Unpaired Manga colorization

Otávio F. Jacobi*†, Théo Matricon*†, Dylan Hertay*†, Julien Mazué*†
* *Equal contributions.*
† *ENSEIRB-MATMECA*
Bordeaux-INP
Bordeaux, France
Email: {ofloresjaco,tmatricon,dhertay,jmazue}@enseirb-matmeca.fr

***Abstract*—Colorization techniques with deep neural networks have shown tremendous improvements over the last few years. We propose to apply image colorization techniques to manga. It is important to note that manga colorization is not simple, given that the grayscale version of the colored manga images have a different color distribution compared to the original black and white manga. The issue is that we could not find any publicly available dataset. This is a major setback. We scrapped a dataset online, however the pairwise mapping is not perfect e.g. translation, different fonts and not suited for classical supervised approaches. We focus on techniques that do not need a pairwise mapping. We first review the results obtained using CycleGAN, then we review the results obtained using CUT.**

1. Introduction

Colorization techniques with deep neural networks have shown tremendous improvements over the last few years [10][12], especially with the rise of applications of Generative Adversarial Networks (GANs) [2]. We propose to use these new techniques to color manga images from the original black and white (B&W) manga to their colored version. This is a different problem than simply colorizing an image from its B&W version. Indeed, when transforming a colored manga image to a B&W image, the images obtained differ from the manga original B&W image. This can be explained because specific techniques are used when producing only B&W images that do not apply to the colored image production.

For the task of manga colorization, we found no public dataset. One approach which was used in [3], a similar work, was to use a single training image at the price of adjusting transformations that were needed to produce a result of good quality. We instead built a web scrapper to download images from websites as a way to get colored images and B&W images. Unfortunately, the images are not a perfect mapping from one space to another, there are differences in the texts, the fonts, they are also not perfectly aligned.

So we decided to focus on techniques that could produce good results even with imperfect pairwise mapping from one state to another. Thus we will review the results we obtained using CycleGAN [13], and the next iteration of the same algorithm CUT[7].

Our source code is available at https://github.com/Theomat/colorization-av-enseirb-2020.

2. Related Works

Generative Adversarial Networks (GANs) [2] In the deep generative models field, one of the main problems has always been to try and generate some data that cannot be easily discerned from true pieces of data. One particular model has emerged in response to this problem, introducing a dual model combining two different models. These two models are called discriminator and generator, and there respective goals are opposites : The generator wants to generate data that the discriminator cannot discriminate from true data, and the discriminator wants to be able to discern better the generated data from the truth.

As such, let us call G the model responsible for generating data from an input z obtained from a random distribution p_z. Let us call D the model responsible for labelling the input x as coming from a reference data-set p_{data} or as being generated by G. On one hand, the discriminator will try to maximize the probability of correctly assigning labels over the mix of true and generated data, while on the other hand the generator will try to maximize the probability for its generated data to not be recognised as such. This second part is done through minimizing the distance between the prediction from the discriminator and 1, that represents the probability of being from the data-set from the discriminator point of view. Let us denote the resulting value function as V(G, D) so that :

$$\min_G \max_D V(D,G) =$$
$$\mathbb{E}_{x\sim p_{data}}[logD(x)] + \mathbb{E}_{z\sim p_z}[log(1-D(G(z)))]$$

3. Problems

3.1. Data Issues

Most colorization techniques work on CIE *Lab* color space instead of the classic RGB color space[10][12], mostly

because it allows for a clear separation of the luminosity (uncolored image) L and the colorful spectrum a and b. These techniques try to learn the mapping $\mathcal{G} : L \rightarrow (a, b)$ where the final image $I = (L, a, b)$ has a satisfactory color.

Because we could not find any publicly available dataset, we created a web scrapper to download manga images and create our own dataset. The web scrapper script allows to download images of different Mangas from different websites, but in this project we mostly work on the manga *One Piece*, given that this one had the most colored images of the original manga. Our dataset presents many challenges: first, the original manga image can be slightly rotated and shifted compared to its colored version. Also, the language of the exclamations and the font of the text can be completely different from the original and its pair, as shown on Figure 1.

Figure 1: From left to right: original Manga image, grayscale version of colored image and colored image. We can see that exclamations in the original manga are in English while on the colored version are in Japanese. The text in the balloons are also different. When comparing pixel-wise, the images are slightly shifted, and for other cases they are also rotated.

All these problems could be solved if we used only the colored manga and its respective grayscale version, however, we would not be able to colorize the original manga, as shown in section 3.3.

