Tag2Pix: Line Art Colorization Using Text Tag With SECat and Changing Loss

Hyunsu Kim*, Ho Young Jhoo*, Eunhyeok Park, and Sungjoo Yoo

Seoul National University
{gustnxodjs, mersshs, eunhyeok.park, sungjoo.yoo}@gmail.com

Abstract

Line art colorization is expensive and challenging to automate. A GAN approach is proposed, called Tag2Pix, of line art colorization which takes as input a grayscale line art and color tag information and produces a quality colored image. First, we present the Tag2Pix line art colorization dataset. A generator network is proposed which consists of convolutional layers to transform the input line art, a pre-trained semantic extraction network, and an encoder for input color information. The discriminator is based on an auxiliary classifier GAN to classify the tag information as well as genuineness. In addition, we propose a novel network structure called SECat, which makes the generator properly colorize even small features such as eyes, and also suggest a novel two-step training method where the generator and discriminator first learn the notion of object and shape and then, based on the learned notion, learn colorization, such as where and how to place which color. We present both quantitative and qualitative evaluations which prove the effectiveness of the proposed method.

1. Introduction

Line art colorization is an expensive, time-consuming, and labor-intensive task in the illustration industry. It is also a very challenging task for learning methods because the output is a fully colorized image, but the only input is a monotone line art and a small amount of additional information for colorization (*e.g.*, color strokes). The multimodal learning of segmentation and colorization is essential.

Recently, various studies have been conducted on colorization. Most of those works are based on generative adversarial network (GAN) [8], and we focus on colorization using text and line art. In researches on text hint colorization, some works tried to colorize a grayscale image with information given by a text sentence describing the color of each object [4, 19], while others modified the color of a particular part of the image using sentences [6, 21]. Though several studies exist for text-based colorization, none of them have focused on line art colorization, which is more difficult due to the relatively low amount of information contained in the input image.

*equal contribution

In the case of line art colorization, there are two typical ways to give hints for colorization. In user-guided colorization [18, 25, 33, 36], short lines with the desired color are drawn over target locations on the line art, and the output is generated by naturally filling the remaining space. In the style-transfer method [7, 9, 18, 35], an existing sample image is used as a hint for the generative network, and the output is generated following the color distribution of the given sample image. These methods successfully simplify the colorization process, but still require intervention through either skilled professionals (user-guided case) or images with similar patterns (style-transfer case), both of which are expensive.

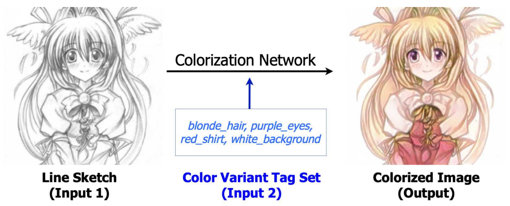

Colorization Network
blonde_hair, purple_eyes, red_shirt, white_background
Line Sketch (Input 1)
Color Variant Tag Set (Input 2)
Colorized Image (Output)

Figure 1. Example of tag-based colorization method.

As an alternative, we define a new problem for implementing line art colorization based on tag data, as shown in Figure 1. The generator network receives input data as a monotone line art and a corresponding color variant tag (CVT), such as *blue_hair* or *red_eyes*, after which the network uses these two input data to colorize the monotone line art based on the given tag information. This tag-based method minimizes the effort of offering hints for colorization; thus it can provide high quality colorization output without requiring the intervention of a skilled professional.

In this paper, we propose a GAN approach to tag-based line art colorization as well as a novel architecture for multi-label segmentation and colorization. We also present a novel training scheme which improves the stability of GAN training as well as its output quality. In addition, we offer our source codes with accompanying pre-trained networks so that readers can reproduce the results.[1] Our contributions are as follows:

- **Tag2Pix dataset**
  We offer a dataset for training the Tag2Pix network. The dataset is composed of four sets of data: color illustrations, monotone line arts, color invariant tags (CITs), and CVTs.

- **Tag2Pix network**
  We present an adversarial line art colorization network called Tag2Pix. This network is a variation of auxiliary classifier GAN (ACGAN) [22] and designed to generate a colorized image based on a monotone line art and CVTs.

- **SECat: Squeeze and Excitation with Concatenation**
  We propose a novel network structure that enhances multi-label segmentation and colorization. This method helps to color even small areas such as eyes.

- **Two-step training with changing loss**
  We present a novel loss combination and curriculum learning method for the Tag2Pix network. This method divides the learning focus between segmentation and colorization in order to train the network in a stable and fast manner.

2. Related work

2.1. GAN for colorization

GAN [8] offers superior quality in generation tasks compared to conventional image generation methods. Pix2pix [13] changes grayscale images to color images by adding GAN's adversarial loss to the reconstruction loss typically used for CNN. ACGAN [22] and Zhang *et al*. [35] show that an auxiliary decoder network can increase the stability and expression power of a generative network. In this paper, this aspect of ACGAN is utilized in the design of our network because it is suitable for learning the features of CVTs and CITs.

2.2. Sketch-based line art colorization

Several studies have been conducted on GAN for line art colorization. A typical method of line art colorization is to provide a color hint for the line art. PaintsChainer [33], Ci *et al*. [5], and Scribbler [25] implement automatic colorization by using short lines of specified color as hints for the colorization of targeted line art areas. In StyleTransfer [35], the color style of an illustration used as input data is transferred to the original line art image. Style2Paints [18, 36] extends StyleTransfer by adding a refinement stage, which provides a state-of-the-art result. However, both methods still require expertise from the user for either specifying colors at every line-separated location, or for preparing a properly styled illustration as input data for each original line art. Due to these limitations, colorization remains expensive and difficult. In this work, we propose a simple tag-based colorization scheme to provide a low-cost and user-friendly line art colorization option.

