Semi-automatic Manga Colorization Using Conditional Adversarial Networks

Maksim Golyadkin and Ilya Makarov(✉)

HSE University, Moscow, Russia
myugolyadkin@edu.hse.ru, iamakarov@hse.ru

Abstract. Manga colorization is time-consuming and hard to automate. In this paper, we propose a conditional adversarial deep learning approach for semi-automatic manga images colorization. The system directly maps a tuple of grayscale manga page image and sparse color hint constructed by the user to an output colorization. High-quality colorization can be obtained in a fully automated way, and color hints allow users to revise the colorization of every panel independently. We collect a dataset of manually colorized and grayscale manga images for training and evaluation. To perform supervised learning, we construct synthesized monochrome images from colorized. Furthermore, we suggest a few steps to reduce the domain gap between synthetic and real data. Their influence is evaluated both quantitatively and qualitatively. Our method can achieve even better results by fine-tuning with a small number of grayscale manga images of a new style. The code is available at github.com.

Keywords: Generative adversarial networks · Manga colorization · Interactive colorization

1 Introduction

In the last decade, due to the growth of computing power and the amount of data available, the area of machine learning - deep learning - has experienced a great increase. There have been considerable successes in the ability of computers to understand data of all formats: images, speech, texts.

The success of deep learning models in various fields of science motivates apply them to image colorization task. Over the last few years, many methods using Convolutional Neural Networks (CNNs) [15] and Generative Adversarial Networks (GANs) [8] for the photograph and drawing colorization have been proposed. They exhibited promising results proving the ability of neural networks to colorize images.

In this paper, we consider the task of colorizing the black and white manga. Usually, the manga is created with a pen and ink on white paper. To make an existing manga more attractive, there is a great demand for its coloring. It is done manually using applications such as Photoshop or Autodesk, which is a time-consuming task, so finding an easy way to create realistic coloring for black and white drawings is an urgent task.

The key difference between manga and black and white photographs is that in many cases manga is almost a binary image (some areas may be flooded with a homogeneous grey) and the photographs are grayscale images. Grayscale images contain more information that limits the number of possible colorings (light areas can't be colored in a dark color) and simplifies the implicit solution of the segmentation problem during colorization. Besides, the manga page consists of several panels containing different scenes and it is a natural requirement that their coloring should be homogeneous.

We use deep conditional GAN to perform high-quality end-to-end manga page colorization. Employing conditional input allows the user to manually manipulate coloring for achieving a desirable result.

Since there are no publicly available datasets consisting of image pairs required for supervised learning, we utilize synthetic data. This creates a domain gap between the objects of the training set and manga images. To significantly reduce it, we implement some techniques that do not require auxiliary models.

2 Related Work

Most early methods using CNNs [4,10,14,21] are based on training with a large number of images to map a grayscale image to a color one. Neural networks can learn how to use the information contained in low-level and high-level features to perform colorization. These approaches are fully-automatic and don't require any human involvement. However, since in the presented works the training relies on minimizing the distance between the values of color image pixels and neural network output (MSE or MAE), the colors of the produced image are faded and unnatural. To solve this problem, GANs are used, namely models based on Pix2Pix [11] architecture. The main idea is to add the adversarial loss to the main loss function and use the main neural network as a generator while some additional neural network is used as a discriminator. This approach leads to more plausible and colorful images.

All mentioned approaches have one common drawback: colorization is an ill-posed task. There are several suitable colorings for the majority of black and white images [3] and fully-automatic approaches can obtain only one. Hence, if some object has several suitable colors, the neural network will produce a value close to the average of these colors that leads to degradation of the results. The solution is to use some human-generated hints. These hints can be, for example, sparse images with some pixels labeled with the desired color (color hint) [22] or set of images that provides the requested color distribution (reference images) [9].

In addition, this paper relies on approaches designed for drawing colorization using color hints [5], reference images [17], both of these approaches [7], and text descriptions [13].

3 Method

3.1 Data Preprocessing

Edge Detection. There is no dataset containing pairs of black and white and color manga images, so we had manually colorized images only. Building our dataset by matching pages of black and white and color editions was problematic since corresponding images can only be obtained from different sources, therefore have different sizes and arbitrarily different margins, thus hindering the pixel-wise correspondence essential for training Pix2Pix-like architecture. For this reason, we decided to use synthetic data obtained with an edge detector

Classic edge detectors such as Canny and Sobel operator were found to be unable to obtain an image similar to the black and white manga while retaining the distinguishable text, so we decided to use XDoG [18] algorithm that produces convincing images while retains fine details and text (see Fig. 1). A large number of parameters and the dependence of line thickness on the image size can be considered among the disadvantages of this method.