3.2. GAN issues

The idea behind GANs is to have a discriminator and a generator. They are adversary, that is to say their reward is the opposite of the other. The goal of the discriminator is to find if an image is from the dataset or if it has been generated by the adversary. The goal of the generator is to fool the discriminator.

Usually this is done using two neural networks, one for the generator and one for the discriminator. One network is trained for a few epochs while the other is frozen during these epochs.

There are many issues with GAN [8]. First, there is mode collapse, that is when the generator slowly but surely converge to produce one particular kind of image or one specific image. For example, on the MNIST dataset, mode collapse could cause the generator to only produce images of 1. Second, as it can be guessed from mode collapse, there is a lot of instability akin to deep reinforcement learning techniques. This instability, as well as mode collapse, are depicted as a consequence of catastrophic forgetting [9], which defines the fact that a GAN may ruin later in the training what had previously been learned. There are some research that are conducted in order to avoid mode collapse and ensure convergence such as a new type of GAN named Wasserstein GAN [1] introducing a new distance function in order to obtain a continuous and differentiable loss function. And third, GAN use two networks, that is they need twice the memory, and are computationally expensive to train.

These issues are to be kept in mind, because they restrain the practical use of GANs. In other words, the choices we made in this work were motivated by trying to mitigate these issues.

3.3. CIELAB issues

The CIE *Lab* correspond to 3 values respectively: L represents the luminosity of the image, a and b are the four unique colors which human being can see (blue, yellow, green and red). The a axis correspond of the scale between green and red color while the b axis correspond of the scale between blue and yellow color. This color space is better when we want to use the image colorization given that the black and white image correspond of the lightness L. Therefore, there is no need to vary the intensity of the color but only to choose the color that will appear most natural to the human eye.

Unfortunately, even if we had a perfectly pixel-wise matched dataset, because the color distribution between the original manga and the gray scale version of the colored one is so different(Figure 2), this approach cannot be used. We can simulate best possible results of this approach by using the original manga L channel and the colored version a and b channels. For this example, we manually found one (rare) example where exclamations and texts were exactly the same and also manually adjusted rotation and shift of these images, in order to have the best possible simulation, as shown in Figure 3, combining the L component of original manga and the a and b components of the colored version does not generate very good perceptual results.

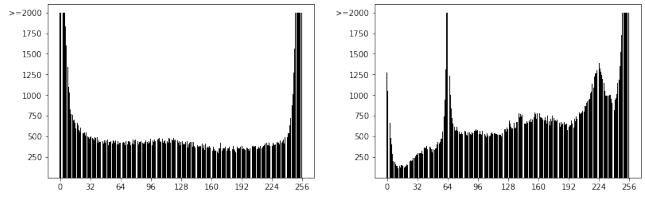

Figure 2: Original Manga's color distribution and gray scaled colored version distribution. Both images were manually re-scaled/rotated to be perfectly matched pixel-wise.

Thus, even if we had a function $\mathcal{G}$ that could map (a, b) from L, results would look like the first image in Figure 3.

Because of all these issues we decided to try unpaired image translation techniques, where we try to learn how to colorize the images without having perfectly matched/paired data.

Figure 3: Original Manga's L colorized with a and b of colored image and ground truth colored image.

4. CycleGAN

This work was developed in [13]. Their objective is to work with an unpaired dataset. Having a dataset in space X and a dataset in space Y, they want to learn a mapping $F : X \to Y$. Both dataset are not paired, that is to say there is no local match from X to Y or from Y to X.

Thus the key idea is to introduce a cycle-consistency loss on top of adversary loss on domains X and Y. That is considering the opposite transformation $G : Y \to X$, this loss aims to ensure that:

$$\begin{aligned} \forall x \in X, x \approx G(F(x)) \\ \forall y \in Y, y \approx F(G(y)) \end{aligned} \tag{1}$$

Formally, we have sets $\{x_i\}_{i=1}^N$ and $\{y_j\}_{j=1}^M$ with respective distributions $x \sim p_{data}(x)$ and $y \sim p_{data}(y)$. Four networks will be trained. One network will be the transformation F, and another will be the transformation G, the two others are discriminators respectively for the sets X and Y. The discriminator D_X aims to distinguish the x from the $G(y)$, and the discriminator D_Y aims to distinguish the y from the $F(x)$.