2.3. Text related vision task

StackGAN [34] accepts full text sentences to synthesize a color image. SISGAN [6] and Nam *et al*. [21] use sentences to change the color of a particular area in a color image. Manjunatha *et al*. [19] and Chen *et al*. [4] colorize grayscale images using sentences that describe objects with colors. Meanwhile, Illustration2vec [24] uses a VGG [29] network to extract semantic tags from color illustration. Based on Illustration2vec, Jin *et al*. [14, 15] present a study on creating artificial anime-style faces by combining a given tag with random noise. The concept of extracting semantic features using CNN is adopted in our work to extract CITs from a given line art.

2.4. Rebalance weights in feature maps

In styleGAN [16], a style-based generator is used to improve the output quality by modifying intermediate features after convolutional layers based on adaptive instance normalization (AdaIN [12]). By injecting the encoded style information into the intermediate feature maps, the generator can synthesize realistic images for various target styles. In this work, we propose a novel network structure called SECat, which improves the stability and quality of line art colorization with minimal effort.

3. Problem and Dataset

3.1. Problem definition

Our problem is a tag-based line art colorization which automatically colors a given line art using color variant text tags provided by the user. Line art is a grayscale image that includes only the edges of objects, and a CVT determines the desired color of the target object in the line art. In order to address this problem, our proposed network extracts a feature, such as *hat* or *backpack*, which provides information about the shape of the image for color invariant tags (CITs) in a given line art. Thus, as shown in Figure 1, our proposed network colorizes a given line art with color variant tags and color invariant tag features. Note that the user

[1] https://github.com/blandocs/Tag2Pix

does not need to provide color invariant tags, and only has to provide the color-related tags.

3.2. Tag2Pix dataset

Data filtering We used a large-scale anime style image dataset, Danbooru2017 [1]. Each illustration of Danbooru2017 has very detailed tags for pose, clothing, hair color, eye color, and so on. In order to achieve our goals, we selected 370 CITs and 115 CVTs that were each used more than 200 times for our training dataset.

We also selected images with only one person in simple background. Each image used from the Danbooru2017 set has a size of 512×512 letterboxed. Through mirror-padding, we removed the letterbox and obtained $39,031$ high-definition images. Additional face images were also extracted through lbpascade_animeface [20] to provide more delicate face colorization. A total of $16,224$ face images were obtained by selecting images with a resolution of 128×128 or higher.

Line art extraction We needed to extract line arts from color illustrations for supervised learning, but traditional edge detection algorithms, such as Canny Edge Detector [3], failed to create natural artistic line arts. Thus, in order to obtain line arts with clear edges and various styles, we used multiple methods of line art extraction as illustrated in Figure 2.

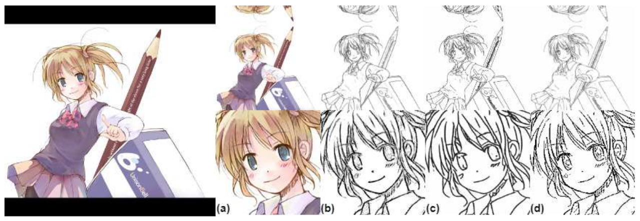


Figure 2. Example of line art extraction methods. Left: sample image from Danbooru2017. Top: training data obtained by (a) removing letterbox with mirror padding, (b) sketchKeras [17], (c) Sketch Simplification [27], and (d) XDoG [31]. Bottom: cropped face images using lbpcascade_animeface [20].

We mainly extracted sketches through the sketchKeras [17] network specialized in line art creation in the style of anime. However, the sketchKeras images showed a tendency toward the style of a pencil sketch, with the presence of an unnecessary depiction or incorrect control of the line thickness. We therefore used a simplified sketch based on sketchKeras, additionally using Sketch Simplification [27, 28]. These images are close to digital line art because they have nearly constant line thickness. Because the line thickness of the simplified result depends on the resolution of the input image, the input was enlarged to 768×768. Lastly, we extracted an algorithmic line art from the grayscale of the original color image using XDoG [31].

Training a network with only a single type of sketch, such as sketchKeras, has shown a tendency to overfit to the sketch input and retrace the RGB values from the sketch. Through various styles of sketches, we have been able to avoid this effect.

