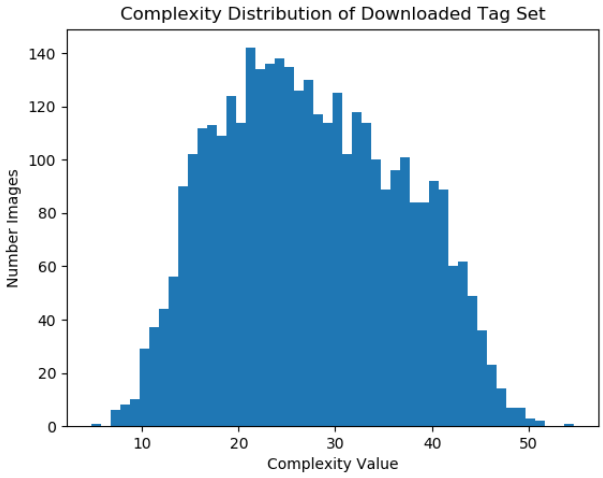

Figure 4: Three examples of the colored and corresponding sketchified images used within the training set.

Figure 5: Three examples of the automatically generated color cues corresponding to a given colored image.

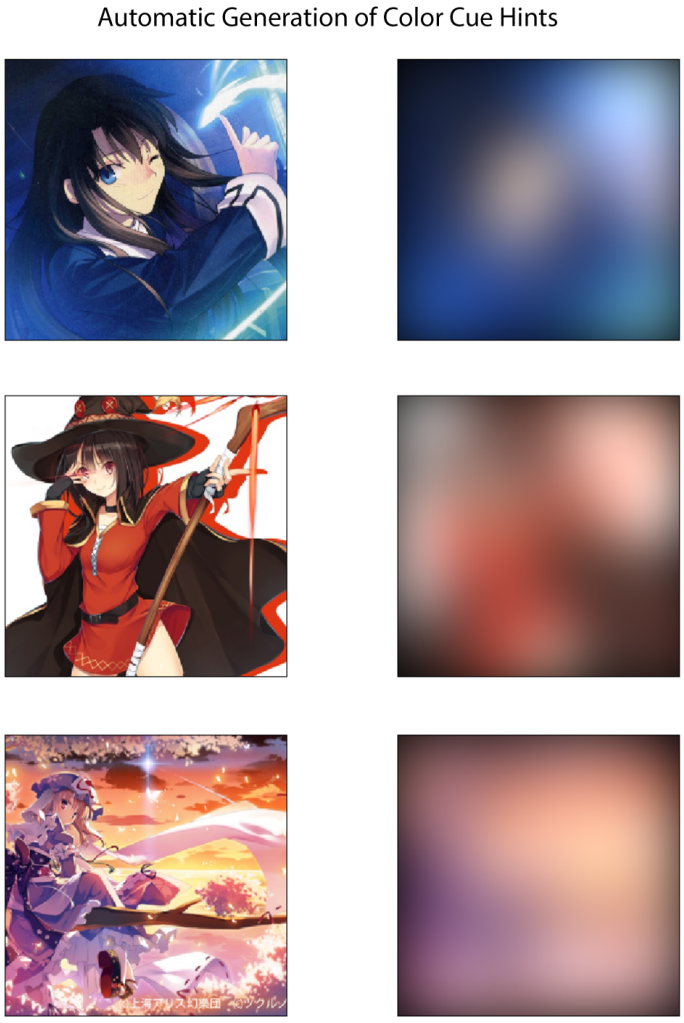

Figure 6: A visual schematic of the generator used in this work

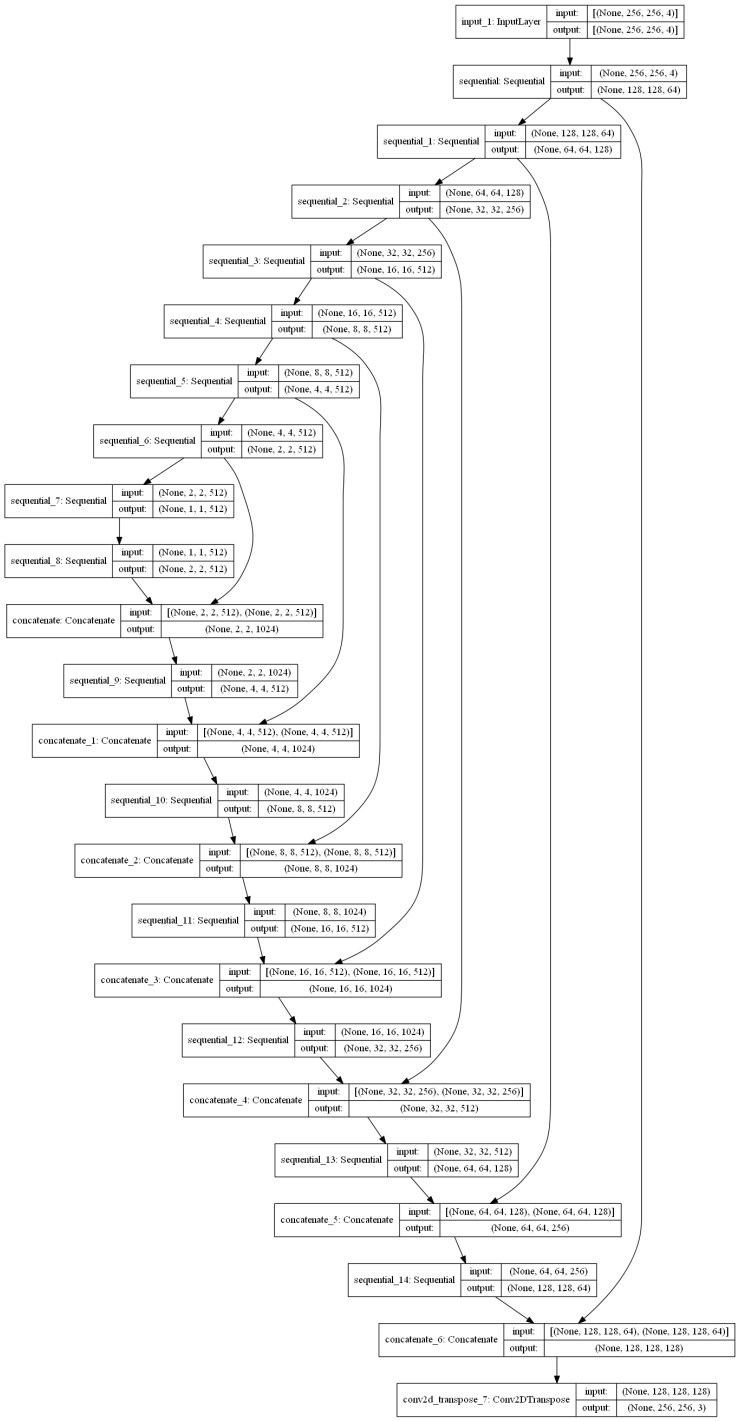