The adversarial objective for the mapping G and its discriminator D_Y is written as:

$$\begin{aligned} \mathcal{L}_{GAN}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{data}(y)}[log D_Y(y)] + \\ \mathbb{E}_{x \sim p_{data}(x)}[log(1 - D_Y(G(x)))] \end{aligned} \tag{2}$$

G aims to maximize this objective, and the discriminator D_Y aims to maximize, that is to say $\min_G \max_{D_Y} \mathcal{L}_{GAN}(G, D_Y, X, Y)$. The same objective is introduced for F and D_X.

In theory, the adversarial objective (2) is enough to produce mappings G and F that outputs identically distributed data. But, it also means given enough neurons and layers, the learned mappings can represent any permutation of x_i to y_i. Thus the adversarial loss is not enough to learn a specific mapping, that is it is not enough to guarantee the cycle consistency.

The idea of cycle consistency loss (1) can be translated in terms of:

$$\begin{aligned} \mathcal{L}_{cyc}(G, F) = \mathbb{E}_{x \sim p_{data}(x)}[\|F(G(x)) - x\|_1] + \\ \mathbb{E}_{y \sim p_{data}(y)}[\|G(F(y)) - y\|_1] \end{aligned} \tag{3}$$

The idea is clear, the expected L1 norm should be as small as possible as to guarantee cycle-consistency from domains X and from domain Y. This is the same idea, as if $G \circ F : X \to X$ and $F \circ G : Y \to Y$ are trained as autoencoders.

Merging both losses, gives the full objective:

$$\begin{aligned} \mathcal{L}(G, F, D_x, D_y) = \mathcal{L}_{GAN}(G, D_Y, X, Y) + \\ \mathcal{L}_{GAN}(F, D_X, X, Y) + \\ \lambda \mathcal{L}_{cyc}(G, F) \end{aligned} \tag{4}$$

where λ represents the relative importance of the two types of objectives. The solution wanted to (4) is:

$$G^*, F^* = \arg \min_{G,F} \max_{D_x, D_Y} \mathcal{L}(G, F, D_x, D_y).$$

4.1. Empirical Results

λ is set to 10 in (4), a batch size of 1 image is sued with the Adam solver [4]. They trained once on 128×128 images and then on 256×256 images.

With ablation studies both the adversarial objective (2) and the cycle-consistency loss (3) are required to obtain full results. Using only one part of the cycle-consistency loss that is only the X domain part or only the loss for the Y domain part tends to produce instability and mode collapse.

The authors report little success when geometric changes are needed. Fortunately, manga coloration does not need any geometric transformation. In the examples shown, they typically have $N \approx M \approx 1000$.

5. CUT

Contrastive Unpaired Translation has been developed in [7]. This is another method to solve image translation. The aim is to keep the structure of the original image domain but use the appearance of the second image domain. The authors argue that cycle-consistency is too restrictive because it assumes that $G \circ F$ and $F \circ G$ are bijections. An alternative is proposed to maintain the correspondence between the two domains. A direct advantage of not using cycle-consistency is that we can only train two networks instead of four like in CycleGAN.

The core idea is to add a patch-wise contrastive loss. That is patches of the original image are taken, and the idea is that the corresponding patch in the output image should maximize the mutual information between the two patches compared to other random patches in the original image.

The idea of contrastive learning is to learn to associate a "query" to its "positive" example, the other examples are thus the "negatives". That is $\mathbf{v}, \mathbf{v}^+ \in \mathbb{R}^K$ are respectively the "query" and the "positive" while $\mathbf{v}^- \in \mathbb{R}^{N \times K}$ is the matrix of the N negatives, that is for a mapping to K dimensional

vectors. $\mathbf{v}_n^- \in \mathbb{R}^K$ denotes the n-th negative example. The cross entropy loss for such a problem is defined as:

$$l(\mathbf{v}, \mathbf{v}^+, \mathbf{v}^-) = -\log\left[\frac{p^+}{p^+ + \sum_{n=1}^{N}\exp(\mathbf{v}^\top \mathbf{v}_n^-/\tau)}\right] \quad (5)$$
$$\text{where } p^+ = \exp(\mathbf{v}^\top \mathbf{v}^+/\tau)$$