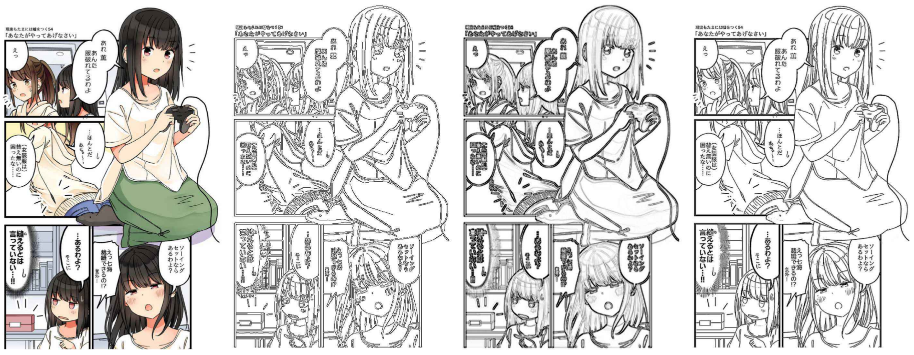
現実もたまには嘘をつく54
「あなたがやってあげなさい」
あれ 薫 あんた服破れてるわよ
えっ
…ほんとだ あちゃ~…
(女装服は)替え無いのに困ったな……
ソーイングセットならあるわよ?
えっ七海裁縫できるの!? 意外…
…あるわよ? そこに
縫えるとは言っていない…!!

Fig. 1. Comparison of edge detection algorithms: original image, Canny, Sobel, XDoG.

Denoising. On the one hand, since XDoG partially ignores noise, mapping an image without noise to a noisy one can destabilize the learning process. On the other hand, usage of a noisy input image can result in poor colorization since XDoG ignores noise just partially. Therefore, it seems reasonable to use some method that will remove noise while reducing image quality insignificantly.

The neural network model FFDNet [20] was chosen as such a method. This is a convolutional neural network trained to remove additive Gaussian noise via minimization of the mean square error between the pixel values of some image and the output of the network when the input was artificially corrupted with noise.

We used a network that was trained with photographs, but network's high generalization ability enables it to be used with drawings as well.

3.2 Model Overview

We adopt the system proposed in [5] for line art anime drawings colorization with color hints. This work shows the ability of conditional GANs to perform high-quality colorization of drawings being trained on synthetic data generated with XDoG.

Our system takes three objects at the input (see Fig. 2). The first object is a binary, sparse image $x \in \mathbb{R}^{H \times W \times 1}$ generated by XDoG. The second one is the distance field map $d \in \mathbb{R}^{H \times W \times 1}$ as proposed at [17]. It simplifies the learning process since sparse input and color output have different nature and some intermediate representation helps to establish matching.

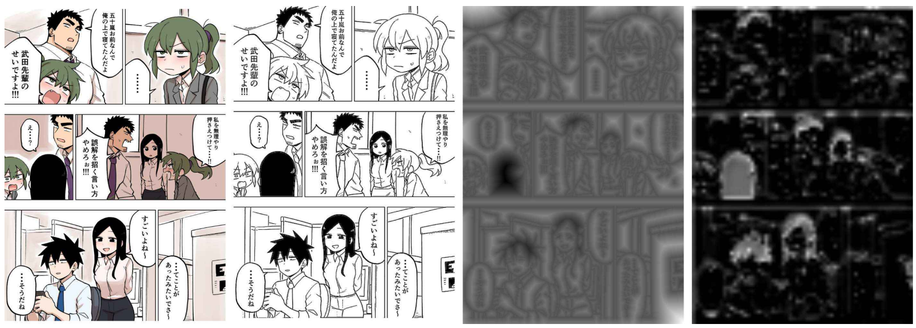
五十嵐お前なんで俺の上で寝てたんだよ
武田先輩のせいですよ!!!
……
私を無理やり押さえつけて…!!
誤解を招く言い方やめろぉ!!!
え…?
すごいよね～
…てことがあったみたいでさ～
…そうだね

Fig. 2. Different image representations: original, XDoG, DFM, local features map.

