Two-stage Sketch Colorization

LVMIN ZHANG*, Soochow University
CHENGZE LI*, The Chinese University of Hong Kong
TIEN-TSIN WONG, The Chinese University of Hong Kong
YI JI, Soochow University
CHUNPING LIU, Soochow University

Fig. 1. Our framework colorizes a sketch with two stages. The first stage renders the sketch into a rough color draft image. To refine the color draft, we introduce a second stage to identify mistakes and refine them, in order to obtain the final result. © AU (DeviantArt artist).

Sketch or line art colorization is a research field with significant market demand. Different from photo colorization which strongly relies on texture information, sketch colorization is more challenging as sketches may not have texture. Even worse, color, texture, and gradient have to be generated from the abstract sketch lines. In this paper, we propose a semi-automatic learning-based framework to colorize sketches with proper color, texture as well as gradient. Our framework consists of two stages. In the first drafting stage, our model guesses color regions and splashes a rich variety of colors over the sketch to obtain a color draft. In the second refinement stage, it detects the unnatural colors and artifacts, and try to fix and refine the result. Comparing to existing approaches, this two-stage design effectively divides the complex colorization task into two simpler and goal-clearer subtasks. This eases the learning and raises the quality of colorization. Our model resolves the artifacts such as water-color blurring, color distortion, and dull textures.

We build an interactive software based on our model for evaluation. Users can iteratively edit and refine the colorization. We evaluate our learning model and the interactive system through an extensive user study. Statistics shows that our method outperforms the state-of-art techniques and industrial applications in several aspects including, the visual quality, the ability of user control, user experience, and other metrics.

CCS Concepts: • **Applied computing** → **Arts and humanities**; *Fine arts*; *Media arts*;

Additional Key Words and Phrases: colorization, sketch, line arts

ACM Reference format:
LvMin Zhang*, Chengze Li*, Tien-Tsin Wong, Yi Ji, and ChunPing Liu. 2018. Two-stage Sketch Colorization. *ACM Trans. Graph.* 37, 6, Article 261 (November 2018), 14 pages.
https://doi.org/10.1145/3272127.3275090

This work was partially supported by National Natural Science Foundation of China (NSFC Grant No. 61773272, 61272258, 61301299, 61572085, 61170124, 61272005), Provincial Natural Science Foundation of Jiangsu (Grant No. BK20151254, BK20151260), Science and Education Innovation based Cloud Data fusion Foundation of Science and Technology Development Center of Education Ministry(2017B03112), Six talent peaks Project in Jiangsu Province (DZXX-027), Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University (Grant No. 93K172016K08), and Provincial Key La-boratory for Computer Information Processing Technology, Soochow University.
This work is also supported by Research Grants Council of the Hong Kong Special Administrative Region, under RGC General Research Fund (Project No. CUHK14217516 and CUHK14201017).

1 INTRODUCTION

Coloring is one of the most important but time-consuming steps during the art creation. Creating an impressive and expressive painting requires a nice color composition and also proper usage of texture and shading. Achieving the task is not trivial, as it requires both the sense of aesthetics and the experience in drawing. Even professionals may spend a significant amount of time and effort in producing the right color composition and fine texture and shading details. An automatic or semi-automatic colorization system can greatly benefit the community. With the system, novice artists can learn how to color and texture, while experienced artists can try out

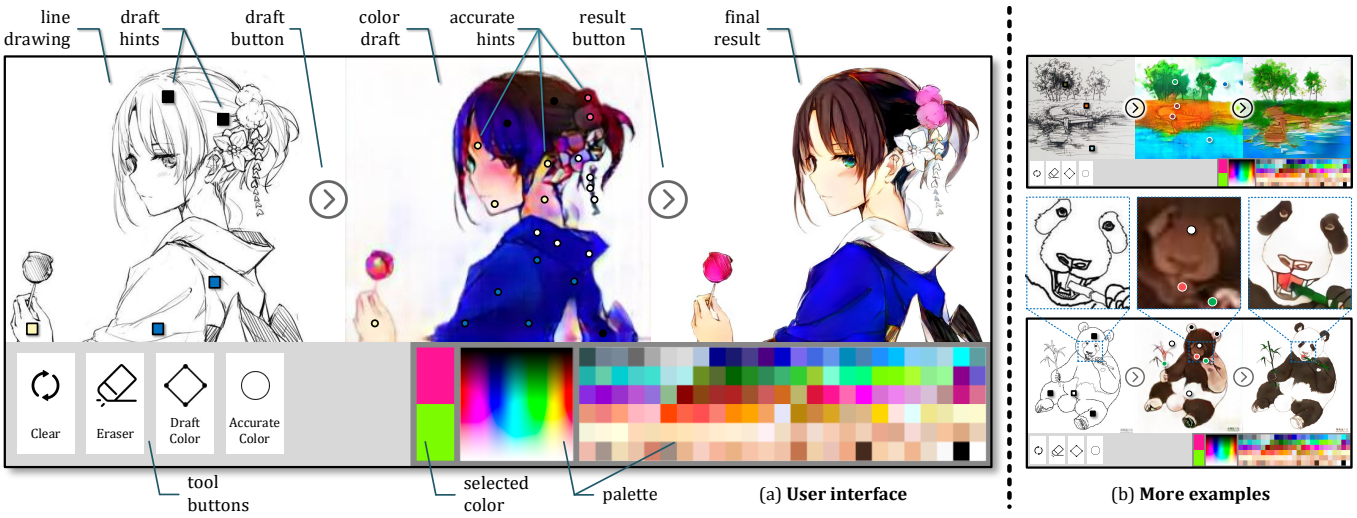


Fig. 2. **The interface of our interactive software.** Users can achieve high-quality results step-by-step through our software. They firstly make a color draft with the draft color tool. After that, they put some more hint points on the generated draft canvas to control details and achieve the final result. © ecosia (a), kumantang (all in b).

different color compositions and save much time on the tedious colorization. However, there are multiple challenges in achieving such automation. Firstly, automatic understanding of sketches is difficult due to their unlimited subjects and drawing styles. Furthermore, sketches are usually expressed in a simplified and abstract form. Texture or smooth shading cues are seldom available for guiding the colorization.

Due to the effectiveness of deep learning in image processing tasks, several learning-based methods have been proposed and developed. For examples, PaintsChainer [TaiZan 2016] and Comicolorization [Furusawa et al. 2017] provide an easy-to-use colorization frameworks and create some impressive results. However, their results often contain obvious color mistakes and artifacts, such as watercolor blurring, color bleeding and color distortion[1]. Although these methods accept user input to adjust the color, such color adjustment is usually not precise, and hurting the ability of user control. As a result, the community still prefers traditional coloring techniques over getting assistance from intelligent methods.

In this paper, we propose a semi-automatic method that allow users to precisely control over colorization on real-world sketches. No extensive input is required from the users. To achieve this, we borrow the idea of drafting from the artist painting practices. Professional artists like to make drafts before the detail painting on sketches [Hart 2015]. Drafting determines the initial color composition and bridges the simple sketch and fine painting. We hence propose a two-stage CNN-based framework for colorization accordingly. The first *drafting stage* aggressively splashes colors over the canvas to create a color draft, with the goal of enriching the color variety (Fig. 1(a)). The draft may contain mistakes and blurry textures, but is usually dressed with a rich and vivid color composition. The second *refinement stage* corrects the color mistakes, refines details and polishes blurry textures to achieve the final output. This two-stage design effectively divides the complex colorization problem into two relatively simpler and goal-clearer subtasks. With this design, the learning is more effective and the learned model is more robust in handling a wide range of real-world sketches of different content and different drawing styles. We also found that this refinement stage can also be used independently and applied to refine the results from other methods such as PaintsChainer [TaiZan 2016].

We developed a GUI software based on our model (Fig. 2). Users can paint over the sketch in a two-step manner. In the first step, users can instantly preview the global color composition of the painting (draft) with a few hints. Then they can progressively refine the local details of the painting by adding more hints in the second step. Comparing to the PaintsChainer family [TaiZan 2016, 2017a,b], our two-stage design achieves better visual quality as well as better user experiences. Besides, our software can better follow the user hints and hence more precise user control.

We validate the effectiveness of our method by testing it over a set of challenging real-world sketches with a wide range of content and styles. We also conduct a multi-dimensional user study and compare our results to state-of-the-art methods/systems from both academic and industry. The statistics shows that our method outperform existing ones in terms of the visual quality, the color vividness, and the user experience. Our contribution can be summarized as follows:

- We present a two-stage model that can achieve high-quality colorization over real-world challenging sketches.
- Our solution can robustly refine the common artifacts in existing learning-based sketch colorization, e.g., watercolor blurring and color bleeding.
- We develop an interactive software to realize our model and offer precise user control.

2 RELATED WORK

[1]Watercolor blurring refers to blurry textures as if the results are painted with a watercolor pen. Color bleeding refers to color leakage across the boundaries. Color distortion refers to wrong or distorted colors across a region.