which is the probability of the positive example being selected over the negatives. Now, we will explain how this is used in the CUT loss. First, note that we want correspondence both at the pixel level and at a more larger scale, the color gradient of a shirt for example is important and at the same time the buttons of the shirt are also necessary. This leads us towards a multi-layer, patch-based objective. Our mapping G is made of an encoder G_{enc} and a decoder G_{dec} with $G = G_{enc} \circ G_{dec}$. Note that for the encoder part, we can know which part of the image was used to build the feature after layer l of the encoder. That is each feature can be associated to a patch of the original image. Now, for a layer $l \in [|1; L|]$ of G_{enc} we can train a two-layer MLP H_l and compute the features denoted $\{\mathbf{z}_l\}_L = \{H_l(G_{enc}^l(\mathbf{x}))\}_L$. We denote $s \in [|1; S_l|]$ where S_l is the number of spatial locations layer l. We can then introduce the "positive" $\mathbf{z}_l^s \in \mathbb{R}^{C_l}$ and the "negatives" as $\mathbf{z}_l^{S\setminus s} \in \mathbb{R}^{(S_l-1)\times C_l}$ where C_l is the number of channels of layer l. A similar process can be used for the translated image, that is we introduce the $\{\hat{\mathbf{z}}_l\}_L = \{H_l(G_{enc}^l(G(\mathbf{x})))\}_L$. Now, we can introduce the *PatchNCE* loss that computes this cross entropy loss patch-wise for each image from (5):

$$\mathcal{L}_{PatchNCE}(G, H, X) = \mathbb{E}_{\mathbf{x}\sim X}\Big[\sum_{l=1}^{L}\sum_{s=1}^{S_l} l(\hat{\mathbf{z}}_l^s, \mathbf{z}_l^s, \mathbf{z}_l^{S\setminus s})\Big]. \quad (6)$$

The final objective is a combination of (2), (6):

$$\begin{aligned}\mathcal{L}_{CUT}(G, H, D, X, Y) = \mathcal{L}_{GAN}(G, D, X, Y) \\ + \lambda_X \mathcal{L}_{PatchNCE}(G, H, X) \\ + \lambda_Y \mathcal{L}_{PatchNCE}(G, H, Y).\end{aligned} \quad (7)$$

In practice the $\lambda_X = 1$ with identity loss, $\lambda_Y = 1$, is named CUT and is the one we will be using. Another version without identity loss, $\lambda_Y = 0$, and with $\lambda_X = 10$ is named FastCUT.

6. Experimental Results

6.1. Dataset

As mentioned previously, our dataset is generated by ourselves, and we mainly try two versions: first, learning on resized images of size 256x256 and secondly on 512x512 images. All the tests were done using a single Tesla T4 16GB GPU and although training time may change between different approaches, it usually takes around 4 to 8 days of training.

6.2. Quality Evaluation

Because of the adversarial nature of GANs, the loss function of both the generators and discriminators does not reveal much about the quality of results. Therefore, this project defines 3 different metrics to evaluate the quality of results over the course of many epochs: PSNR, SSIM and Color Histogram Difference.

Peak signal-to-noise ratio (PSNR) is a standard metric in image processing for comparing similarity between two images, usually used to measure the quality of reconstruction of an image. Since in our context the reference image can have expressive differences such as text, exclamations... from the predicted image, PSNR does not fit perfectly as as measure. Nevertheless, it is still a way to compare similarity between two images, and therefore we use it as one of our metrics, having as goal to generate images with higher PSNR value when compared to the reference image.

The PSNR metric represents the ratio between the maximum possible power of a signal, the image, and the power of its corrupting noisy signal. Given the reference RGB colored image y and the GAN RGB colored image $\hat{y}$, the PSNR between them can be expressed mathematically as:

$$PSNR(y, \hat{y}) = 10\log_{10}(255^2/MSE(y, \hat{y})) \quad (8)$$

where

$$MSE(y, \hat{y}) = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}(y_{ij} - \hat{y}_{ij})^2 \quad (9)$$

Structural similarity index measure (SSIM) [11] is a perception-based model and differently than the PSNR metric, it incorporates many aspects of the overall image such as contrast and luminance instead of a simple pixel-wise absolute error metric, which makes the SSIM metric much more interesting for our problem.

SSIM has three components: luminance (l (10)), contrast (c (11)) and structure (s (12)) which are computed independently in different windows or patches of an image. For each patch x and z of the images y and $\hat{y}$ we define the components as:

$$l(x, z) = \frac{2\mu_x\mu_z + c_1}{\mu_x^2 + \mu_z^2 + c_1} \quad (10)$$

$$c(x, z) = \frac{2\sigma_x\sigma_z + c_2}{\sigma_x^2 + \sigma_z^2 + c_2} \quad (11)$$

$$s(x, z) = \frac{\sigma_{xz} + c_3}{\sigma_x\sigma_z + c_3} \quad (12)$$

where

- μ_x and μ_z are the average of patches x and z
- σ_x and σ_z are the standard deviation of x and z

- σ_{xz} is the covariance of x and z
- $c1, c2$ and $c3$ are variables to stabilize the division

Finally, the SSIM value is calculated as a weighted combination of these components, as shown in (13).

$$SSIM(x,z) = l(x,z)^{\alpha} \cdot c(x,z)^{\beta} \cdot s(x,z)^{\gamma} \tag{13}$$

In our case, because we use RGB images, SSIM is calculated independently for each channel and then averaged.