4. Tag2Pix network and loss design

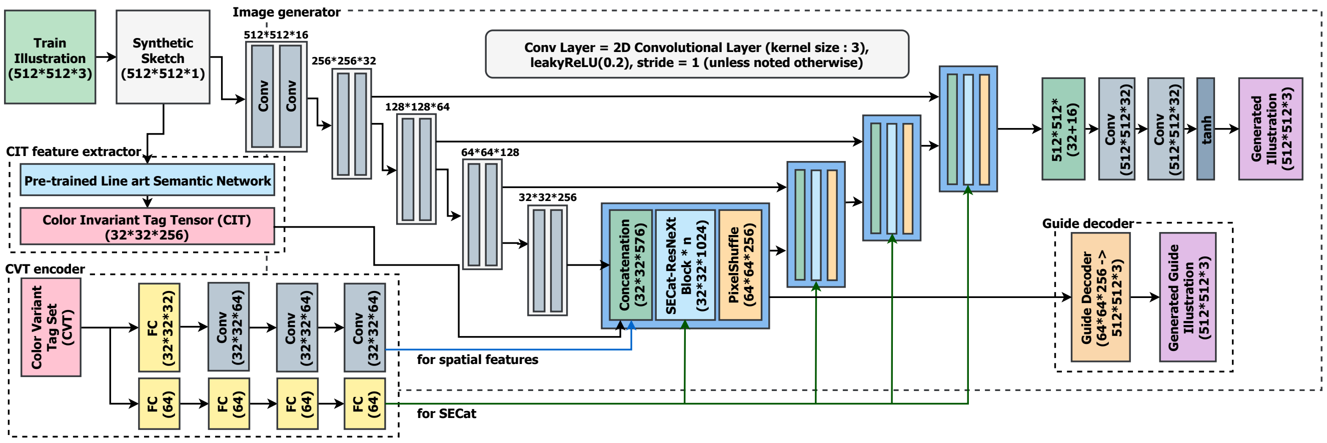
4.1. Generator and discriminator network

Figure 3 shows the architecture of the generator network, which takes a line art and color tags as input and gives a colored illustration as output. As shown in Figure 3, the network is composed of four main parts: CIT feature extractor, image generator, CVT encoder, and guide decoder. The user can obtain a colored line art simply by providing a line art and CVTs without providing CITs, as shown in Figure 3.

CITs tend to represent shape information which can serve as a guide for better coloring, providing useful information for the correct applicaiton of color to desired positions. The CIT feature extractor is designed to extract the features of CITs from the given line art, and to utilize those in our proposed GAN architecture. In order to extract the features of CITs, we pre-trained a feature extraction network based on SE-ResNeXt-50 [11, 32]. The network was trained to predict the multi-label CITs and the intermediate feature map after *conv3_4* with ReLU is provided to the image generator.

The CVT encoder is designed to embed the given CVTs into the latent space. The CVT encoder consists of two sets of layers for output, providing spatial features to be merged into the encoded feature map of image generator, and assisting the layer re-calibration feature of SECat. SECat is explained in Section 4.2. The CVT input is first encoded as a one-hot vector, and the vector is separately embedded through multiple fully-connected (FC) layers and convolution layers. Even though CVTs do not have spatial information, convolution layers have better performance than FC layers in lower computation overhead.

The image generator is based on U-Net [23], which is designed to generate high-resolution images. As Figure 3 shows, the image generator first produces an intermediate representation ($32 \times 32 \times 576$) by concatenating the feature maps of convolutional layers ($32 \times 32 \times 256$) with both the CIT feature extractor output and the spatial features output of the CVT encoder for U-Net. The multiple decoder blocks then run to produce a high-quality colored illustration. Specifically, each decoder block takes as input the feature maps from the previous decoder block and the convolutional layer of the same spatial dimension in the U-Net structure. Each decoder block is based on the pixel shuffle operation, which is an upsampling method used to reduce checkerboard artifacts [26].



Train Illustration (512*512*3)
Synthetic Sketch (512*512*1)
Image generator
512*512*16
Conv
Conv
256*256*32
128*128*64
64*64*128
32*32*256
Conv Layer = 2D Convolutional Layer (kernel size : 3), leakyReLU(0.2), stride = 1 (unless noted otherwise)
CIT feature extractor
Pre-trained Line art Semantic Network
Color Invariant Tag Tensor (CIT) (32*32*256)
CVT encoder
Color Variant Tag Set (CVT)
FC (32*32*32)
Conv (32*32*64)
Conv (32*32*64)
Conv (32*32*64)
FC (64)
FC (64)
FC (64)
FC (64)
for spatial features
for SECat
Concatenation (32*32*576)
SECat-ResNeXt Block * n (32*32*1024)
PixelShuffle (64*64*256)
512*512* (32+16)
Conv (512*512*32)
Conv (512*512*32)
tanh
Generated Illustration (512*512*3)
Guide decoder
Guide Decoder (64*64*256 -> 512*512*3)
Generated Guide Illustration (512*512*3)

Figure 3. The overall structure of the generator network. The numbers within brackets represent tensor dimension (width×height×depth).

The image generator has a deep structure, and can result in the vanishing gradient problem. In order to facilitate network training, we adopt the guide decoder [35]. As shown in Figure 3, the guide decoder is connected to the first decoder block, and produces a colored illustration. The guide decoder offers a new loss path to an intermediate feature map, which improves quality and helps to mitigate the vanishing gradient problem.

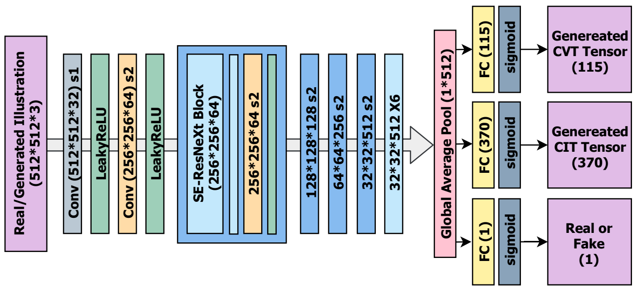


Figure 4. The overall structure of the discriminator network.

Figure 4 shows the overview of the discriminator. After receiving a color image as input, the discriminator determines whether the input is genuine, and at the same time predicts which CVTs and CITs are used.