* indicates equal contribution

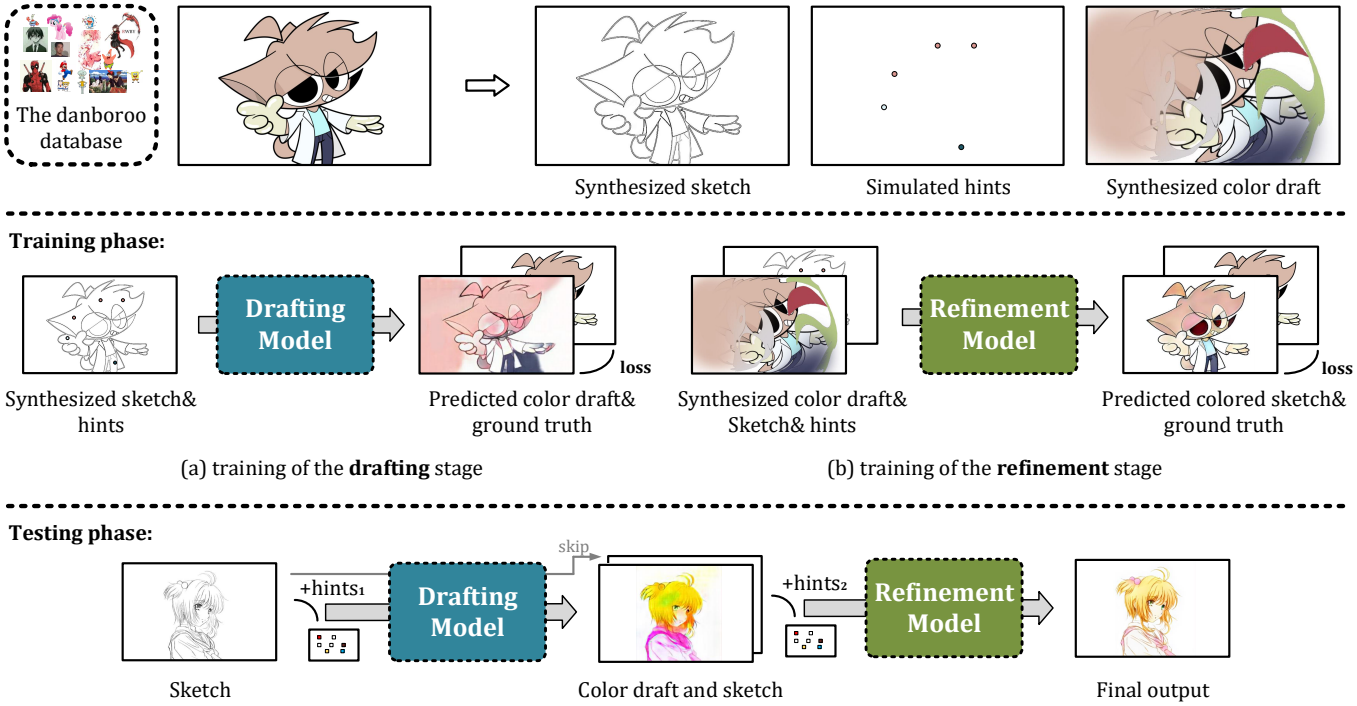


Fig. 3. **Overview of our framework,** including data preparation, the training and testing pipeline. © Senor-G (mouse), style2paints (girl).

Color Propagation. Traditional methods strongly rely on texture or gradient information to propagate the user specified colors. Qu et al. [2006] proposed to colorize the manga based on the local texture information. Lazy Brush [Sykora et al. 2009] performs a smart color flood filling. However, these methods simply colorize the whole region with a single color specified by the user. They do not automatically choose colors, nor synthesize color shading or textures to produce enriched content.

Learning-based Sketch Colorization. There exists tailormade solutions for sketch colorization. Scribbler [Sangkloy et al. 2017] learns to generate different scenes on sketches, especially bedroom, with user hints. Furusawa et al. [2017] proposed to colorize manga with a given different color palette. Deepcolor [Frans 2017] and Autopainter [Liu et al. 2017b] introduce adversarial losses to achieve better line arts colorization. Commercial product PaintsChainer [TaiZan 2016, 2017a,b] achieves state-of-the-art visual quality among these learning-based methods. It reads the sketch input and automatically creates colored painting according to the user hints. Although these learning-based methods can generate texture and color gradient, the results are not convincing due to the existence of many color mistakes and artifacts. Methods like [Frans 2017; Liu et al. 2017b] can only produce low-quality results with obviously mistake color and texture. PaintsChainer V1 [TaiZan 2016] has the watercolor problem and lack of color vividness, while PaintsChainer V2 [TaiZan 2017b] has the problem of color bleeding and dull texture. PaintsChainer V3 [TaiZan 2017a] improves the clearness and sharpness of the painting but suffers from the color bleeding, and the line and color distortions. In contrast, our framework is capable of detecting, refining mistakes and minimizing these artifacts.

In addition, an off-line coloration method [Hensman and Aizawa 2017] is proposed. The method requires an off-line training for every single manga colorization task. Paired color reference for every manga is also needed, which is relatively hard to collect. In the field of sketch processing, [Simo-Serra et al. 2018a,b, 2016] are proposed to simplify sketches, while [Li et al. 2017] is proposed to extract structural lines from mangas.

Image Style Manipulation. Style transfer is also widely used in colorization tasks. Neural style transfer [Gatys et al. 2016] can colorize images by transferring low-level colored features to grayscale images. Meanwhile, deep image analogy [Liao et al. 2017] can also be applied for colorization, by matching correspondences between the colored and the grayscale image. Some style transfer methods are designed for colorizing sketches, such as anime style transfer [Zhang et al. 2017a] and Comicolorization [Furusawa et al. 2017]. They can deal with sketches to certain degree, but the results are not so vivid. We believe one major reason, that current style transfer methods are inferior to colorize sketches, is probably due to the fact the VGG classifier [Simonyan and Zisserman 2014] used in these methods are trained with natural images, instead of line drawings in our case. In addition, precise user control another major drawback of the style transfer methods due to their limited user interaction.

Generic image-to-image translation methods can also be used for colorization, such as paired image translations [Chen and Koltun 2017; Isola et al. 2017; Wang et al. 2018b,a] and unpaired image translations [Bousmalis et al. 2017; Donahue et al. 2017; Kim et al.

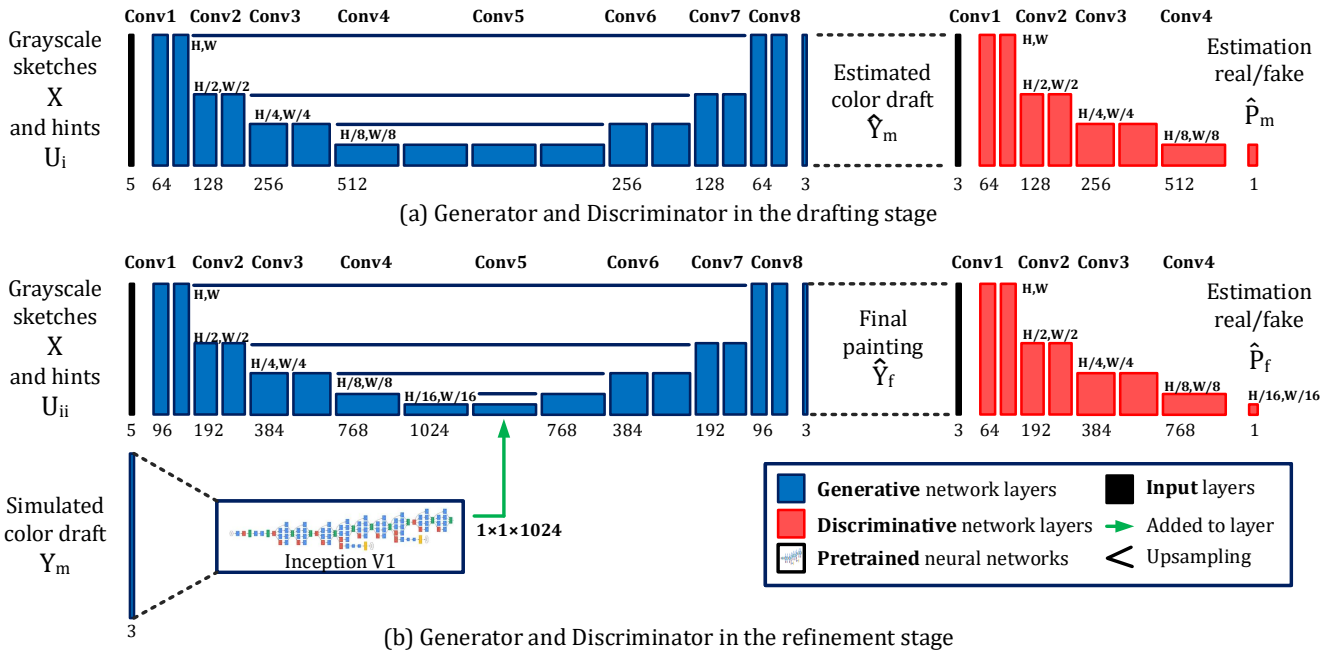


Fig. 4. **Network architectures** of all mentioned models. An $1 \times 1 \times 1024$ vector is extracted from ImageNet Inception [Szegedy et al. 2015] after its first global average pooling. All scale transformations are implemented by subsampling and upsampling. The *add* operation is done by matrix broadcasting. All weights are trainable. Rectified Linear Units [Nair and Hinton 2010] are used as activations. We do not use normalization layers (except for pretrained Inception).

2017; Liu et al. 2017a; Yi et al. 2017; Zhu et al. 2017]. These methods show the potential of transforming images across categories, e.g. *maps to aerials*, *edges to cats*, and in our case, *sketches to paintings*. However, in real-world sketch colorization, both high-frequency features are sparse and low-frequency semantics are noisy, applying these generic methods to sketch colorization is not satisfactory. The resultant color is usually dull, and artifacts like watercolor blurring and color bleeding remain. We compare these methods to ours in later section.