Histogram difference is a simple measure that compares the color distribution of two images. This metric is very interesting for our problem, given that we aim to keep the structure of the images and what we want to learn is how to properly colorize it.

There are many measures to compare two histograms, we empirically chose Bhattacharyya distance (14), as it is designed to measure the similarity between two probability distributions. Given histograms H_1 and H_2, the Bhattacharyya histogram difference can be calculated as:

$$d(H_1, H_2) = \sqrt{1 - \frac{1}{\sqrt{\bar{H}_1 \bar{H}_2 N^2}} \sum_I \sqrt{H_1(I) \cdot H_2(I)}} \tag{14}$$

where $\bar{H}_1$ and $\bar{H}_2$ are the mean values respectively for histograms 1 and 2, N is the number of histograms bins, and $H_i(I)$ is the value of the bin I for histogram i.

6.3. Training results

In order to evaluate both CycleGAN and CUT, we trained both networks from scratch and evaluated the results on a small test set we developed. We used the parameters advised by the original authors please refer to their work [13], [7] for details. We tried to change a few hyperparameters, however these configurations consistently performed worse. Training these networks can be very tricky and time consuming given that the network is learning without any information: the network does not even know that it is a colorization task.

Analysing the results displayed on Figure 4 and Figure 5 we can see that our metrics improve consistently with the CUT method whereas there is little improvement if any with the CycleGAN method. The last image is one example of colorization with the last model of each method. Perceptually, we can point out that CUT results are much better than CycleGAN's results, which shows correlation between human perception and the results given by the metrics we chose. Because the CUT method performed much better than CycleGAN in our tests, most of the results displayed in the next section were generated with the CUT technique.

Because the loss computation is different on CycleGAN and CUT, and because the loss usually does not say much about the quality of the final image in GAN scenarios, these values are considered irrelevant, and instead, we use the metrics we previously defined.

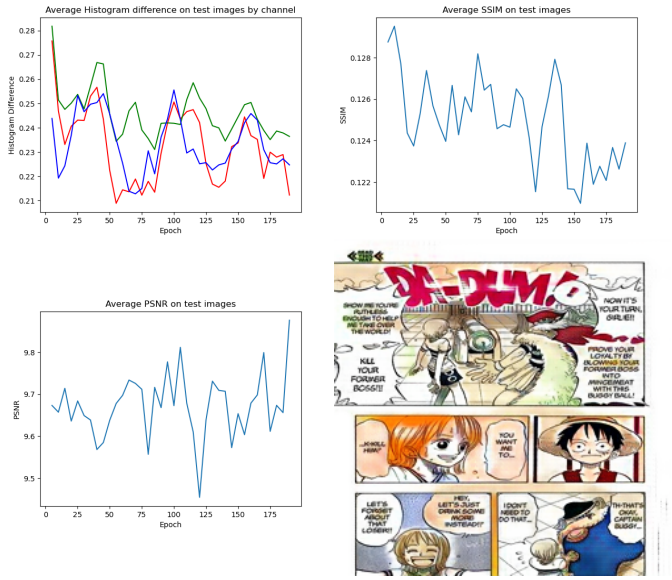

Figure 4: CycleGAN training results on 256x256 test images. The last image is one example of colorization done by the network.

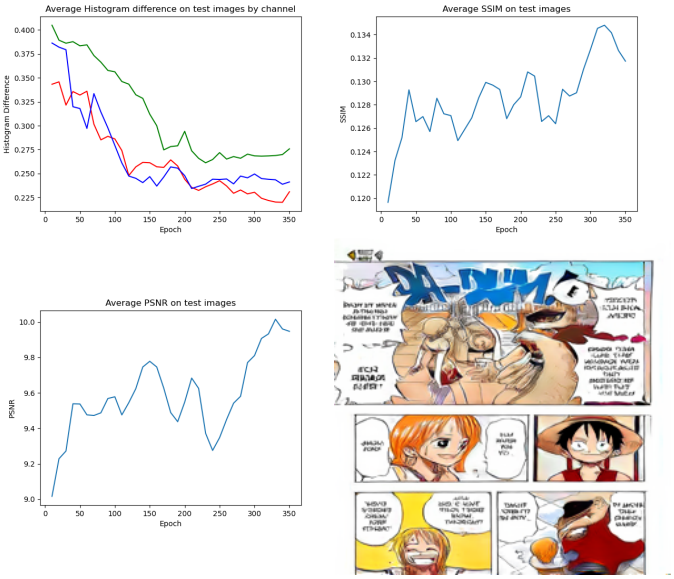

Figure 5: CUT training results on 256x256 test images. The last image is one example of colorization done by the network.