This network is inspired by ACGAN [22], as the multi-label classification (especially for CVTs) plays a key role in the target task. The generator and discriminator of the existing ACGAN are trained to successfully generate images of a single class. On the contrary, our GAN is trained to generate colored illustrations and multi-label classification using CVTs and CITs.

4.2. SECat - Squeeze and Excitation with Concatenation

Generally, colorization hints are encoded into a high-level feature which is provided to the generator network as an input to the decoder block [2, 5, 35]. However, according to our observation, the aforementioned method has a disadvantage for tag-based line art colorization. When we adopt this method for the proposed network by encoding the CVT input into a spatial feature and merging it with the input feature of the generator's decoder block, colorization is performed well for large objects such as hair, but not for the detailed small objects such as eyes.

In order to overcome this limitation, we propose a novel structure named SECat (Squeeze and Excitation with Concatenation). SECat is inspired by the network structure of styleGAN [16], which adjusts the intermediate feature map using affine transform in order to inject style information to the generator with similar structure to SENet [11] in terms of weight rebalancing.

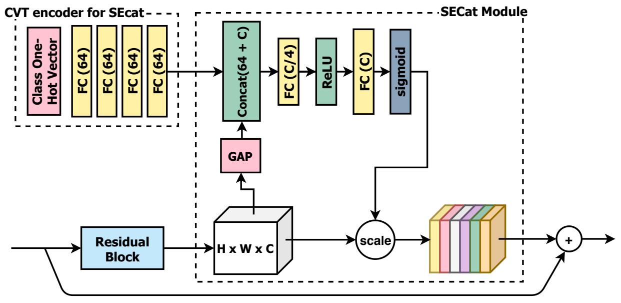


Figure 5. SECat block.

Figure 5 shows an overall structure of SECat. In SENet [11], the *squeeze* is the global information extracting process that generates a channel-dimension vector using 2-D spatial global average pooling. *Excitation* is the adaptive recalibration process that generates channel-wise importance weights using two FC layers with a bottleneck. The output feature maps of a residual block are scaled by channel-wise importance weights obtained through a squeeze and excitation process. In the case of SECat, the user-given CVTs are encoded by several FC layers, and the encoded vectors are merged to the output feature of the squeeze process. This merged feature is propagated to the excitation process so that CVT information is not only incorporated, but also utilized to emphasize important features in all decoding blocks of the generator.

4.3. Loss design with two-step curriculum training

Unlike previous works which utilize hints containing both color and location information [5, 33], a tag-based hint does not contain any information of location and shape to guide coloring. Thus, semantic segmentation for localized colorization is especially difficult to perform due to the lack of spatial information and RGB value hints. However, if the network is trained for both tasks in a single stage, the generated images often suffer from the problems of color mixing and bleeding, as exemplified in Figure 6. According to our observation, when training the network for both segmentation and coloring, it is essential for the network to learn two tasks in the curriculum, first learning semantic segmentation, followed by colorization.

In order to obtain the curriculum-based learning, we propose a two-step training with changing loss where learning proceeds while applying two different losses sequentially.

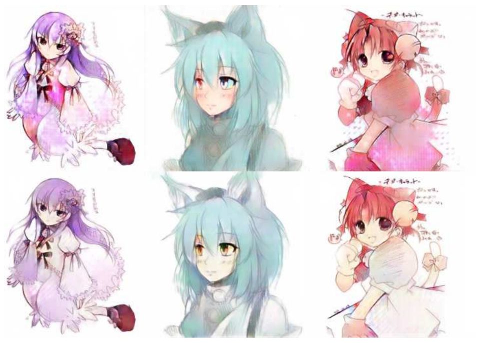


Figure 6. Illustration of two-step training effect. Top: single-step trained result, Bottom: two-step trained result. In the single-step case, color is mixed and bled in each segment.

Step 1. Segmentation In the first step, we focus on low-level features like edge and shade. The tag classification losses are not utilized for both generator and discriminator; thus the generator and discriminator are trained based only on adversarial loss $\mathcal{L}_{adv}$ and reconstruction loss $\mathcal{L}_{rec}$ as follows:

$$\begin{aligned} \mathcal{L}_D &= -\mathcal{L}_{adv}, \\ \mathcal{L}_G &= \mathcal{L}_{adv} + \lambda_{rec}\mathcal{L}_{rec}. \end{aligned} \tag{1}$$

The more detailed equation of adversarial loss is as follows:

$$\mathcal{L}_{adv} = \mathbb{E}_y[\log D_{adv}(y)] + \mathbb{E}_x[\log(1 - D_{adv}(G_f(x, c_v)))], \tag{2}$$

where x and y are the paired domain of line arts and real color illustrations, c_v is the CVT values, G_f is the synthesized color image from generator, and D_{adv} is the discriminator output of real or fake. In Equation 1, λ_{rec} is a weighting factor for the reconstruction loss $\mathcal{L}_{rec}$ that is expressed as follows:

$$\mathcal{L}_{rec} = \mathbb{E}_{x,y}[||y - G_f(x, c_v)||_1 + \beta||y - G_g(x, c_v)||_1], \tag{3}$$

where G_g is the output of the guide decoder and hyperparameter β is 0.9. $\mathcal{L}_{rec}$ is a pixel-wise L1 loss between an authentic and the generated images. We set λ_{rec} large enough (1000) to make the network to follow the original image distribution rather than deceiving each other. This training stage guides the network to learn semantic information regarding line arts, resulting in a more precise and clearer borderline with adequate light and shade.