The third input is color hint represented with $h \in \mathbb{R}^{H \times W \times 4}$. During training, it's generated by randomly sampling pixels in the following way:

1. Sample matrix $r \in \mathbb{R}^{H \times W \times 1}$, where $r_{i,j} \sim U(0, 1)\, \forall i, j$.
2. Generate binary matrix $b = r > |\xi|$, where $\xi \sim N(1, 0.0005)$.
3. The color hint is $h = \{y * b, b\}$, where $\{\}$ denotes concatenation, $*$ - element-wise product and y is color image.

We add the mask itself to the color hint to distinguish pixels of some color from empty pixels. For example, if the pixel values belong to $[0, 1]$, then 0 corresponds to the black color.

Our architecture consists of 3 fully convolutional networks: local features extractor $\mathcal{E}$, generator $\mathcal{G}$, discriminator $\mathcal{D}$. Extractor use pretrained weights that are never updated during training and generator and discriminator trained simultaneously.

Local Features Extractor. Manga images are quite complex. They contain a variety of objects and are drawn in different styles. Therefore, for high-quality colorization, it is crucial to correctly detect the boundaries and extract semantic features. For this purpose, some pretrained network is used that receives a binary image at the input, and activations of some layer of network are considered as an image description. Moreover, the weights of extractor are frozen since there is a domain gap between synthetic and real data, and such an approach prevents overfitting on synthetic data that leads to reducing the domain gap. In this work, we use SE-ResNeXt-50, activations of *conv3_4* layer are used as local features maps.

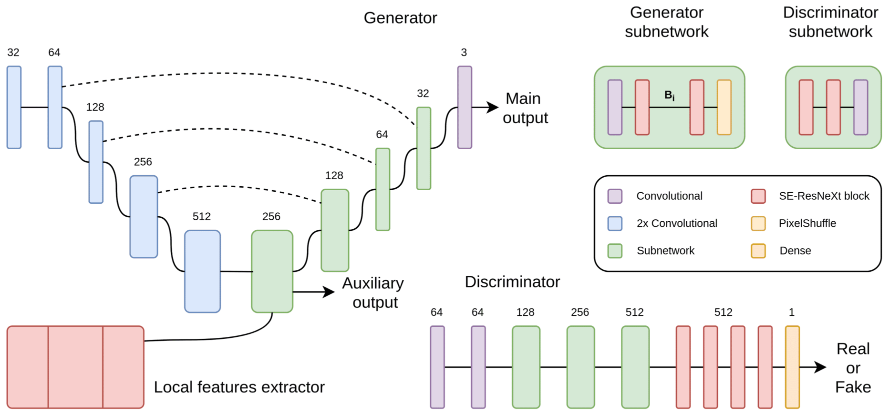

Fig. 3. Model architecture.

Generator. The generator has a UNet-like architecture. Its encoder is built from a sequence of convolutional layers that rapidly reduces the size of the input by 16 times to the size of the local features maps. This simplicity of the encoder is caused by the fact that the local features extractor is used to extract the features, and the main function of the encoder is to propagate color from a color hint.

The decoder is a sequence of 4 subnetworks and a convolutional layer at the end with tanh activation that converts the intermediate result into a final coloring. Each i-th subnetwork consists of B_i SE-ResNeXt blocks with a convolutional layer at the beginning and a sub-pixel convolutional layer at the end. Each subnetwork increases the height and width of the input by 2 times. Moreover, dilated convolutions are added to all subnetworks. It enables us to increase the receptive field without increasing the calculation cost. All activation functions are LeakyReLU except the last one. In this work B_i values are equal to 20, 10, 10, 7 respectively.

The output of the first subnetwork is also used for computing the auxiliary image using an additional subnetwork consisting of a sequence of transposed convolutions. The loss function is calculated using this image to improve the gradient propagation.

This generator is much deeper than most existing gans. The work [5] shows that the large receptive field and increased capacity of the model have a positive impact on its quality and helps produce more natural images.

The main and auxiliary outputs are tensors of size $H \times W \times 3$ that are an estimation of coloring in RGB format.

Discriminator. It is used for adversarial learning. Receives real color images and generator outputs and estimates the probability that the input image is real. It comprises a sequence of convolutional layers and SE-ResNext blocks with a sigmoid activation function at the end as shown in Fig. 3. The spectral normalization [16] is applied to the convolutional layers.