Photo Colorization. Several optimization-based methods [Levin et al. 2004] and learning-based methods [Iizuka et al. 2016; Larsson et al. 2016; Zhang et al. 2016, 2017b] have been proposed to successfully colorize grayscale photos. They can create very realistic color photos even without known priors. However, these methods cannot be applied to our sketch colorization application, because they heavily rely on texture and gradient information existing in the photos. In our case, both texture and gradients seldom exist. Instead our goal is to synthesize the texture and shading, in addition to the color.

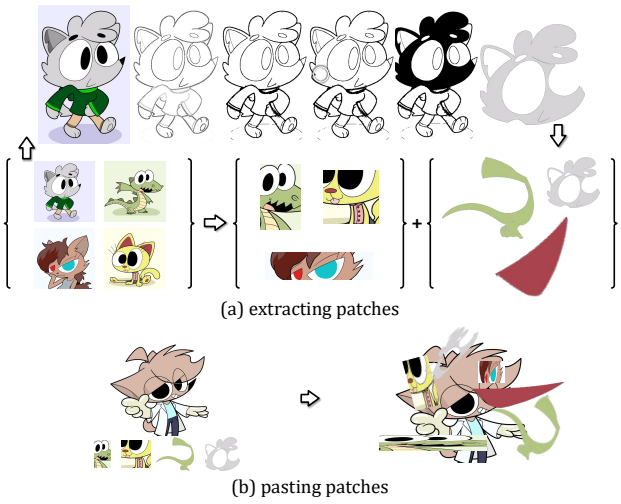
3 OVERVIEW

The two-stage framework consists of the *drafting* and the *refinement stages*. Given an input sketch and the initial set of user hints, the drafting stage is trained to determine the color composition and predict a color draft with the goal of painting vividly. After that, the refinement stage identifies incorrect color regions and refines them with an additional set of user hints. The two models behind the two stages are trained independently. During the testing phase, the two stages are connected together to generate the final output. Fig. 3 illustrates the framework. This two-stage design narrows the gap between sketch and painting, by dividing the complex colorization task into two simpler and goal-clear subtasks, drafting and refinement. This can simplify the learning and raise the quality of colorization results. On the other hand, existing single-stage colorization methods are less capable of identifying unnatural coloring and fixing the artifacts, due to the learning difficulty.

To prepare the training data, we use the publicly available Danboroo database [DanbooruCommunity 2018] as the painted images in training data. The corresponding sketches in the training data are extracted from the paintings, using the line extractor from PaintsChainer [TaiZan 2016]. To simulate discrete point-wise user input, we adopt the method described in [Zhang et al. 2017b]. The neural network models for both the drafting and the refinement stages are essentially generative adversarial networks (GANs) [Goodfellow et al. 2014]. Fig. 3 shows their connection among the stacking layers and their dimensions. For most of all training, we use the Adam optimizer [Kingma and Ba 2014] with $\beta_1 = 0.9, \beta_2 = 0.99$ and $lr = 1e-5$. We update weights on the NVIDIA Tesla P100 card till the convergence of the models, with a batch size of 16. Training samples are randomly cropped into 224×224 patches. As all our models are fully convolutional, the proposed colorization framework supports arbitrary input size during the testing phase.

4 THE DRAFTING STAGE

In this stage, our deep network model learns to determine the global color composition of the input sketch and predicts a color draft.



(a) extracting patches

(b) pasting patches

Fig. 5. **Random region proposal and pasting.** We randomly extract image pieces and paste them onto the original image, to simulate the color mistakes. © Senor-G (all images).

The network is not required to produce high-quality results, but it should follow the user hints and aggressively splash over the sketch with the goal of enriching the color variety. To achieve this, we propose a drafting network G to create the color draft prediction $\hat{y}_m$ from the sketch x and the user hints u_i. The detailed network parameters can be found in Fig. 4(a). Note that some other methods such as PaintsChainer [TaiZan 2016] may also be used to generate the color drafts. But their technical details are not disclosed. In the following discussion, we show that our simple and effective drafting model achieves the state-of-the-art performance, when compared to other learning-based industrial models, e.g. PaintsChainer, if they are used for drafting purpose.

For any sketch x and user hints u_i, we get the feed-forward prediction of $G(x, u_i)$ as $\hat{y}_m$ and optimize the network by optimizing the loss function (Eq. 1) ($\lambda = 0.01$):

$$\begin{aligned}&\arg\min_G \max_D \mathbb{E}_{x,u_i,y_f\sim P_{\text{data}}(x,u_i,y_f)}[||y_f - G(x,u_i)||_1 + \\ &\alpha L(G(x,u_i)) - \lambda \log(D(y_f)) - \lambda \log(1 - D(G(x,u_i)))]\end{aligned} \tag{1}$$

Furthermore, we introduce the *positive regulation loss* to encourage the color richness of the color draft $G(x, u_i)$,

$$L(x) = -\sum_{c=1}^{3} \frac{1}{m} \sum_{i=1}^{m} (x_{c,i} - \frac{1}{m} \sum_{i=1}^{m} x_{c,i})^2 \tag{2}$$

where $x_{c,i}$ is the i-th element on the c-th channel, m is image width $\times$ height. This loss function encourages the increase of the color variance in RGB space of the generated color draft, making the model positively generates saturated and bright colors.

5 THE REFINEMENT STAGE

However, the color draft from the previous stage is not ready for industrial use, as they may contain color mistake and artifact. To refine the painting, we have to identify the problematic regions and fix them. Moreover, we need to provide users a precise control over the local details. To achieve this, we propose another deep neural model which learns to detect and refine the problematic color regions. The model reads a new set of user hints to introduce new colors to the draft. Simultaneously, incorrect colors and artifacts in the draft are detected and removed.

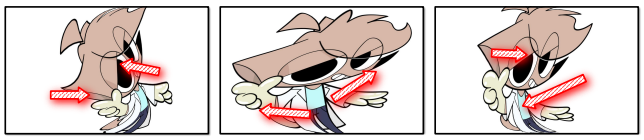


Fig. 6. **Random transform.** We apply random transform to the image to simulate the blurry and the distorted colors. The directions of the transform are marked by red arrows. © Senor-G.

However, it is difficult to gather proper training samples for our task. One possible option is to ask experienced artists to help with refining these color drafts into near-realistic paintings, but this is tedious and time-consuming. Also, the diversity of the content cannot be guaranteed. Another idea is to use the prediction directly from the first stage. However, this approach is not appropriate as the second stage may be overfitted to some specific types of color drafts and hence looses its generalization ability. Even worse, using the results from the previous stage is equivalent to train the two different stages of the framework jointly. As the two stages are intentionally designed for different purposes, joint training is not an option. We demonstrate this in our evaluation and ablation study.

Instead, we introduce an automatic method to synthesize a rich dataset of {*color draft, refined painting*} pairs. The synthesis helps to improve the generalization ability of the refinement model and teaches the model to refine different kinds of artifacts. To achieve this, we first summarize the potential artifacts in a color draft. Based on observation, the major types of artifacts include:

- Color mistakes: mistakenly filled colors such as a blue sun, green human faces and so on.
- Color bleeding: color leakage to surrounding but irrelevant regions. For example, the skin color leaks into the background.
- Blurring and distortion: the results are watercolor blurred with low saturation or the regions are over-textured, messing up some structural lines.

In the rest of this section, we discuss these artifacts and propose a novel approach to synthesize a dataset with these artifacts.

5.1 Random Region Proposal and Pasting

To simulate color mistakes, as in Fig. 5, we firstly crop random rectangular patches from the color images to obtain the pieces. The sizes of the pieces are uniformly distributed over the range from 64×64 to 256×256. To increase the randomness and diversity of color mistakes, we also use region proposal methods to acquire patches of irregular shapes. The regions are extracted based on the edge map of the input image. We first calculate the difference between Gaussian blurred image and the original image. We clip the results to obtain a sharp and clean edge map. Then we clip the obtained map, compute the average value of the map and use it as a threshold to obtain the binarized edge. Finally, we implement a

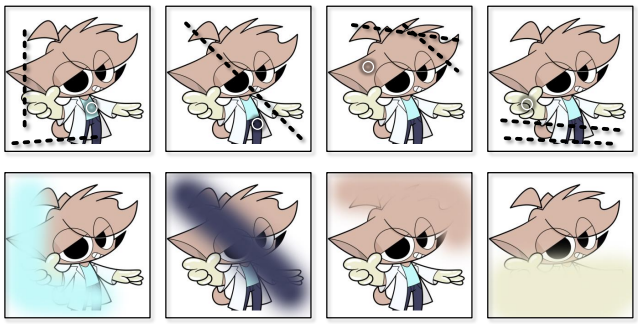


Fig. 7. **Random color spray.** We randomly spray colors to an image to simulate the color bleeding artifact. The sampling points and paths are marked in circles and lines. © Senor-G.

trapped-ball segmentation [Hensman and Aizawa 2017] to obtain the color region mask for extracting the pieces with irregular shapes. With the combination of these two methods, we extract altogether 10,000 distinct image pieces. We randomly rotate and paste these pieces to overlay onto the paintings in order to simulate the color mistake.

5.2 Random Transform

We use the transform to simulate the blurry and the distorted colors, as demonstrated in Fig. 6. Firstly, we generate the randomized 2×3 matrix $T_\theta(G)$, where all matrix elements θmn are normally distributed random values in the range $[0, 0.1^2]$. Then, we use the Spatial Transform Layer (STL) as described in [Jaderberg et al. 2015] to transform images. This transform can blur the local patterns and add global noises simultaneously.