Step 2. Colorization In the second step, the tag classification loss $\mathcal{L}_{cls}$ is introduced, which makes the generator and discriminator learn colorization based on a better understanding of object shape and location. The step 2 losses are as follows:

$$\begin{aligned} \mathcal{L}_D &= -\mathcal{L}_{adv} + \lambda_{cls}\mathcal{L}_{cls}, \\ \mathcal{L}_G &= \mathcal{L}_{adv} + \lambda_{cls}\mathcal{L}_{cls} + \lambda_{rec}\mathcal{L}_{rec}. \end{aligned} \tag{4}$$

In Equation 4, λ_{cls} is the weighting factor indicating the importance of the CIT and CVT. Note that the weighting factor plays an important role in our proposed two-step training with changing loss. The classification loss, which trains the discriminator for evaluating the amount of information associated with tags in the input image, is obtained as follows:

$$\begin{aligned} \mathcal{L}_{cls} &= \mathbb{E}_{y,c_v,c_i}[-\log D_{cls}(c_v, c_i|y)] + \\ &\mathbb{E}_{x,c_v,c_i}[-\log D_{cls}(c_v, c_i|G_f(x, c_v))], \end{aligned} \tag{5}$$

where c_i is the CIT values and $D_{cls}(c_v, c_i|y)$ binary classification for each tag, given y. The discriminator tries to predict CVT and CIT with a high probability. According to our experiments, the two-step approach makes the training of our GAN model more stable and faster.

Colorization for real-world sketch Although well-colored results can be obtained with two-step training, but a problem remains, as our network tends to be overfitted to our artificial sketch input (Section 3.2). The artificial sketch becomes very clear and neatly drawn, but the authentic sketch is blurred and has indistinct and thin edges, as shown in Figure 7. Due to the different characteristics between the training sketches and the real-world sketches, the colorization results from real-world line arts (middle images in Figure 7) were blurred and diffused.

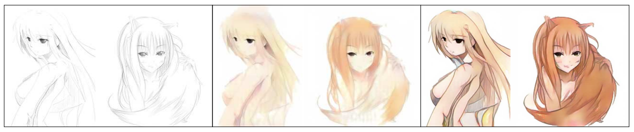


Figure 7. Real-world line arts (left), colorization results without (middle) and with brightness adjustment (right).

In order to adapt the proposed network to the real-world sketch, we trained an additional 3 epochs with a brightness control technique [25]. During the additional training stage, the brightness of the input image was scaled along $\mathcal{U}(1, 7)$, intentionally weakening the thickness and strength of the black lines so that proper segmentation was performed for an extremely faint sketch.

5. Experiment

5.1. Colorization with various CVTs

Figure 8 shows the colorized images generated by Tag2Pix. Images of each row in the figure are colorized with two common CVTs and one different CVT. The images demonstrate that our network colorizes the line art naturally with various combinations of color tags.

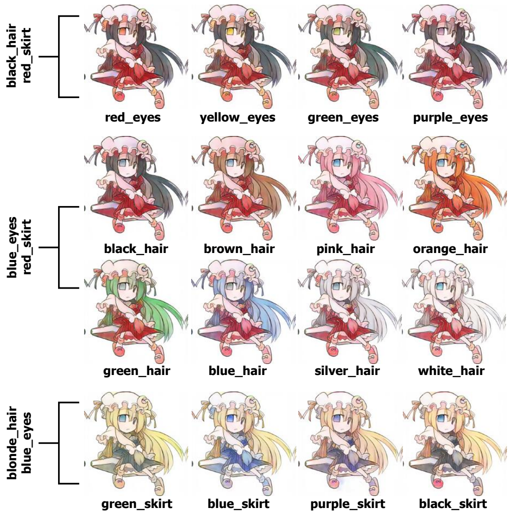


Figure 8. Colorization results.

5.2. Comparison to other networks with user study

As users will utilize automatic coloring solutions, their real judgments of the output are critical. In order to evaluate the proposed method, we conducted user studies comparing our network to other networks for sketch-based and text-based colorization. 20 people were employed offline, and they compared the quality of colorized output without any prior knowledge using a five-point Likert scale over four categories. The evaluation criteria were as follows:

- **Color Segmentation**
  The extent to which the colors do not cross to other areas, and individual parts are painted with consistent colors.
- **Color Naturalness**
  How naturally the color conforms to the sketch. The colors should match the mood of the painting.
- **Color Hints Accuracy**
  How well the hints are reflected. Output results should have red hair if a hint for red hair is given.
- **Overall Quality**
  The overall quality of the colorized result.

Sketch-based network First, our method is compared against sketch-based line art generation methods. We chose Style2Paints (specifically style-transfer network V3) [18] and PaintsChainer [33], which give hints using a reference image and color strokes, respectively. In Style2Paints, we created comparative images using publicly available code. In PaintsChainer, we used the service provided through the official website. We colorized 140 real-world line arts using three colorization methods. Thus, 140 test sets were made available, and each user evaluated 30 sets randomly selected out of the 140 test sets.

Categories	PaintsChainer (*)	Style2Paints	Tag2Pix
Segmentation	2.51	3.51	**3.94**
Naturalness	2.44	3.47	**3.91**
Accuracy	3.28	3.73	**3.93**
Quality	2.43	3.47	**3.86**

Table 1. User study of sketch-based colorization network. (*) Coloring network version was randomly selected between Tanpopo, Satsuki, and Canna.