Figure 7: A visual schematic of the discriminator used in this work

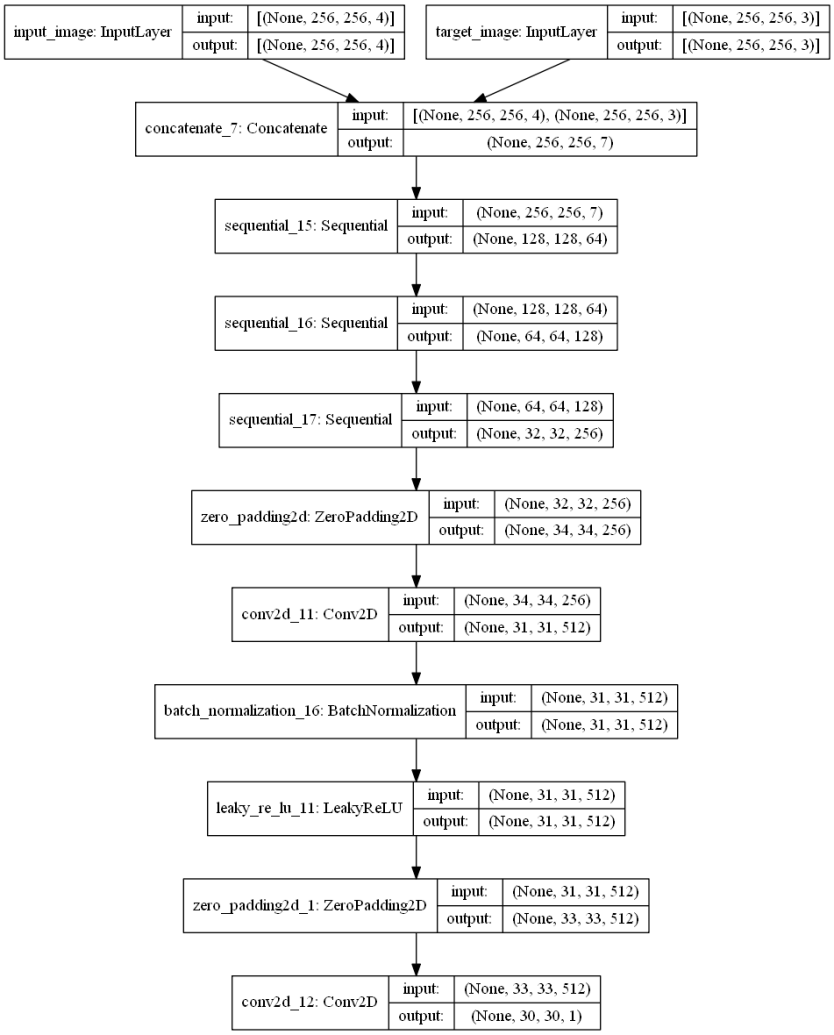

Figure 8: Results of the trained model without color cues for the three testing set challenges. For each set of four figures, the top left displays the real colored image, the top right displays the sketchified image, the bottom left displays the inputted color cues, and the bottom right shows the generator output.