5.3 Random Color Spray

We simulate the artifact of color bleeding via spraying random colors over the canvas, as in Fig. 7. Firstly, a random color is chosen from the image. Then we spray the selected color with a relatively large spread of randomly selected width r, along several random linear paths, where r is a value uniformly distributed over the range of [64,128]. The cross-sectional profile of spray is designed to mimic the color bleeding. Please refer to the supplemental material for details of the spray texture.

5.4 Model Optimization

The model architecture and parameters can be found in Fig. 4. We synthesize the erroneous color drafts y_m by simultaneously applying the three methods mentioned above. The sketch x, user hints u_{ii} and the synthesized color draft y_m are regarded as inputs, and $\hat{y}_f$ are the outputs. The ground truth painting y_f is regarded as the label. We train our model with mean absolute error (MAE) and the adversarial loss. We use the pretrained ImageNet [Fei-Fei 2010] inception V1 [Szegedy et al. 2015] as the initial weights of the encoder for synthesized color draft y_m. Combining all loss functions, the final optimization can be written as ($\lambda = 0.01$):

$$\begin{aligned} &\arg\min_G \max_D \mathbb{E}_{x,u_{ii},y_m,y_f\sim P_{\text{data}}(x,u_{ii},y_m,y_f)}[-\lambda \log(D(y_f)) \\ &- \lambda \log(1 - D(G(x,u_{ii},y_m))) + ||y_f - G(x,u_{ii},y_m)||_1] \end{aligned} \tag{3}$$

6 EVALUATION

To evaluate the performance of our method, we first prepare a testing set. The set consists of 53 real-world sketches from various artists collected from the internet. The content of the sketches ranges from human character, animal, plant, to landscape. The test set is never used in training our model. To avoid similar images appear in the training data, we go through all testing data and remove top 3 similar training images for each testing image. The similarity is computed by scaling the testing and training images to the same size and calculating the Mean Square Error (MSE). These similar images are all removed from the training session.

6.1 User Interface

For building a more convenient painting environment, we design a user interface that assists the users to do the two-stage colorization as in Fig. 2. The user interface includes three canvases, a toolbar, and a color picker. The three canvases are for input sketch, color draft, and final result. Different from previous methods, e.g. [Zhang et al. 2017b], our method presents results from the two stages separately and accepts hint inputs for both stages. In our study, this can speed up the colorization procedure and improve the user experience. We discuss more about the design in the later section.

6.2 Visual Comparison

We first visually compare the results from our method to that of other methods. As some methods are trained with scribble hints while others are trained with point hints, we manually sample from a continuous colorization hint and create the proper hint map for each method. To have a fair comparison, the identical hint maps are fed to our both stages. Note that, using different hint maps can further improve the quality of our results.

Color Propagation Methods. We compare our method to TVPaint 11 LazyBrush [Sykora et al. 2009], which is a trapped-ball flood-filling method based on color propagation and edge detection. Their results are shown in Fig. 8. We can see the method can only lay plain colors on the canvas without any spatially varying shading and texture. Also, as it is based on simple flood-filling, it may cause a severe color-bleeding problem. For example, the snake in Fig. 8(c) is unexpectedly filled with solid black color. The headwear of the girl in Fig. 8(e) exhibits the same problem. In comparison, our method can generate proper texture and shading according to the user hint, with much more appropriate filling according to the context.

Learning-based Sketch Colorization Methods. Next, we compare our method to learning-based colorization methods tailored for sketches. Firstly, we compare to Comicolorization [Furusawa et al. 2017], an automatic neural model to colorize manga images with no input hint. It is apparent that their method fails to colorize sketches. It mainly paints random grayness over the sketches. Next, we compare to the state-of-the-art industrial systems, PaintsChainer (V1 to V3) [TaiZan 2016, 2017a,b]. Their results are shown in the columns

Fig. 8. **Visual comparison.** We compare the colorization results on a simple character (a), a male character with a detailly drawn animal, and (b) a girl in a dress of complex pattern and fine decoration (d). Note that Comicolorization [Furusawa et al. 2017] is fully automatic and require no user input. For our method, we use the same hint for both stages. © xueshiwang (a), style2paints (b), KimonoAoi (d).

Fig. 9. **Results of automatic colorization**. (a) PaintsChainer V2; (b) PaintsChainer V3; and (c) uurs. No human hints are provided. Zoom for details. © karen, style2paints.

5 to 7 in Fig. 8. All methods from the PaintsChainer family suffers from the color bleeding problem. The color of the snake leaks to the background (Fig. 8(b)) and the color of the girl's headdress (Fig. 8(e)) leaks as well. Besides, PaintsChainer V1 [TaiZan 2016] has the watercolor blurry problem. The results are blurry with low saturation colors. PaintsChainer V2 [TaiZan 2017b] improves the color vividness but is not satisfactory in generating textures. For example, the snake in 8(b) is colored with less texture than the previous version. The hair of the girl in 8(d) is colored with nearly solid colors without fine texture and shading. PaintsChainer V3 [TaiZan 2017a] has improvement on painting texture and shading to a certain degree, but the details and lines are distorted. Some detail regions are colorized with oversaturated and distorted colors, like the headdress in Fig. 8(e). In comparison, our method identifies suitable regions according to the user hint and colorize the regions with the sense of layering and highlights. In Fig. 8(a,b,d), the hairs and clothes are

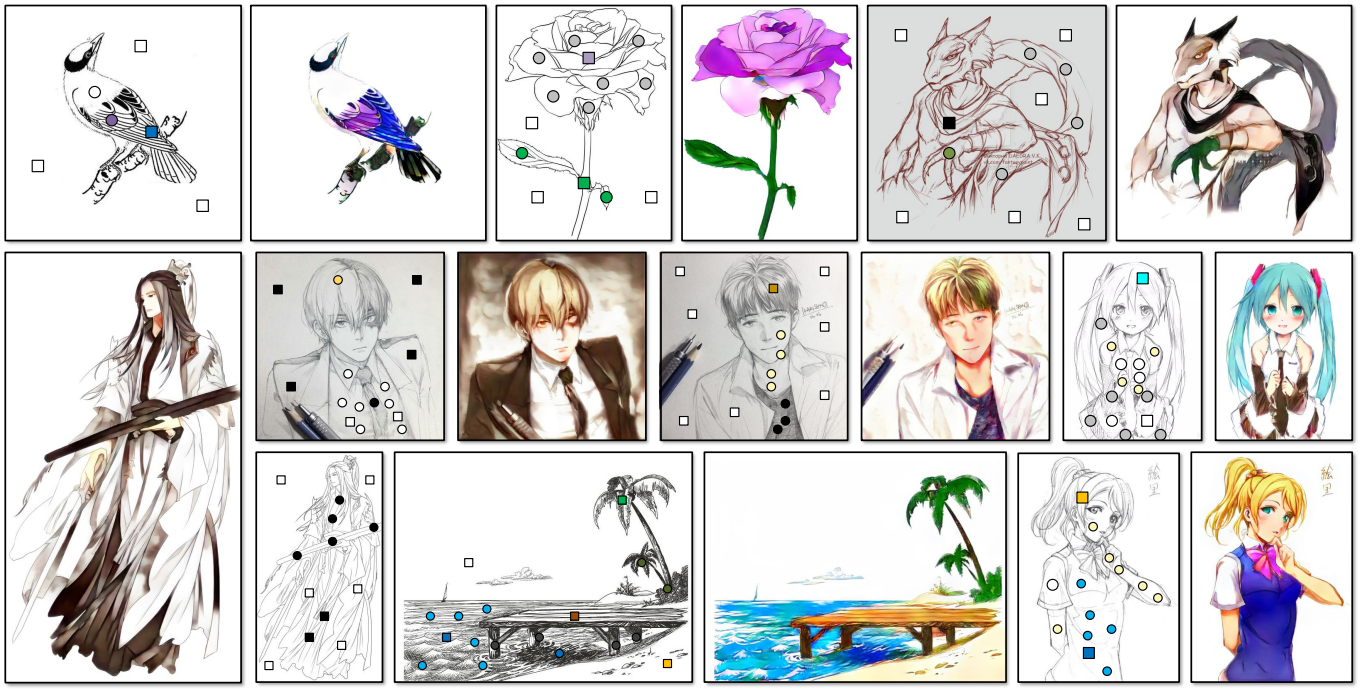


Fig. 10. **Results on real-world sketches** gathered from our user study. All sketches are from real artists. We use these real-world results to demonstrate the generalization ability of our model. The squared color hints are from the drafting stage and circular hints are from the refinement stage. © left to right: kumantang, kumantang, VictoriaDAEDRA (DeviantArt), shinji (DeviantArt), shinji (DeviantArt), Perez, style2paints, kumantang, style2paints.

vividly colored with consistent and smooth highlights and shadows. We show a challenging case in Fig. 8(c), where all existing methods fail to colorize the head of the snake properly. Our method successfully colorizes the snake with a clear and mottled texture. Furthermore, with the refinement stage, our results are free from the color bleeding problem.