6.4. Results on 256x256

The first attempts were made on a 256x256 pixel image dataset. This dataset consisted of unpaired data of 5674 original manga and colored images each. Image 6 shows first test results using the CUT technique. Image 6 is a great example for showing the advantages and disadvantages of

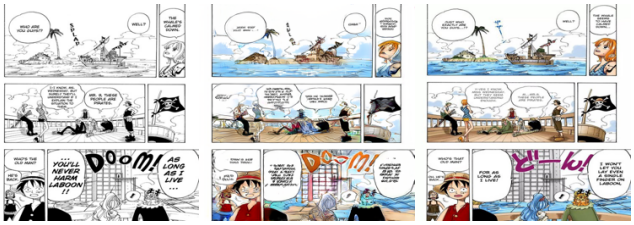

Figure 6: From left to right: original Manga image, ours and ground truth.

our approach. As advantages, we note that our generator G learned to recognize and colorize aspects of the world such as the proper color of water and correct colorization of clouds and also characters: the correct skin, hair and accessories colors are usually used.

More interesting than that, we notice that our network learned to recognize the manga "exclamations" , the name we chose for big capital letters like "DOOM" on Image 6, and color them. The impressive part is that in our original manga these expressions are done with English alphabet, while in the colored version, they are in Japanese characters, nevertheless, the network learned to recognize it and color them accordingly.

As disadvantages of our approach we can see that because the text inside the balloons are different sometimes only by font, the generator tries to re-write something similar to what it is shown in the colored version, usually messing up the text and turning it into unreadable text.

Figure 7: From left to right: original Manga image, ours and ground truth.

There are many cases where the network does not see enough of the same character and end-up colorizing it in some different way then the original colorization, as for example in Figure 7 with the character in the middle panel.

6.5. Results on 512x512

After our first attempt with colorizing 256x256 images, we tried to move to 512x512 images, mainly because of readability issues for 256x256, unfortunately, the problem on 512x512 images is much harder.

The first issue with 512x512 is that memory consumption is much larger, and therefore, we had to reduce our dataset size, otherwise training would have taken over three weeks. Secondly, the networks in both techniques were designed to operate on 256x256 or smaller images, and therefore, the architecture seems to be unable to properly fit on 512x512 images.

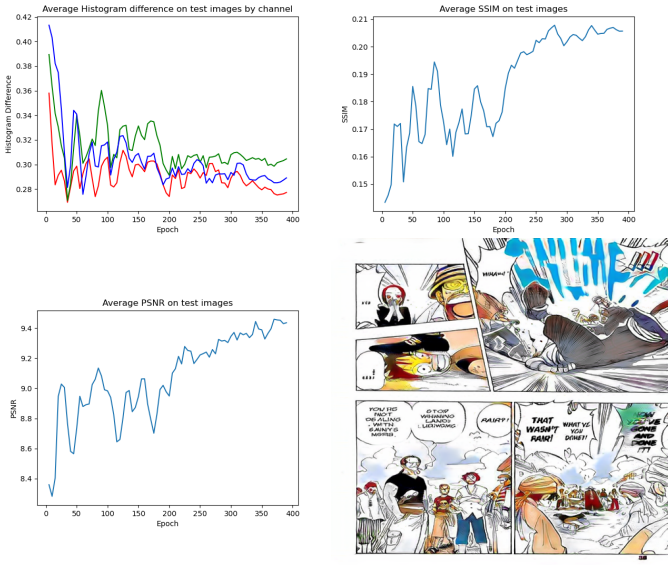

Figure 8: CUT training results on 512x512 test images. The last image is one example of colorization done by the network.

Even though on Figure 8 we can see that our metrics improve over time and this is reflected in the generated image, the final image is never, visually as good as the results obtained in 512x512 images. We consider that a bigger neural network, adapted for 512x512 images, and even more training time could be a solution for colorization of 512x512 images.