Table 1 shows that Tag2Pix performed best in all evaluation metrics. In the style-transfer methods, the color is very clear and the light and shade are noticeable. However, as shown in Figure 9, color mismatching and color mixing occur frequently if the reference image has a different pose relative to the line art. In PaintsChainer, the color bleeding problem is very serious. For example, eye color spreads to

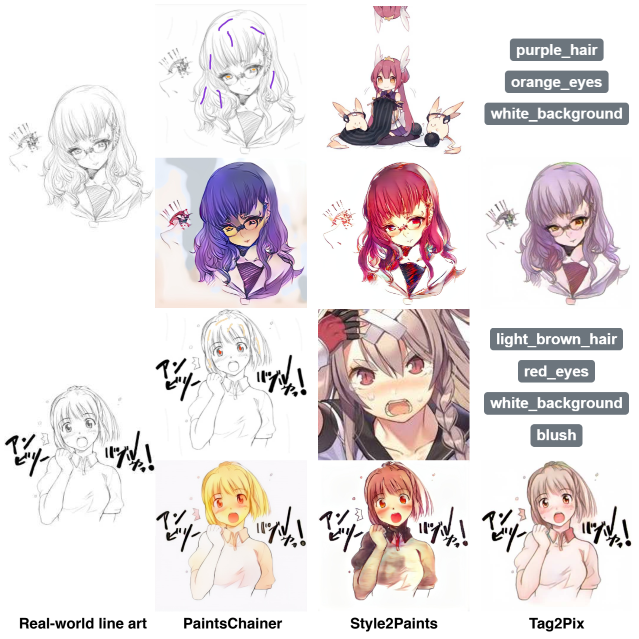


Figure 9. Comparison in sketch-based networks, PaintsChainer (left) [33] and Style2Paints (middle) [18], and ours (right). All networks painted real-world line arts. The above image of each network is a hint for colorization and the below image of each network is the colorized output. The figure shows ours has better segmentation than PaintsChainer and Style2Paints.

the face frequently due to the lack of proper segmentation for small areas. Compared to those methods, even though there is not any information about RGB values or tag location, Tag2Pix segments each part very accurately and reflects CVT hints well.

Text-based network We also conducted a comparison of Tag2Pix against the text-based colorization network. Chen *et al*. [4] is a conceptual design, so fair comparison is difficult because it requires a separate implementation per dataset. Additionally, SISGAN [6] completely failed to colorize, despite a small input size (74×74), failing to preserve the outline of the sketch, and producing a strange result. Manjunatha *et al*. [19] was instead selected as a comparison target, and the evaluation was conducted using the publicly available code. Because [19] colorizes the image using a sentence, we converted CVTs to sentences for both training and testing, matching linguistic expression levels to ensure fair comparison. For example, *red_hair* and *blue_skirt* tags were converted to *a girl with red hair wearing blue skirt*.

Table 2 shows that the proposed network significantly outperforms the baseline. As shown in Figure 10, because [19] receives a grayscale image that preserves the shade and light of the original image, it seems plausible at a glance. However, it cannot produce a detailed segmentation of hair, eyes, etc. Our proposed network segments the specific object areas well mainly due to structural and training improvements from the CIT feature extractor, SECat, and two-step training.

Categories	Manjunatha *et al.*	Tag2Pix
Segmentation	3.16	**4.13**
Naturalness	3.46	**4.00**
Accuracy	3.27	**3.99**
Quality	3.27	**3.86**

Table 2. User study of text-based colorization network. Ours outperforms especially in segmentation.

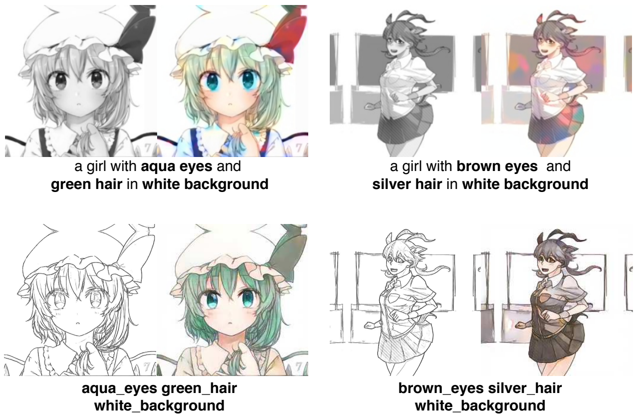

a girl with aqua eyes and green hair in white background
a girl with brown eyes and silver hair in white background
aqua_eyes green_hair white_background
brown_eyes silver_hair white_background

Figure 10. Comparison in text-based networks, Manjunatha *et al*. [19] (top), and ours (bottom). [19] used a grayscale image and a text sentence, and ours used a line art and text tags. [19] suffers from color bleeding and color mixing. Our colorized images have very clear lines and colors associated with the color tags.

Tag-based colorization has the advantage of being free from grammar and sentence structure, which significantly simplifies input encoding. Meanwhile, natural language sentences have advantages in generalization. We plan to extend the proposed approach to support natural language inputs in future work.

5.3. Comparison between CVT embedding schemes

Fréchet Inception Distance (FID) [10] is a well-known metric for the quantitative evaluation of generated output quality. The FID quantifies the similarity between generated and ground truth images by using a pre-trained Inception-v3 [30] model to evaluate the difference of output distribution.