3.3 Loss

To train the model on producing analogous coloring for similar patterns and to diversify the range of colors used, we employ a combination of several loss functions.

L1 Loss. In order to enforce the generator to produce similar coloring to the ground truth, we use a pixel level L1 metric that calculates the distance between output image $\mathcal{G}(x, d, h, \mathcal{E}(x))$ and ground truth image y:

$$\mathcal{L}_{L1}^{\mathcal{G}} = ||\mathcal{G}(x, d, h, \mathcal{E}(x)) - y||_1 \tag{1}$$

Perception Loss. To ensure similarity not only at the pixel level but also at the structural level, we use perceptual loss presented in [12]. It's an L2-loss based on the distance between some CNN feature maps for generated and ground truth images. We compute it as follows:

$$\mathcal{L}_{per}^{\mathcal{G}} = \frac{1}{chw}||\mathcal{V}(\mathcal{G}(x, d, h, \mathcal{E}(x))) - \mathcal{V}(y)||_2^2, \tag{2}$$

where c, h, w denotes the number of channels, height, width of features maps and $||\cdot||_2$ denotes Frobenius norm. We use VGG-16 pretrained on ImageNet [6] as a CNN, $\mathcal{V}$ denotes activation after 4th convolutional layer.

Adversarial Loss. To diverse colorings, we use adversarial loss that computed in the following way:

$$\mathcal{L}_{G_{adv}} = \mathbb{E}_{x,h}[\log(1 - \mathcal{D}(\mathcal{G}_{main}(x, d(x), h, \mathcal{E}(x))))], \tag{3}$$

$$\mathcal{L}_{D_{adv}} = -\mathbb{E}_y[\log \mathcal{D}(y)] - \mathbb{E}_{x,h}[\log(1 - \mathcal{D}(\mathcal{G}_{main}(x, d(x), h, \mathcal{E}(x))))], \tag{4}$$

where $\mathcal{G}_{main}(\cdot)$ and $\mathcal{G}_{aux}(\cdot)$ are main and auxiliary output of the generator correspondingly.

White Color Penalty. Our model tends to leave objects uncolored if it fails to recognize them. There can be a lot of such objects because of the domain gap, so we penalize the network if the pixel value is close to the maximum corresponding to the white color. $\mathcal{L}_{white}$ is calculated as the mean square of the channel-wise sums (we assume that values of $\mathcal{G}$ output belong to [0, 1]).

Overall Loss Function. Combining all above terms, we set loss functions for generator and discriminator:

$$\mathcal{L}_G = \lambda_{L1}(\mathcal{L}_{L1}^{G_{main}} + \lambda_{aux}\mathcal{L}_{L1}^{G_{aux}}) + \mathcal{L}_{per}^{G_{main}} + \mathcal{L}_{G_{adv}} + \mathcal{L}_{white} \tag{5}$$

$$\mathcal{L}_D = \mathcal{L}_{D_{adv}} \tag{6}$$

We set $\lambda_{L1} = 10$, $\lambda_{aux} = 0.9$ to force the generator to reconstruct the original image rather than deceiving discriminator.

4 Experiments

In this section, we describe the collected dataset. Then we discuss the learning and inference processes and evaluate model performance.

4.1 Dataset

Web-scraping was used to extract images of manually colored manga: One Piece, Akira, Bleach, JoJo's Bizarre Adventure, Dragon Ball, Naruto. We also collected artistic anime drawings from the dataset Danbooru2019 [1]. We made use of images that had the tag *colorized*. It means that these images were produced from line art by manual coloring. Such images usually have distinct black edges helping to synthesize a higher quality XDoG image. Overall, 59164 color images were collected. In addition, we collected images of such black and white manga as Fullmetal Alchemist, Lone Wolf and Cub, Wolf and Spice, Pandora Hearts, Phoenix to evaluate the performance of the model and perform fine-tuning. These titles are not included in the training set and have different drawing styles, thus allowing an objective assessment to be made.

The images are often uploaded to the Internet in JPEG format with a high compression factor to reduce the memory consumption, but this causes the images to have artifacts that distort their content, thus affecting the learning process. To restore images, FFDNet was applied.

We used the following method to synthesize a monochrome image:

1. Resize the image to be with the shortest side equals $512k$, where k randomly selected from $\{2, 3, 4\}$.
2. Apply XDoG algorithm with parameters $\tau = 0.95$, $\varphi = 90$, $\epsilon = 0$, $k = 4$, $\sigma = 0.5$ and binarization using the Otsu method to the intermediate image.
3. Resize the image to be with the shortest side equals 512.