6.6. Progression of results

Both techniques need many training epochs to produce results. Figure 12 in Appendix A shows the evolution of the CUT Generator Network over the course of many epochs. Initially the generator network does not know anything about the world and its characters, so only using random colors is enough to fool the discriminator. But as epochs go by, the network learns the characters and the environment. One interesting example is on the third image, our network learned to color the grass as green, even though that part is not colored in the colored version.

6.7. Super sampling 256x256 results

Since the results on images of size 256x256 are promising, we wondered if using deep super sampling techniques enabled us to get better results for images of size 512x512. We used three different techniques to scale up our images, namely bicubic as a reference, enhanced deep super-resolution network (EDSR) [6], and Laplacian Pyramid

Super-Resolution Network [5]. We will be using the models for a scale factor of 2 referred in the OpenCV documentation.

On Figure 13 in Appendix B, you can see a comparison between the generated 512x512 image and different upsampling schemes used. It might be a bit hard to tell the difference at first sight, but when looking at the left middle panel, the character's face is more blurry with bicubic than with LapSRN, and LapSRN is more blurry than EDSR. Visually, it seems that EDSR performs best among the upsampling techniques. CUT on 512x512 produce images of higher quality however the colouring is clearly not up to par with stains of colors as in the cape of the character in the bottom right panel.

Since in the last section, we found that the metrics we chose are a good indicator of the performance of the model to the human eye. We took fives images generated by CUT at epochs 5, 50, 100, 300, 400, upsampled them with the different techniques and plotted the metrics on Figures 9, 10, 11.

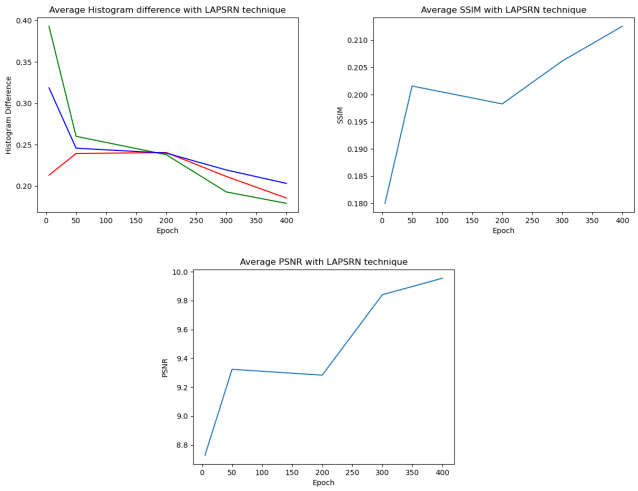

Figure 10: Metrics for LapSRN

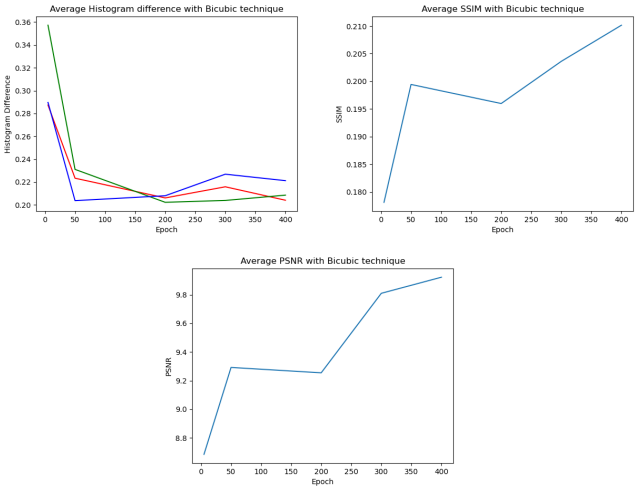

Figure 9: Metrics for bicubic interpolation

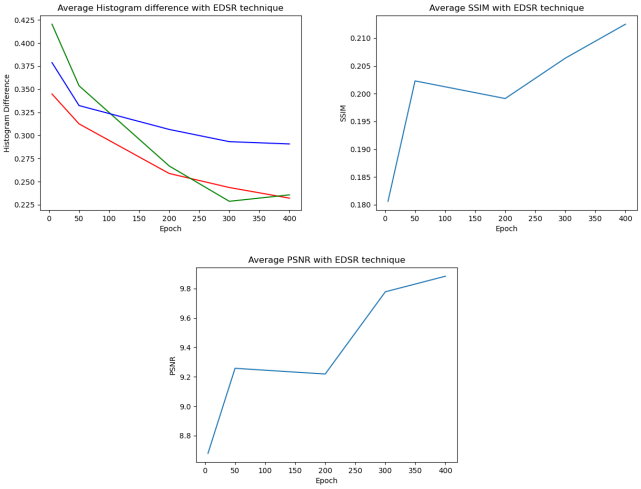

Figure 11: Metrics for EDSR

It can be seen that the PSNR is approximately the same for the three techniques, it is also close to the original PSNR of the 256x256 images and bit above the PSNR for the 512x512 CUT images. The same phenomenon occurs for the SSIM, except of course the comparison with the 256x256 image. This clearly explains in part the absence of obvious differences between the three techniques.