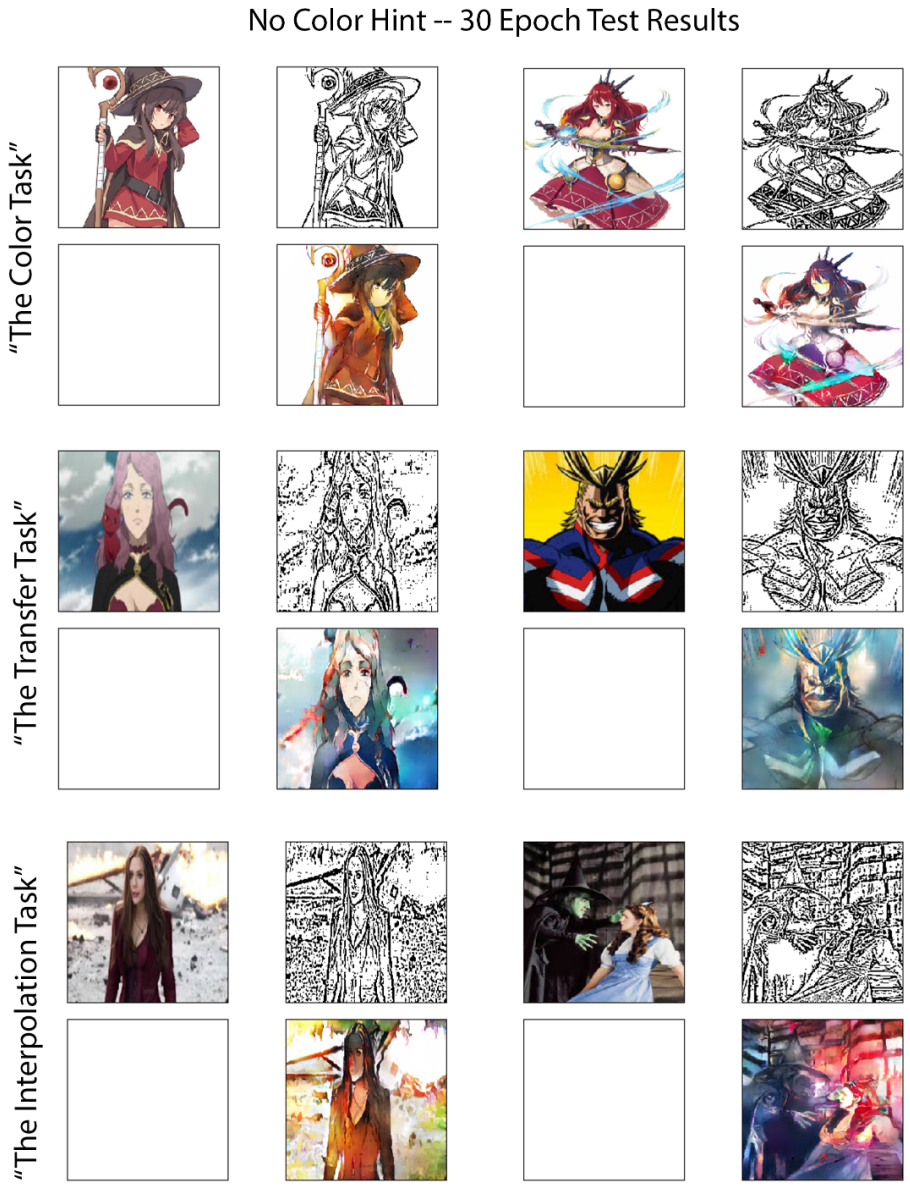

Figure 9: Same as figure 8 but for the experiment with Color Cues provided by the artist