We need to resize the image to the uniform size in order to perform training. Increasing the image size helps to preserve fine details and text more efficiently when applying the XDoG algorithm and produce monochrome images with better quality.

4.2 Training Setup

The following models with pretrained weights are used for training:

- VGG-16 pretrained on ImageNet for evaluating perceptual loss.
- SE-ResNeXt-50 pretrained on monochrome drawings for tag prediction.
- FFDNet pretrained on photographs for denoising.

The weights of the generator and the discriminator are initialized with Xavier initialization. The training takes 20 epochs using the ADAM optimizer with hyperparameters $\beta_1 = 0.5$, $\beta_2 = 0.9$ and batch size is 4 since limited computational resources. Longer training leads to overfitting, increasing the domain gap. During the first 15 epochs, the learning rate of the generator and the discriminator is equal to 10^{-4} and $4 * 10^{-4}$ respectively, whereas during the last 5 epochs it is 10^{-5} and $4 * 10^{-5}$. We use an imbalanced learning rate to keep the ratio of the discriminator and generator updates equal to 1:1 since the utilization of spectral normalization slows down discriminator learning as shown in [19]. One-sided label smoothing is applied for discriminator training.

During training random cropping to 512×512 size and random horizontal flipping are applied. Moreover, we use random color shift before XDoG algorithm application.

We generate the mask for color hint with probability 0.5 the way described above, with probability 0.49 the mask contains all zeros and with probability 0.01 the mask contains all ones. The use of an empty mask provides quality network performance without hints, whereas a full mask allows the network to correctly deal with a large number of colored pixels.

The neural networks are implemented using the PyTorch library for Python. Image augmentations are performed with the Albumentations [2]. DFM images are generated using the library Snowy.

The training takes about 60 h on a device with a single NVIDIA GTX 1080TI graphics card and a Xeon® E5-2696 CPU.

4.3 Fine-Tuning

Regrettably, the XDoG image and the black and white manga image differ sufficiently, so that a network trained with one type of image cannot work properly with the other. To even greater regret, XDoG of black and white image and XDoG of color one also slightly different. Despite that visually they are almost indistinguishable, the neural network is capable to find features at XDoG of the color image that contain the information about the pixel intensity of the original image, that may not be presented at black and white images. As a result,

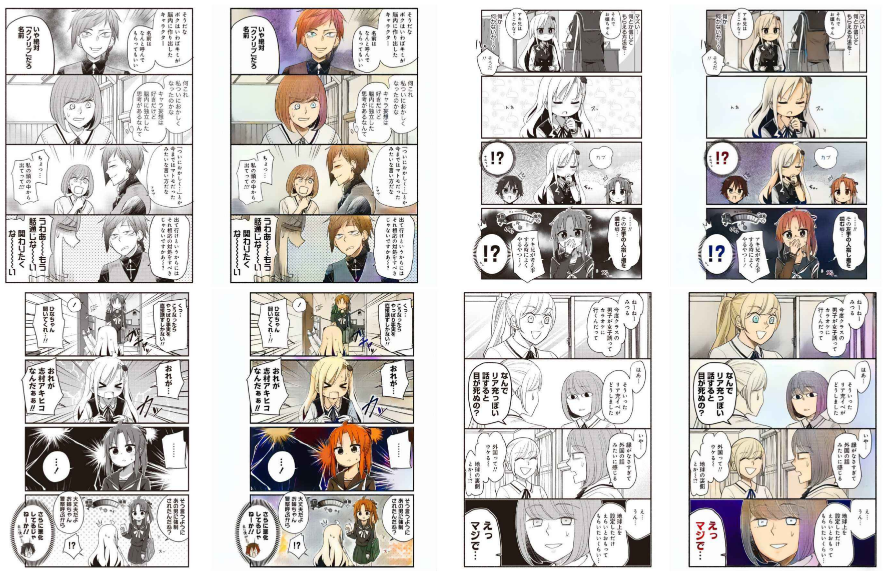
そうだな ボクはいわばキミが脳内に作り出したキャラクター
名前はなんと呼んでもらってもいい
いや絶対「クソリプ」だろ名前
何これ 私ついにおかしくなったのかな
キャラ妄想は好きだけど脳内に独立した思考があるなんて
「ついにおかしく…」とか今まではマトモだったみたいな言い方だな
ちょっ… 私の頭の中から出てって!!!
出て行けというからにはそれ相応の対処をすべきじゃないですかあ~?
うわあ~~もう話通じな~~い 関わりたくな~~~~い
くっ…こうなったらやっぱり事実を直接話すしかない!!
ひなちゃん聞いてくれ…!!
おれが…
おれが志村アキヒコなんだぁぁ!!
……
…!
そう言うようにあの男に強制されたんだね?
大丈夫だよお姉ちゃん警察呼ぶから
!?
さらに悪化してるじゃねーか!!
マズい… 何とか信じてもらえる方法を…
それでお嬢ちゃん
アキ兄はどこかな?
何か… 何かないか…?
そうだ!!
カプ
!?
……!! その左手の人指し指を噛む癖…
アキ兄が考え事する時によくやるやつ…!
!?
ねーねーみつる
今度クラスの男子が女子誘ってカラオケに行くんだって
はあ…
そういったリア充イベがどうしました
なんでリア充っぽい話すると目が死ぬの?
いや… 縁がなさすぎて外国の話みたいに感じる
外国って!! ウケる~
地球の裏側とか~!?
え… うん…
地球上を設定しただけえらいとおもってもらいたいくらい…
えっ マジで…