The difference is more apparent on the histogram difference. The average histogram difference is 10% lower for bicubic and LapSRN which should entails a better colouring than in EDSR yet this is not apparent to the human eye.

We believe the subtle difference is not important enough to consider using generic advanced deep sampling methods. Considering it took us 1min to upsample an image using EDSR, 3s for LapSRN and less than a second for bicubic interpolation, the computing cost is not worth it.

Perhaps using deep super sampling models specifically trained for manga could get us better performance, especially for the text that stays unreadable after upsampling.

7. Discussion

Both techniques present very interesting methods. Even though our results are far from enough to, for example, automatically colorize new manga as they are published, there are many interesting points in our examples.

The first thing that caught our attention was that it learned to colorize exclamations in English by only looking at exclamations in Japanese. Secondly, the network learned to colorize characters that it never saw or saw just a couple of times before, even though it uses different colors, and the colorization is most of the time acceptable.

Besides that, the network learned to recognize main characters and colorize them with the original author's color and also how to colorize the environment such as clouds,

sea, trees as shown on Figure 14 in Appendix C. One clear example is the main character's hat, that has a red stride, which is almost always correctly colorized.

8. Conclusion

We have managed to color manga images using different techniques. CycleGAN proved to not be enough and to be consuming a lot of computing resources. While CUT got us good results for images of size 256x256. The resolution is too low for the text to be readable it is clear that the generator learned to colorize elements of the manga accordingly such as grass, or some of the main characters. On some pages, the generator could probably fool a human that has never seen the manga. However, we tried to generate 512x512 images, but the results were poorer and not on par with the lower resolution. Perhaps an extensive search of hyperparameters tuning could lead to CUT successfully generating 512x512 images, since in the original paper the authors focused on 256x256 images.

As a way to circumvent the problem, we tried upsampling the 256x256 images with different techniques. The different techniques offered close results to one another, and did not improve the quality of the images. We believe the use of models trained specifically for manga upsampling is necessary to obtain improved results.

Overall, most of us were not familiar with the concepts used in colorization. While we were familiar with GANs, we discovered the techniques used in CycleGANS and CUT for the specific case of unpaired data. On one hand, we only used the architectures provided by the original authors, it was already a complex task to make these techniques work on our data. On the other hand, to try more exotic configurations more time was needed. We found that the use of metrics to quantify the results is a hard problem. Even in our case, while it seems the metrics agree with visual perception, for the case of upsampling, it seems that the metrics would give an edge to bicubic producing the best results and EDSR the worst while it is the opposite. As such it seems rather complex to scientifically estimate the results of such techniques without requiring an empirical review by human eyes. In the end, it was a great learning experience for all of us.

References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan, 2017.

[2] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[3] Paulina Hensman and Kiyoharu Aizawa. cgan-based manga colorization using a single training image, 2017.

[4] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[5] Wei-Sheng Lai, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsuan Yang. Deep laplacian pyramid networks for fast and accurate super-resolution. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[6] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 136–144, 2017.

[7] Taesung Park, Alexei A. Efros, Richard Zhang, and Jun-Yan Zhu. Contrastive learning for unpaired image-to-image translation. In *European Conference on Computer Vision*, 2020.

[8] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans, 2016.

[9] Hoang Thanh-Tung and Truyen Tran. On catastrophic forgetting and mode collapse in generative adversarial networks, 2020.

[10] Patricia Vitoria, Lara Raad, and Coloma Ballester. Chromagan: Adversarial picture colorization with semantic class distribution. In *The IEEE Winter Conference on Applications of Computer Vision*, 2020.

[11] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 2004.

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

[13] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

Appendix A.
Evolution of results for CUT training over epochs

Figure 12: Example of CUT method results over different epochs on 256x256 test images.

Appendix B.
Results with upsampling

(a) 512x512 CUT image

(b) upsample with bicubic

(c) upsample with LapSRN

(d) upsample with EDSR

Figure 13: Comparison of images obtained with different upsampling techniques and the CUT 512x512 image

Appendix C.
Additional 256x256 results

Figure 14: More examples of GAN colored 256x256 images.