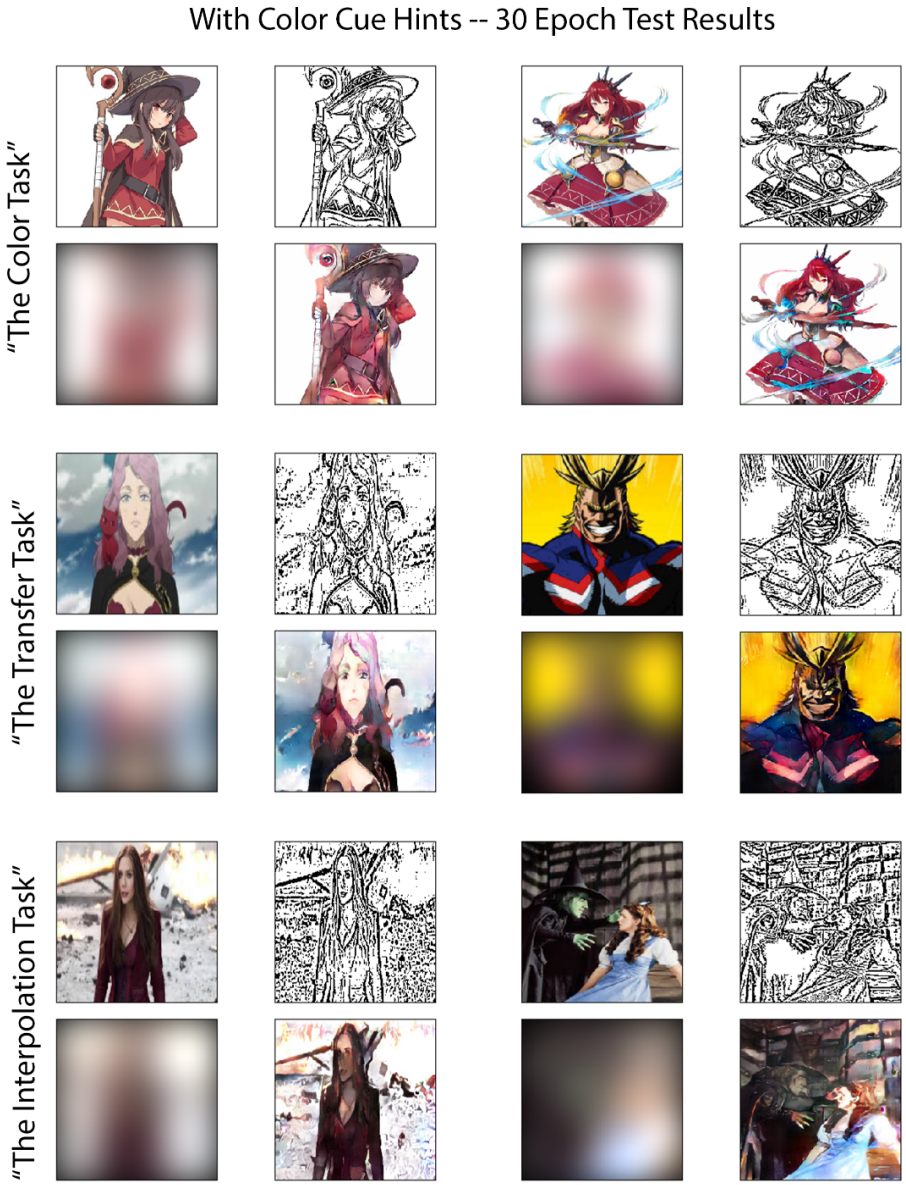

Figure 10: Additional Examples of Color Cue Manga Colorization