Fig. 4. Example of coloring without user interaction.

a domain gap is made between the training set and the images handled during inference.

To address this problem, we utilize fine-tuning with XDoG images of black and white manga. The same preprocessing procedure is applied to these images as to the color ones. During 5000 iterations, the network is trained with XDoG of color images while learning rate equal to $4 * 10^{-5}$ and 10^{-5} for the discriminator and generator. Every fifth iteration generator receives XDoG of black and white image and an empty color hint. Here, we apply the same loss function to the discriminator and use $\mathcal{L}_{G_{adv}}$ with a learning rate 10^{-6} for the generator.

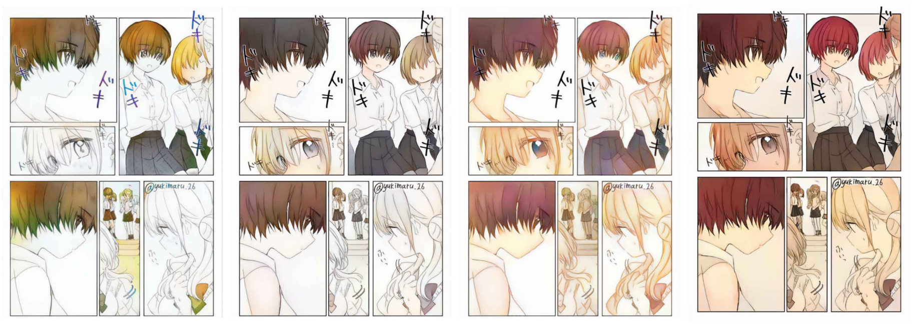

Fig. 5. The influence of white color penalty and fine-tuning on the manga that was not represented in the training and fine-tuning set: AlacGAN [5], without penalty and fine-tuning, without fine-tuning, our model.

This method enhances the quality of coloring both for images of the fine-tuning set and for the others while not requiring any labeling. Therefore, if the model is intended to be applied to the big set of images of one style, it is reasonable to perform fine-tuning using images of that set.

With a longer learning time, the coloring for images of fine-tuning set spoils due to the absence of the $L1$ or perceptual loss. For the rest of the images, the positive effect disappears as the network overfits for particular images.

4.4 Inference

Summing up the above mentioned, the algorithm for colorizing black and white manga is as follows:

1. The user inputs a black and white image of the manga page of size $H \times W$.
2. If the image corrupted with noise or JPEG artifacts, FFDNet with user-defined noise level σ is applied.
3. This image is used to generate XDoG and DFM images, as described earlier. Their size is $512 \times (W * \frac{512}{H})$, if $H \leq W$, and $(H * \frac{512}{W}) \times 512$ otherwise.
4. Zero tensor is assigned to the color hint.
5. The coloring for an empty color hint is generated, and it is used as a color image to create a nonempty color hint using the method described above (optional).
6. The acquired objects are used to produce the coloring with generator.
7. If the result is unsatisfactory, the color hint is modified by the user and go to step 6.