Here, we discuss a special scenario, automatic colorization. Learning-based colorization methods can colorize without any human input and is useful when there is no particular preference in colorization, or the user simply wants to get some inspiration before painting. In Fig. 9, we present a comparison between PaintsChainer V2, V3 and ours on colorization of male and female characters on automatic colorization. We observe the same color bleeding problem in Fig. 9(a),(b). Due to the absence of human inputs, we also find wrong or oversaturated colors in their results. Benefiting from the refinement stage, our framework has a chance to refine the details and thus generate visually better results with fewer artifacts, which are shown in Fig. 9(c).

Image Style Manipulation Methods. Style transfer approaches are also capable of colorizing sketches by applying styles from the reference (style) image. In Fig. 12, we present a comparison between our proposed method to four style transfer methods: neural style transfer [Gatys et al. 2016], deep image analogy [Liao et al. 2017], anime sketch colorization [Zhang et al. 2017a], and Comicolorization [Furusawa et al. 2017]. From the results, all state-of-the-art style transfer methods are not satisfactory for sketch colorization. Deep image analogy tries to apply the color distribution in the style image but causes incomplete coloring in their results. The method also exhibits the color bleeding problem. [Gatys et al. 2016] and [Zhang et al. 2017a] fail to create visually pleasing colorization results because they basically fill the reference image colors randomly over the sketches. [Furusawa et al. 2017] performs better by successfully colorizing the hairs, but it still lacks vivid color and proper shading. The clothes are mostly ignored during the colorization. In contrast, our method achieves more reasonable results by color hint sampled from the style images. Even though our method is semi-automatic and may not cover all the colors in the style image, our method still obtains high quality outputs with better obedience to the color composition of the reference image.

Image-to-image translation methods such as Pix2Pix [Isola et al. 2017] can also translate sketches into paintings. We present a comparison between ours and [Isola et al. 2017] in Fig. 13. As demonstrated, [Isola et al. 2017] can only generate paintings with relatively low saturation and have the watercolor problem.

The real-time user-guided colorization [Zhang et al. 2017b] follows the idea of image translation and colorize photos given the user input. We train the system of [Zhang et al. 2017b] with our dataset and compare it to ours in Fig. 13. As [Zhang et al. 2017b] focuses on colorization of natural images, it is not suitable for sketch colorization, with noticeable wrong colors and severe color bleeding problem. Our results are significantly better in terms of texture generation and color leak-proofing.

Besides a side-by-side comparison to other methods, we also demonstrate the robustness of our framework in Fig. 10, by showing

Fig. 11. **Colorization on various drawing styles.** We try to colorize a sketch consisting of abstracted eyes and complex cloth patterns. (a) Input sketch and user hints; (c) deepcolor; (e,b,d) PaintsChainer V1, V2, V3; and (f) ours. Same hints are used for fairness. © paintschainer [TaiZan 2016] comparison.

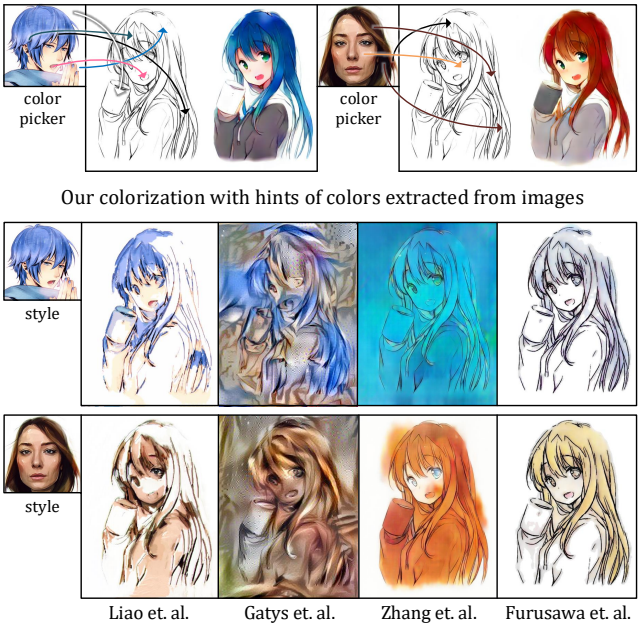


Fig. 12. **Comparison with style transfer methods.** © xiaoshou (the content image), [Liao et al. 2017] (the style images).

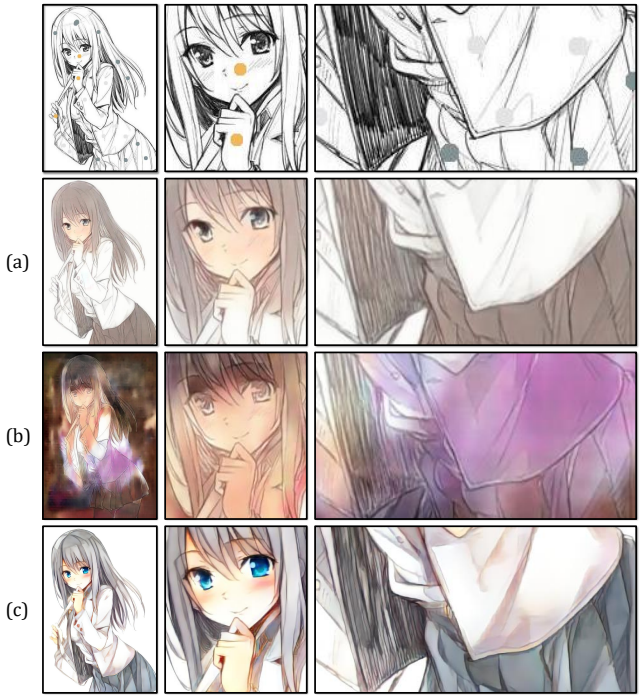


Fig. 13. **Comparison with generic image translation and photo colorization.** Results of (a) Pix2Pix [Isola et al. 2017] trained with our dataset; (b) user-guided colorization [Zhang et al. 2017b] trained with our dataset; and (c) ours. © Ruoqiyinv-G.

a gallery of our results over a rich variety of real-world sketches including, male and female characters, landscape, animal, flower, and even imaginary creature. These sketches have different subjects and drawing styles, but our model can still robustly colorize them. Fig. 24 shows another gallery of results done by professional artists.

Another challenge in real-world sketches colorization is to understand various drawing styles. Different artists have different drawing habit and may draw the same character and object in an entirely different style. For example, some artists prefer to draw very detailed and textured eyes, while others prefer only simple circles to represent eyes (Fig. 11) and intentionally to ignore to draw the pupil. Such habit is quite common among Asian artists. At the same time, some artists like to draw complex patterns on the sketch, which may confuse the neural networks. The diverse drawing styles increase the difficulty of generating proper texture and gradients. Fig. 11 shows that our competitors fail to render the color of the eyes and unable to generate texture over detailed-drawing areas, while our method can deal with all these complications satisfactorily. Our robustness is accounted by the introduction of the two-stage design, which separates the complex colorization task into two relatively simpler and goal-clear tasks. This effectively reduces the complexity of colorization. It allows the refinement stage to focus only on the refinement of detailed color and texture. The refinement stage also earns a second chance to improve the output quality. Hence, our model can robustly handle these tricky drawing.

6.3 User study

We first conduct a user study to validate the effectiveness of our framework quantitatively. To evaluate the user experience and satisfaction of our model, we invite 10 participants to use our GUI software in an interactive colorization session. Each participant is given a simple tutorial and asked to colorize a set of 5 randomly selected sketches, using our method and three competitors (the PaintsChainer family). They can use any hints. We also record the

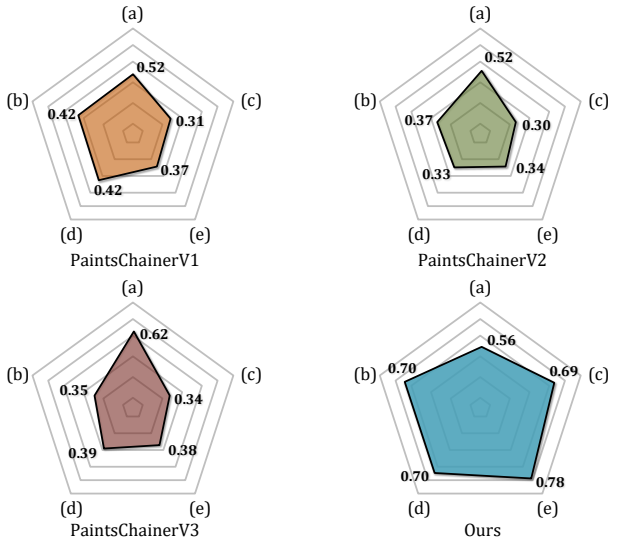


Fig. 14. **Visualization of our user study.** (a) Timing; (b) user experience; (c) regional obedience; (d) color obedience; and (e) visual quality. We here present the normalized average scores.

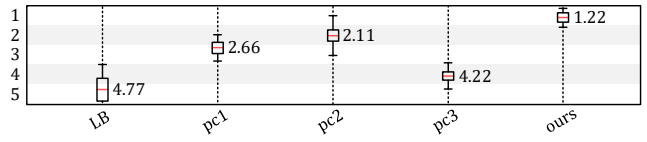


Fig. 15. **Comparison of regional obedience.** We compare the average ranking of regional obedience among LazyBrush [Sykora et al. 2009], the PaintsChainer family [TaiZan 2016, 2017a,b], and ours.