To show that SECat works better than other multi-label embedding methods, we embedded CVT features to our network in diverse ways and calculated the FIDs. Figure 11 shows various kinds of decoding blocks in the generator. Blocks (a) and (b) are ResNeXt and SE-ResNeXt blocks, respectively. In blocks (c) and (d), CVT vector (64) from CVT Encoder is the same as SECat (e). The encoded vector is expanded to $H \times W \times 64$ and is concatenated in front of the first conv layer of the ResNeXt block (c), or every conv layer (c′). (d) uses AdaIN and affine transformation as in styleGAN, and (e) is our SECat block.

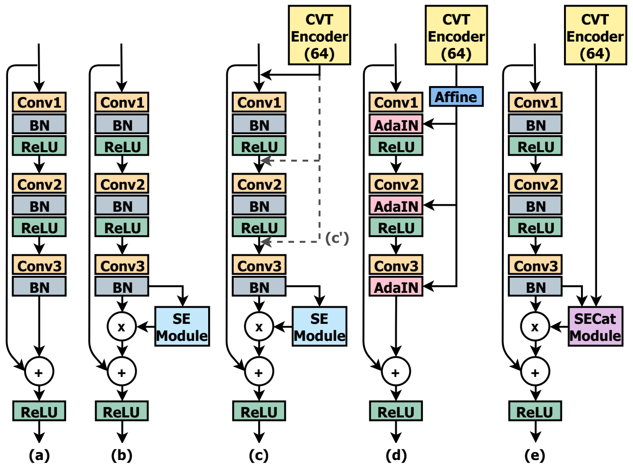


Figure 11. Diagram of each network blocks. (a) ResNeXt block, (b) SE-ResNeXt block, (c) Concatenating encoded CVT, (d) AdaIN, and (e) Ours (SECat).

We generated 6,545 colorized images using real-world line arts and random color tags. We calculated the FID by comparing 4,127 real illustrations in a test set. Note that the background tag is fixed as white to minimize the disturbance of background, and 10 epochs were used for each step of the two-step training. The metric was evaluated when the network showed the lowest FID during training. The input image size was adjusted to 256×256 for fast comparison.

Table 3 shows that our SECat block gives the lowest FID, and also has fewer parameters than other embedding methods (c) and (d). Although there is only a slight difference in FID between (c) and ours, as Figure 12 shows, our method more clearly colorizes small features such as eyes.

We conducted an additional user study to compare embedding methods in more detail. All settings are almost the same as those in Section 5.2, but we hired 27 new users and adopted 6,545 images used for calculating FIDs. As shown in Table 4, (b) merges CVT features for spatial features once with the deepest decoder block, and is inferior to (c) and (e), which merge additional light CVT features in every decoder block. By incorporating CVT information into the network and utilizing that information to emphasize the features, (e) SECat dominated all evaluation criteria.

5.4. Ablation studies and colorization for sequential sketches

We conducted extensive studies analyzing the effect of network components and loss terms, and performed colorization using consistent tags for sequential sketches on video. Check the supplementary material for details.

Block	FID	Params
(a) ResNeXt	52.49	13.85M
(b) SE-ResNeXt	45.48	15.71M
(c) Concat front	39.69	16.03M
(c′) Concat all	47.08	17.37M
(d) AdaIN	66.39	16.51M
(e) SECat (ours)	**39.21**	15.81M

Table 3. FIDs and parameter counts of each generator network. Ours gives the best result in FID and the smallest amount of additional parameters.

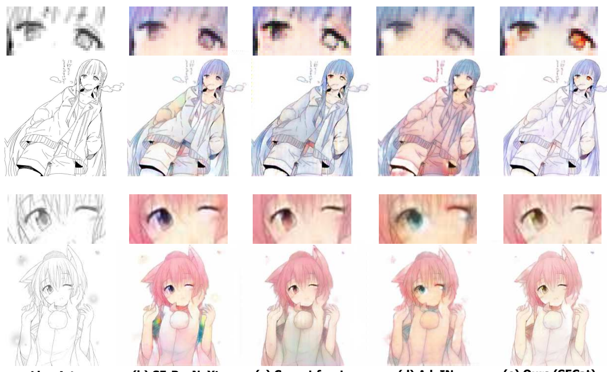


Figure 12. Colorization output of each network block up to 20 epochs. Only our network appropriately colorizes eyes from CVT. Test inputs are real-world line arts. Top: *blue_hair, red_eyes*, and Bottom: *pink_hair, yellow_eyes*.

Categories	(b)	(c)	(e) SECat
Segmentation	2.85	3.29	**3.51**
Naturalness	2.85	3.40	**3.60**
Accuracy	2.75	3.51	**3.60**
Quality	2.78	3.30	**3.54**

Table 4. User study of CVT embedding schemes. (b), (c) and (e) correspond to SE-ResNeXt, Concat front and SECat in Table 3, respectively.

6. Conclusion

In this paper, we presented a novel GAN-based line art colorization called Tag2Pix which produces high-quality colored images for given line art and color tags, and trained the network with an introduced Tag2Pix dataset. The proposed SECat architecture properly colorizes even small features, and the proposed two-step training scheme shows that pre-training the network for segmentation is a prerequisite to learning better colorization. Various user studies and experiments show that Tag2Pix outperforms the existing methods in terms of segmentation, naturalness, representation of color hints, and overall quality.