4.5 Results

As shown in Fig. 4, the model is capable to perform quality colorization even without color hint. The model does not paint spaces between panels if they are not filled in the source. Also, it's able to distinguish different objects on images of different styles (all exhibited images, except one, belong to manga that not represented in the training set). The network is especially effective at recognizing heads, sky, water and grass. Characters within a page have homogeneous coloring if their appearance doesn't differ drastically. Moreover, the model can qualitatively colorize the manga double-page spreads, which have a different aspect ratio than the standard images.

However, objects that the network cannot recognize with confidence are colored either in skin color or green-yellow color or even are not colored at all. Pix2Pix models tend to prefer one color, so this behavior was not unexpected. A network may also colorize text clouds, if they have a shape or content that differs from those that are contained in the training set.

The color hint is used to fix the improper coloring. Using color hint enables to color arbitrary objects in arbitrary colors, it is possible to color even small objects (see Fig. 6). However, if Step 5 of inference isn't applied, colors propagated in color hint can be indirectly distributed to neighboring objects (for example, changing the shade in the direction of the propagated color). That does not allow to change the color of one small object without adding pixels to the color hint for neighboring objects and impedes obtaining a perfect coloring.

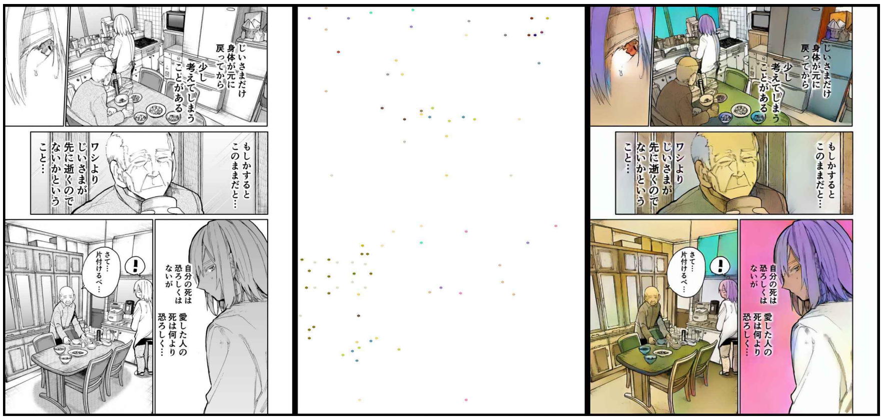
じいさまだけ身体が元に戻ってから
少し考えてしまうことがある
もしかするとこのままだと…
ワシよりじいさまが先に逝くのでないかということ…
自分の死は恐ろしくはないが
愛した人の死は何より恐ろしく…
!
さて…片付けるべ…

Fig. 6. Example of coloring with user interaction.

We use this flaw as an advantage by applying Step 5 of the inference. If the color propagated through the color hint spreads to neighboring objects, we can fill the improper white using initial coloring. In this case, such color spreading does not occur since the majority of objects on the image are already covered with pixels in the color hint. Moreover, the coloring artifacts can be removed. Nevertheless, this approach increases the computational time for obtaining the initial coloring and requires additional actions to modify coloring since we need to remove pixels of the color hint first on the object we want to recolorize.

The FID is used for the quantitative evaluation of the model performance. We calculate the distance between the set of color images of manga and the set of colorings generated by the model without user interaction. Colored images are represented by a holdout set of our manually colored images dataset, and colorings are generated with the same chapters of black and white editions and some samples are taken from the fine-tuning set. The size of both sets is 4000. For comparison, we have trained AlacGAN with our dataset using the training procedure described at [5]. In addition, we perform an ablation study to demonstrate the effectiveness of the proposed methods in bridging the domain gap. The results are presented in Table 1. As can be seen, the numerical metric mostly corresponds to the visual impression. The obtained results confirm the validity of the proposed approaches, as well as evidence the negative impact of prolonged fine-tuning. It can be noticed that AlacGAN outperforms our model without white color penalty and fine-tuning according to FID metric, but these approaches are also applicable to it. However, as can be seen in Fig. 5, AlacGAN tends to leave more uncolored objects, so we stick with our approach.