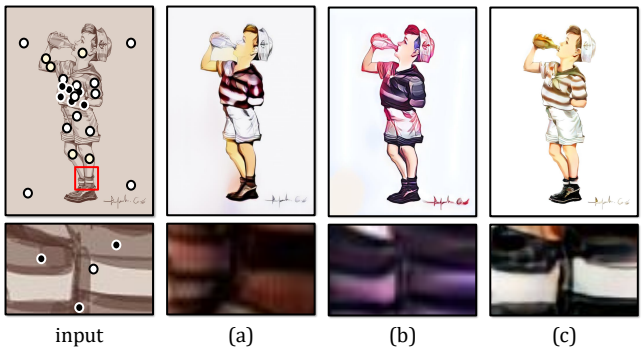


Fig. 16. **Visual comparison of color and regional obedience.** (a) PaintsChainer V2; (b) PaintsChainer V3; (c) ours. In a small and delicate area, a white color hint is surrounded by three black color hints. Our model can properly spread the color without introducing color distortion. © Golb (DeviantArt).

time used for colorizing each sketch for each participant. Then a multi-dimensional survey is conducted (Fig. 14). Five dimensions of scores in the range of [0,1] are scored by the participants in this survey. To validate the significance of the difference among different methods, we also conduct a paired student's T-test. The results are shown in Table 1. More details can be found in the supplementary document.

Visual quality. Participants are asked score their own colorization work. To calibrate their scoring, we show them several sample paintings with scores attached.

Timing. We record the colorization time of every sketch and use a tailored function described in the supplementary document to normalize the duration to the scale of [0, 1].

User experience. Next, we ask them to score their user experience during the colorization process. To calibrate the scoring, we present them a table of evaluation rules of user experiences.

Regional obedience. We ask the participants to evaluate whether a user hint can accurately identify the region to colorize. When multiple color hints are entered, an effective model should distinguish and colorize the corresponding regions accordingly. This metric also implicitly evaluates how well the model can avoid color bleeding and fusing problems.

Color obedience. This evaluates if the model can accurately follow the gamut and color tones of the user hints. When the participant enters a red hint, an effective model should fill the region with a bright red color, instead of orange or purple one.

According to the statistics, our results have a better visual quality and create more vivid paintings than that of the PainsChainer Family. Moreover, our model is more likely to obey users hint with significantly less color bleeding problem. One drawback is that our model consumes more time than our competitors. We believe this is also due to the effectiveness of our ability in precise coloring control.

T-test	Time	UE	RO	CO	VQ
PC1 VS. PC2	0.979	0.474	0.538	0.779	0.62
PC1 VS. PC3	0.201	0.297	0.659	0.580	0.91
PC2 VS. PC3	0.253	0.805	0.236	0.360	0.46
Ours VS. PC1	0.559	**0.0001**	**8e-7**	**3e-9**	**1e-9**
Ours VS. PC2	0.616	**7e-6**	**2e-8**	**1e-10**	**5e-12**
Ours VS. PC3	0.357	**1e-7**	**1e-8**	**5e-10**	**3e-14**

Table 1. **Results of the paired student's T-test.** We use paired student's T-test to measure whether the differences between methods are significant. Scores lower than $1e-2$ are bolded. UE, RO, CO, VQ refer to user experience, regional obedience, color obedience and visual quality accordingly.

As our model is more effective in controlling the fine detail coloring, the participants tend to keep fine-tune the colorization until they are satisfied. While our competitors are less effective in fine detail coloring, participants tend to give up earlier, and hence shorter time in coloring. To illustrate the significance of user hint obedience, we present a visual comparison in Fig. 16. Both PaintsChainer V2 & V3 fail to follow the user hint. Also, their colorization results are blurry and are unable to follow the boundaries. In contrast, our method can distinguish different regions and accurately colorize them according to the user hint.

To further evaluate the regional obedience, we conduct a separate experiment dedicated on region obedience. This time we include

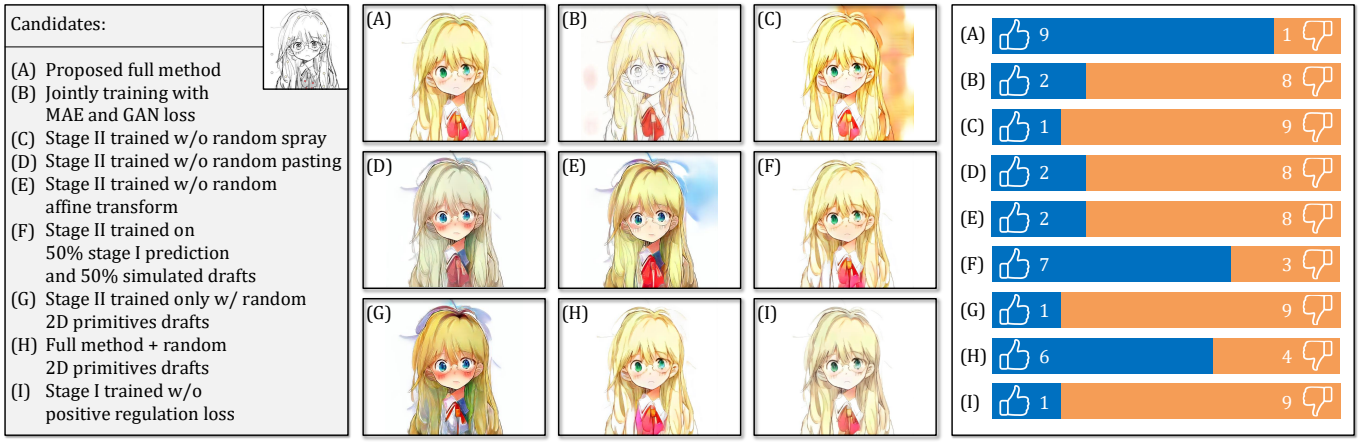


Fig. 17. **Ablation Study**. To seek for a decent training configuration, we train several colorization pipelines with the configurations on the left column. All involved 90 results can be found in the supplementary material. © Zodiac.

Fig. 18. **Correction with unnatural colors.** We ask users to produce color draft with unnatural colors (upper row). Without further color hints given to the refinement stage, our refinement model can automatically correct the unnatural facial colors. The corresponding final results are shown in the lower row. © Ravinage.

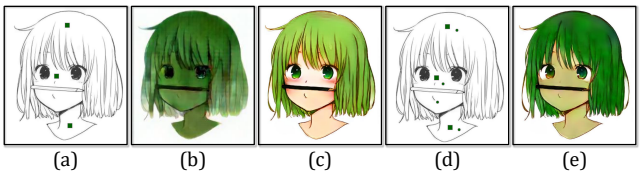


Fig. 19. **Colorization with unnatural colors.** If the user insists to colorize with unnatural color and repetitively provide such color hint in both stages (d), the model will follow the user hint and generate result as (e). The squared color hints are from the drafting stage and circular hints are from the refinement stage. © Ravinage (DeviantArt).

one more competitor, LazyBrush [Sykora et al. 2009] (it is not included before as its colorization quality is too bad). The participants are asked rank all 5 methods in terms of the accuracy of region obedience. The best is ranked number 1, while the worst is number

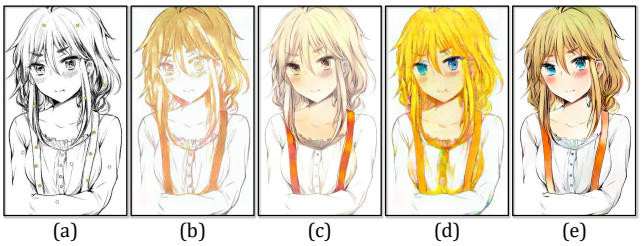


Fig. 20. **Impact of the positive regulation loss (PRL)**. (a) The input; (b) color draft without PRL; (c) final result without PRL; (d) color draft with PRL; and (e) final result with PRL. © Henjo.

5. Fig. 15 plots the average ranking scores. Our method achieves the highest rank in terms of region obedience.

6.4 Discussions and Limitation

Here, we discuss the effectiveness and significance of different components of our method.

Positive Regulation Loss. We first demonstrate the significance of the positive regulation loss (PRL) in our drafting model. Fig. 20 visualizes its effect. Without this regulation, the drafting model tends to predict the average colors of the dataset, resulting in low-saturation output. Although the refinement model can remove incorrect colors and refine the details, the global color tone remains dull. The positive regulation loss helps to improve the color vividness and diversity in the drafting stage, and leads to richer color composition in the final results.