Acknowledgement

This work was supported by National Research Foundation of Korea (NRF-2016M3A7B4909604) and Samsung Electronics.

References

[1] Anonymous, Danbooru community, Gwern Branwen, and Aaron Gokaslan. Danbooru2017: A large-scale crowd-sourced and tagged anime illustration dataset. `https://www.gwern.net/Danbooru2017`, 2018. Accessed: 2019-03-22.

[2] Hyojin Bahng, Seungjoo Yoo, Wonwoong Cho, David Kee-tae Park, Ziming Wu, Xiaojuan Ma, and Jaegul Choo. Coloring with words: Guiding image colorization through text-based palette generation. *European Conference on Computer Vision (ECCV)*, 2018.

[3] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 1986.

[4] Jianbo Chen, Yelong Shen, Jianfeng Gao, Jingjing Liu, and Xiaodong Liu. Language-based image editing with recurrent attentive models. *Computer Vision and Pattern Recognition (CVPR)*, 2018.

[5] Yuanzheng Ci, Xinzhu Ma, Zhihui Wang, Haojie Li, and Zhongxuan Luo. User-guided deep anime line art colorization with conditional adversarial networks. *ACM International Conference on Multimedia (MM)*, 2018.

[6] Hao Dong, Simiao Yu, Chao Wu, and Yike Guo. Semantic image synthesis via adversarial learning. *International Conference on Computer Vision (ICCV)*, 2017.

[7] Chie Furusawa, Kazuyuki Hiroshiba, Keisuke Ogaki, and Yuri Odagiri. Comicolorization: Semi-automatic manga colorization. *SIGGRAPH Asia Technical Briefs*, 2017.

[8] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Neural Information Processing Systems (NIPS)*, 2014.

[9] Paulina Hensman and Kiyoharu Aizawa. cgan-based manga colorization using a single training image. *International Conference on Document Analysis and Recognition (ICDAR)*, 2017.

[10] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Neural Information Processing Systems (NIPS)*, 2017.

[11] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. *Computer Vision and Pattern Recognition (CVPR)*, 2018.

[12] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. *International Conference on Computer Vision (ICCV)*, 2017.

[13] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[14] Yanghua Jin. Make.girls.moe. `https://make.girls.moe`, 2017. Accessed: 2019-03-22.

[15] Yanghua Jin, Jiakai Zhang, Minjun Li, Yingtao Tian, Huachun Zhu, and Zhihao Fang. Towards the automatic anime characters creation with generative adversarial networks. *arXiv:1708.05509*, 2017.

[16] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[17] lllyasviel. sketchkeras. `https://github.com/lllyasviel/sketchKeras`, 2018. Accessed: 2019-03-22.

[18] lllyasviel. style2paints. `https://github.com/lllyasviel/style2paints`, 2018. Accessed: 2019-03-22.

[19] Varun Manjunatha, Mohit Iyyer, Jordan Boyd-Graber, and Larry Davis. Learning to color from language. *North American Chapter of the Association for Computational Linguistics (NAACL)*, 2018.

[20] nagadomi. lbpcascade_animeface. `https://github.com/nagadomi/lbpcascade_animeface`, 2011. Accessed: 2019-03-22.

[21] Seonghyeon Nam, Yunji Kim, and Seon Joo Kim. Text-adaptive generative adversarial networks: Manipulating images with natural language. *Neural Information Processing Systems (NIPS)*, 2018.

[22] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier GANs. *International Conference on Machine Learning (ICML)*, 2017.

[23] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2015.

[24] Masaki Saito and Yusuke Matsui. Illustration2vec: A semantic vector representation of illustrations. *SIGGRAPH Asia Technical Briefs*, 2015.

[25] Patsorn Sangkloy, Jingwan Lu, Chen Fang, Fisher Yu, and James Hays. Scribbler: Controlling deep image synthesis with sketch and color. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[26] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. *Computer Vision and Pattern Recognition (CVPR)*, 2016.

[27] Edgar Simo-Serra, Satoshi Iizuka, and Hiroshi Ishikawa. Mastering sketching: Adversarial augmentation for structured prediction. *ACM Transactions on Graphics (TOG)*, 2018.

[28] Edgar Simo-Serra, Satoshi Iizuka, Kazuma Sasaki, and Hiroshi Ishikawa. Learning to simplify: Fully convolutional networks for rough sketch cleanup. *ACM Transactions on Graphics (SIGGRAPH)*, 2016.

[29] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations (ICLR)*, 2015.

[30] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *Computer Vision and Pattern Recognition (CVPR)*, 2016.

[31] Holger WinnemöLler, Jan Eric Kyprianidis, and Sven C Olsen. Xdog: An extended difference-of-gaussians compendium including advanced image stylization. *Computers & Graphics*, 2012.

[32] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[33] Taizan Yonetsuji. Paintschainer. `https://paintschainer.preferred.tech/index_en.html`, 2017. Accessed: 2019-03-22.

[34] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. *International Conference on Computer Vision (ICCV)*, 2017.

[35] Lvmin Zhang, Yi Ji, Xin Lin, and Chunping Liu. Style transfer for anime sketches with enhanced residual u-net and auxiliary classifier GAN. *Asian Conference on Pattern Recognition (ACPR)*, 2017.

[36] Lvmin Zhang, Chengze Li, Tien-Tsin Wong, Yi Ji, and Chunping Liu. Two-stage sketch colorization. *ACM Transactions on Graphics (TOG)*, 2018.