Table 1. Comparison of different models with FID metric. Notation: 1 - adversarial loss, 2 - white color penalty, 3 - fine-tuning, 4 - Step 5 of the inference, 5 - extended fine-tuning

Model	FID
Without 1–5	34,18
Without 2–5	33,79
AlacGAN	33,16
Without 3–5	31,25
Without 4–5	25,43
Without 5 (out model)	**25,05**
Without 4, with 5	29,06
With 1–5	25,74

5 Conclusion

In this paper, we proposed an approach for coloring manga page images with color hints using deep neural networks. To do this, a dataset consisting of the manually colored manga was collected, cleaned of noise and converted to monochrome images to create pairs for supervised learning. The proposed model was successfully trained on that dataset. We also suggested some approaches to reduce the domain gap.

The obtained results show that neural networks are able to produce high-quality and natural coloring of an entire manga page while maintaining homogeneity between the panels, and the use of color hint provides the ability to change the coloring and fix errors. However, due to the use of synthetic data and the removal of noise using a neural network, the model proved to be sensitive to the input and dependent on a large number of data preprocessing parameters. The results can be significantly improved by using a more advanced way of transforming a color image into black and white or by creating a proper dataset to avoid the use of synthetic data.

References

1. Branwen, G.: Danbooru 2019: A large-scale crowdsourced and tagged anime illustration dataset (2020)
2. Buslaev, A., Iglovikov, V.I., Khvedchenya, E., Parinov, A., Druzhinin, M., Kalinin, A.A.: Albumentations: fast and flexible image augmentations. Information **11**(2), 125 (2020)
3. Charpiat, G., Hofmann, M., Schölkopf, B.: Automatic image colorization via multimodal predictions. In: Computer Vision - ECCV 2008 (2008)
4. Cheng, Z., Yang, Q., Sheng, B.: Deep colorization. In: 2015 IEEE International Conference on Computer Vision (ICCV) (2015)

5. Ci, Y., Ma, X., Wang, E.A.: User-guided deep anime line art colorization with conditional adversarial networks. In: Proceedings of the 26th ACM International Conference on Multimedia (2018)
6. Deng, J., Dong, W., Socher, R., Li, L., Kai, L., Fei-Fei, L.: Imagenet: a large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition (2009)
7. Furusawa, C., Hiroshiba, K., Ogaki, K., Odagiri, Y.: Comicolorization: semi-automatic manga colorization. In: SIGGRAPH Asia 2017 Technical Briefs (2017)
8. Goodfellow, I., et al.: Generative adversarial nets. In: Advances in Neural Information Processing Systems 27. Curran Associates, Inc. (2014)
9. He, M., Chen, D., Liao, E.A.: Deep exemplar-based colorization. ACM Trans. Graph. **37**(4), 1-16 (2018)
10. Iizuka, S., Simo-Serra, E., Ishikawa, H.: Let there be color!: joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classificatfion. ACM Trans. Graph. **35**(4), 1-11 (2016)
11. Isola, P., Zhu, J., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
12. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: European Conference on Computer Vision (2016)
13. Kim, H., Jhoo, H., Park, E., Yoo, S.: Tag2pix: Line art colorization using text tag with secat and changing loss. In: Proceedings of the IEEE International Conference on Computer Vision (2019)
14. Larsson, G., Maire, M., Shakhnarovich, G.: Learning representations for automatic colorization. In: European Conference on Computer Vision (ECCV) (2016)
15. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. In: Proceedings of the IEEE (1998)
16. Miyato, T., Kataoka, T., Koyama, M., Yoshida, Y.: Spectral normalization for generative adversarial networks. In: International Conference on Learning Representations (2018)
17. Shi, M., Zhang, J.Q., Chen, S.Y., Gao, L., Lai, Y.K., Zhang, F.L.: Deep line art video colorization with a few references (2020)
18. Winnemöller, H., Kyprianidis, J.E., Olsen, S.C.: Xdog: An extended difference-of-gaussians compendium including advanced image stylization. Comput. Graph. **36**(6), 740-753 (2012)
19. Zhang, H., Goodfellow, I., Metaxas, D., Odena, A.: Self-attention generative adversarial networks. In: Proceedings of the 36th International Conference on Machine Learning (2019)
20. Zhang, K., Zuo, W., Zhang, L.: Ffdnet: Toward a fast and flexible solution for CNN-based image denoising. IEEE Trans. Image Process. **27**(9), 4608–4622 (2018)
21. Zhang, R., Isola, P., Efros, A.A.: Colorful image colorization. In: Computer Vision - ECCV 2016 (2016)
22. Zhang, R., Zhu, J.Y., Isola, E.A.: Real-time user-guided image colorization with learned deep priors. ACM Transactions Graph (2017)