Alternatives for Drafting. As mentioned above, the drafting model of our colorization framework can be replaced by existing colorization methods. Here, we utilize the three generations of PaintsChainer (V1-V3) to replace our drafting model, and form three alternative combinations, (PaintsChainerV1 + our refinement), (PaintsChainerV2 + our refinement), and (PaintsChainerV3 + our refinement). To evaluate the performance of such combinations, we visually

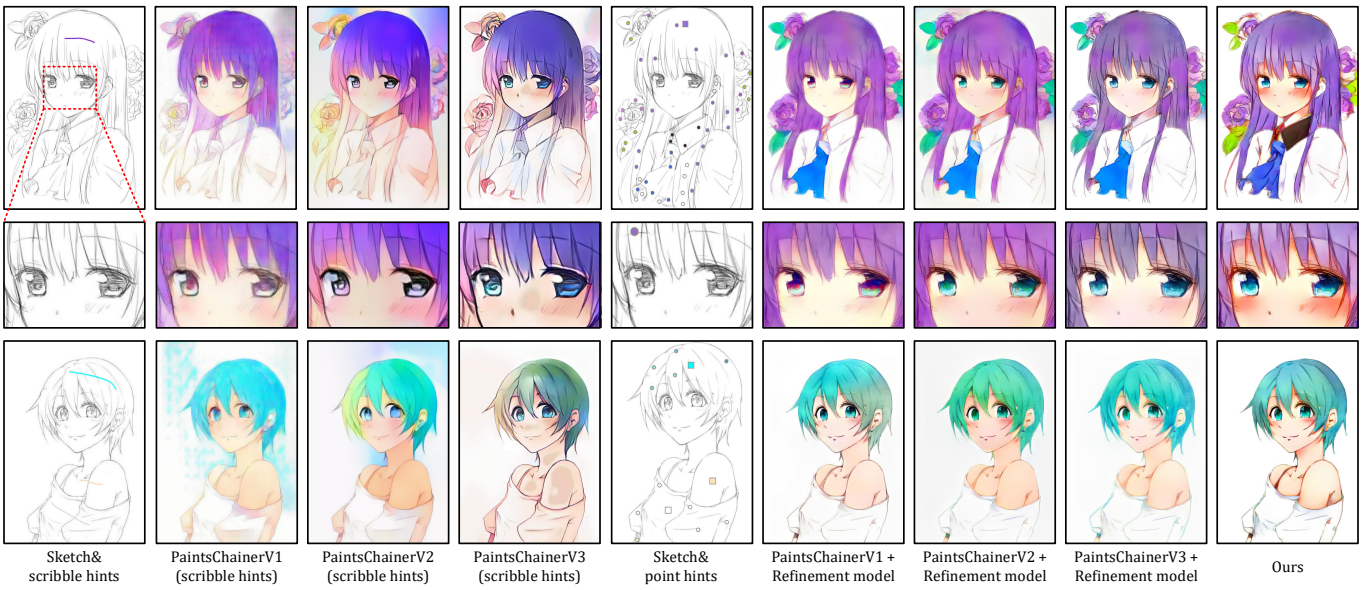


Fig. 21. **Results of combining existing colorization model and our refinement model.** PaintsChainer v1-v3 can replace our drafting model and work together with our refinement model, to achieve closer performance as our proposed model. © xNamii, xueshiwang.

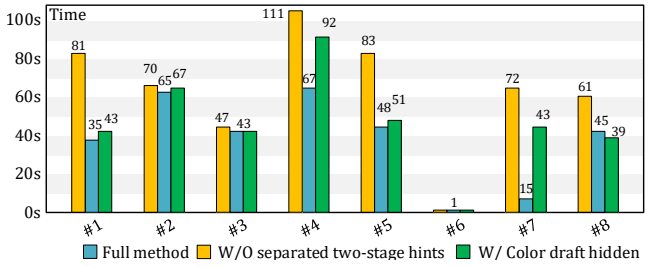


Fig. 22. **Time used for colorization with different UI settings.** #1-#8 are 8 different sketches.

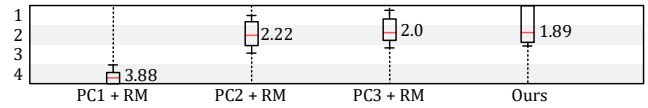


Fig. 23. **Average ranking of the alternative combinations.** "PC" refers to PaintsChainer, while "RM" refers to our refinement model.

compare them to ours in Fig. 21. Thanks to the robustness of our refinement model, all three combinations can achieve comparable performance as our original model. Both hair and the body in these results are properly shaded, with only slight difference in colors. We also conduct a user study of ranking of different combinations. The result (Fig. 23) also confirms such comparable performance. All combinations are much closer to our model in ranking. This is because PaintsChainer V1 - V3 also try to improve the color vividness and diversity, but just not quite able to refine the details and fix incorrect coloring. While the PaintsChainer family has not disclosed its technical details, our drafting model is revealed in this paper, and reproducible. Through this experiment, we also find that our refinement model can be executed independently as a color refinement tool, and cope with other existing colorization methods to achieve high-quality results.

Training Configurations. Given the unstable nature of GAN training and the complexity of our color draft simulation, a decent training configuration is critical to produce high-quality results. To study the contribution of each component in the training pipeline, we perform an ablation study to evaluate 9 different training configurations.

These configurations include leave-one-out tests and those specially designed experiments. In particular, to evaluate the importance of data preparation techniques in the refinement stage training, we introduce another baseline method by simulating color mistakes using simple 2D shape primitives (such as ellipses, triangles, and boxes, as in the embedded figure) of random colors. We also test with joint training configuration, and refinement stage training with 50% of drafting stage output. A complete list of the training configurations are listed in the left panel of Fig. 17. To evaluate the performance of each configuration, we ask 7 participants to colorize 10 sketches. Participants get the visual feedback from all configurations and stop the colorization once they find any satisfying results. We ask the participants to answer if the colorization is acceptable and gather the statistics for each configuration. The final score is shown in the right subfigure of Fig. 17. More study details are revealed in the supplementary document.

According to the result, configurations A, F and H receive much more preference than others. The preferred configurations all ensure the diversity of simulated mistakes when preparing training

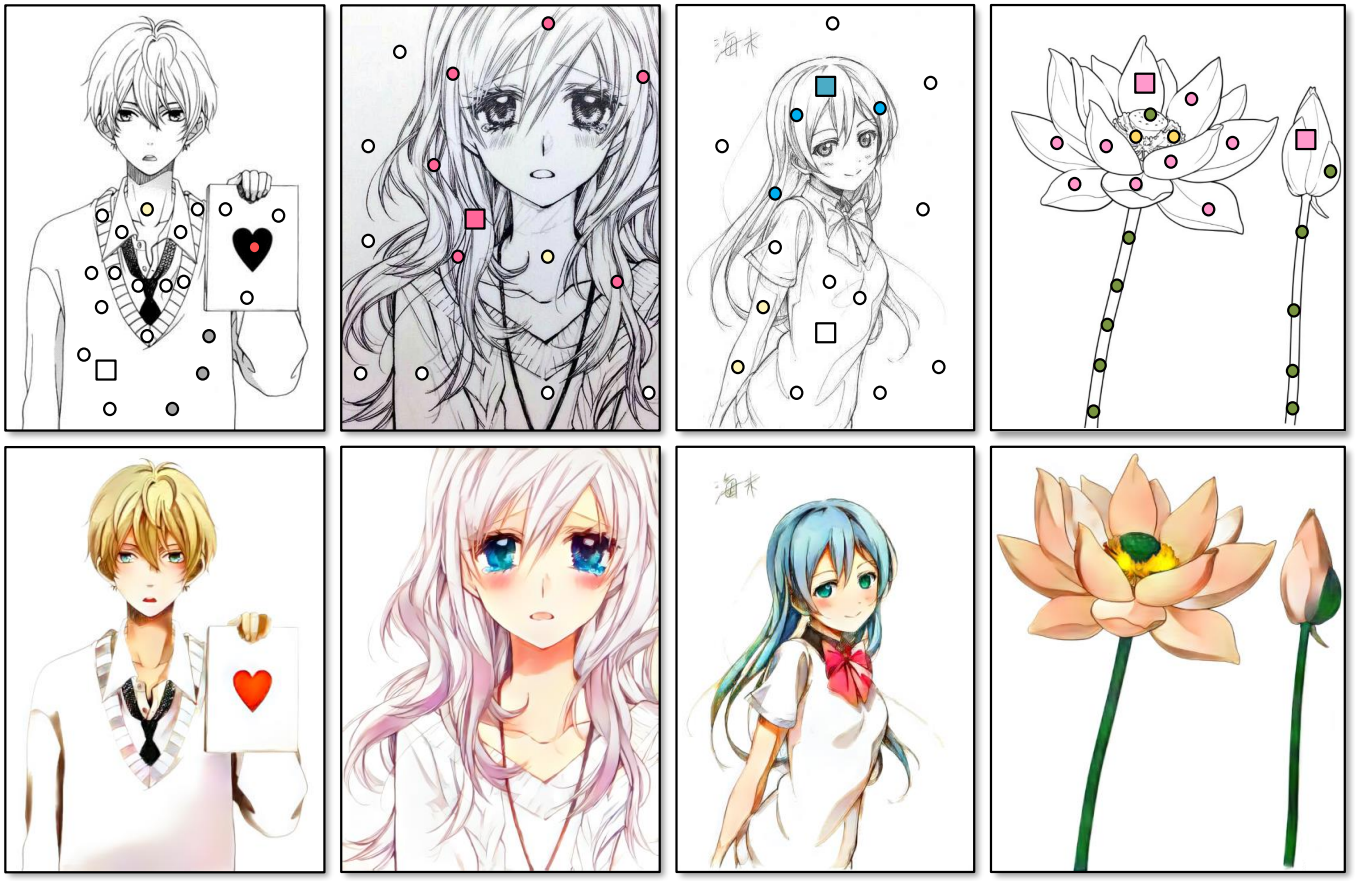


Fig. 24. **Colorization results by professionals.** Squared color hints are from the drafting stage, while circular color hints are from the refinement stage. © wuse (huaban), arinacchi (twitter), style2paints, kumantang.

examples. On the other hand, if some components of the training are missing, the model may lose its ability to detect mistakes and finally result in color bleeding or distortion problems. We also find that, although simulating mistakes with only 2D primitives is simpler, its performance in fixing artifacts is not satisfactory, as demonstrated in Fig. 17(G). In addition, this simple 2D primitive simulation provided seems not only ineffective in improving the performance, but instead deteriorate a bit the performance, as demonstrated in Fig. 17(H) and (A).

Effectiveness of Correcting Colors. To validate the effectiveness of the color refinement in the second stage, we ask our participants to create incorrect color drafts with unnatural input to test its robustness of the color mistake recognition and correction. As shown in Fig. 18, our model can correct all unnatural colors to obtain a more sensible result. On the other hand, our model also allows users to create unnatural colors if they insist. If the unnatural skin color is given in both stages, the user hint can override the auto-correction and the region is colorized as user instructed, as in Fig. 19.

Effectiveness of Two-Stage User Interface. We believe the two-stage UI design gives the user a chance to edit global color tone and local color details simultaneously, and hence shortens the overall coloring time. To validate the significance of this two-stage UI design, we conduct a user study and ask our participants to colorize 7 randomly selected sketches with different UI settings. The results are shown in Fig. 22. We can see that the two-stage design helps to speed up the coloring process. Please refer to the supplementary document for more experimental details.

Limitations. When the sketch is very complex, messy or distorted, our method may fail to incorrectly identify image features as mistakes, and get confused in coloring these regions. For example in Fig. 25, our model occasionally tries to change some proper colors or user hints to incorrect colors. Another limitation of our method is that the color propagation is not strictly constrained as in statistical methods like LazyBrush [Sykora et al. 2009]. This can sometimes lead to the slight diverge from the user hint in the final results.

7 CONCLUSION

In this paper, we introduce a two-stage learning-based framework for sketch colorization. The two-stage effectively breaks down the complex colorization into two relatively simpler and goal-clear stages, drafting and refinement. This both reduces the learning difficulty and raises the quality of the final colorization results. Our

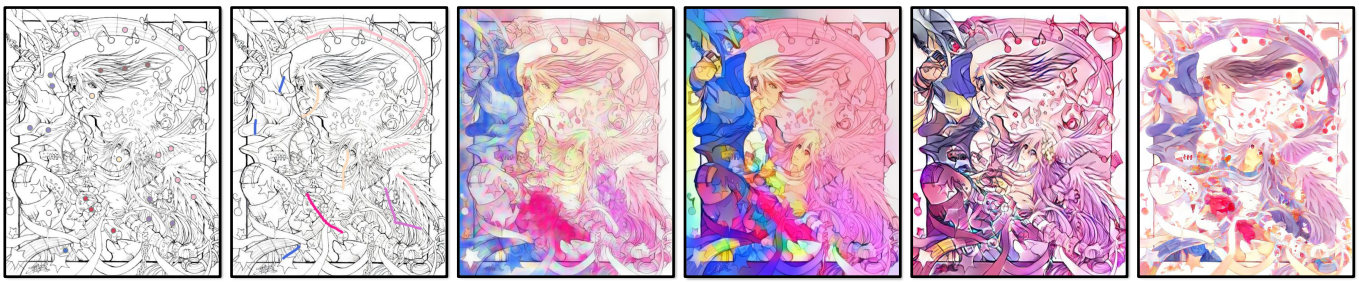


Fig. 25. **Limitation**. Our model gets confused when the content of the sketch is very complex, uncommon, and messy. In this example, the model fails to recognize the content of the sketch and rejects some proper coloring, even the user hint are given. © style2paints.

colorization system and results achieve the state-of-the-art user experience and visual quality. The proposed refinement model can work independently as a color refinement tool to fix/refine the colorization results of other colorization models. We believe this drafting+refinement approach can inspire further study along this direction in the generative modeling of sketch/painting.

REFERENCES

Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. 2017. Unsupervised Pixel-Level Domain Adaptation with Generative Adversarial Networks. *CVPR* (2017).

Qifeng Chen and Vladlen Koltun. 2017. Photographic Image Synthesis with Cascaded Refinement Networks. *ICCV* (2017).

DanbooruCommunity. 2018. Danbooru2017: A Large-Scale Crowdsourced and Tagged Anime Illustration Dataset. (2018).

Jeff Donahue, Philipp KrÃđhenbÃijhl, and Trevor Darrell. 2017. Adversarial Feature Learning. *ICLR* (2017).

L. Fei-Fei. 2010. ImageNet: crowdsourcing, benchmarking & other cool things. *CMU VASC Seminar* (2010).

Kevin Frans. 2017. Outline Colorization through Tandem Adversarial Networks. *In Arxiv* (2017).

Chie Furusawa, Kazuyuki Hiroshiba, Keisuke Ogaki, and Yuri Odagiri. 2017. Comicolorization. In *SIGGRAPH Asia 2017 Technical Briefs*.

Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. 2016. Image Style Transfer Using Convolutional Neural Networks. In *CVPR*. 2414–2423.

Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Networks. *NIPS* 3 (2014), 2672–2680.

Christopher Hart. 2015. *The Master Guide to Drawing Anime: How to Draw Original Characters from Simple Templates*. Paperback.

Paulina Hensman and Kiyoharu Aizawa. 2017. cGAN-based Manga Colorization Using a Single Training Image. *In Arxiv* (2017).

Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. 2016. Let there be Color!: Joint End-to-end Learning of Global and Local Image Priors for Automatic Image Colorization with Simultaneous Classification. *ACM Transactions on Graphics* 35, 4 (2016).

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2017. Image-to-Image Translation with Conditional Adversarial Networks. *CVPR* (2017).

Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. 2015. Spatial Transformer Networks. *NIPS* (2015).

Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim. 2017. Learning to Discover Cross-Domain Relations with Generative Adversarial Networks. *ICML* (2017).

Diederik P Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *Computer Science* (2014).

Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. 2016. Learning Representations for Automatic Colorization. In *ECCV*. Springer, 577–593.

Anat Levin, Dani Lischinski, and Yair Weiss. 2004. Colorization using optimization. In *ACM Transactions on Graphics*. ACM Press.

Chengze Li, Xueting Liu, and Tien-Tsin Wong. 2017. Deep Extraction of Manga Structural Lines. *ACM Transactions on Graphics* 36, 4 (2017).

Jing Liao, Yuan Yao, Lu Yuan, Gang Hua, and Sing Bing Kang. 2017. Visual attribute transfer through deep image analogy. *ACM Transactions on Graphics* 36, 4 (jul 2017), 1–15.

Ming-Yu Liu, Thomas Breuel, and Jan Kautz. 2017a. Unsupervised Image-to-Image Translation Networks. *NIPS* (2017).

Yifan Liu, Zengchang Qin, Zhenbo Luo, and Hua Wang. 2017b. Auto-painter: Cartoon Image Generation from Sketch by Using Conditional Generative Adversarial Networks. *In Arxiv* (2017).

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified Linear Units Improve Restricted Boltzmann Machines. *ICML* (2010), 807–814.

Yingge Qu, Tien-Tsin Wong, and Pheng-Ann Heng. 2006. Manga Colorization. *ACM Transactions on Graphics* 25, 3 (July 2006), 1214–1220.

Patsorn Sangkloy, Jingwan Lu, Chen Fang, Fisher Yu, and James Hays. 2017. Scribbler: Controlling Deep Image Synthesis with Sketch and Color. *CVPR* (2017).

Edgar Simo-Serra, Satoshi Iizuka, and Hiroshi Ishikawa. 2018a. Mastering Sketching: Adversarial Augmentation for Structured Prediction. *ACM Transactions on Graphics* 37, 1 (2018).

Edgar Simo-Serra, Satoshi Iizuka, and Hiroshi Ishikawa. 2018b. Real-Time Data-Driven Interactive Rough Sketch Inking. *ACM Transactions on Graphics* (2018).

Edgar Simo-Serra, Satoshi Iizuka, Kazuma Sasaki, and Hiroshi Ishikawa. 2016. Learning to Simplify: Fully Convolutional Networks for Rough Sketch Cleanup. *ACM Transactions on Graphics* 35, 4 (2016).

Karen Simonyan and Andrew Zisserman. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. *Computer Science* (2014).

Daniel Sykora, John Dingliana, and Steven Collins. 2009. LazyBrush: Flexible Painting Tool for Hand-drawn Cartoons. *Computer Graphics Forum* 28, 2 (2009).

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going Deeper with Convolutions. *CVPR* (2015).

TaiZan. 2016. PaintsChainer Tanpopo. *PreferredNetwork* (2016).

TaiZan. 2017a. PaintsChainer Canna. *PreferredNetwork* (2017).

TaiZan. 2017b. PaintsChainer Satsuki. *PreferredNetwork* (2017).

Chao Wang, Haiyong Zheng, Zhibin Yu, Ziqiang Zheng, Zhaorui Gu, and Bing Zheng. 2018b. Discriminative Region Proposal Adversarial Networks for High-Quality Image-to-Image Translation. *ECCV* (2018).

Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. 2018a. High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs. *CVPR* (2018).

Zili Yi, Hao Zhang, Ping Tan, and Minglun Gong. 2017. DualGAN: Unsupervised Dual Learning for Image-to-Image Translation. *ICCV* (2017).

Lvmin Zhang, Yi Ji, and Xin Lin. 2017a. Style Transfer for Anime Sketches with Enhanced Residual U-net and Auxiliary Classifier GAN. *ACPR* (2017).

Richard Zhang, Phillip Isola, and Alexei A Efros. 2016. Colorful Image Colorization. In *ECCV*.

Richard Zhang, Jun-Yan Zhu, Phillip Isola, Xinyang Geng, Angela S Lin, Tianhe Yu, and Alexei A Efros. 2017b. Real-Time User-Guided Image Colorization with Learned Deep Priors. *ACM Transactions on Graphics* 9, 4 (2017).

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. 2017. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. *ICCV* (2017).