

Guiding Users to Where to Give Color Hints for Efficient Interactive Sketch Colorization via Unsupervised Region Prioritization

Youngin Cho^{*1} Junsoo Lee^{*2} Soyoung Yang¹ Juntae Kim³ Yeojeong Park¹ Haneol Lee⁴
Mohammad Azam Khan¹ Daesik Kim² Jaegul Choo¹

¹KAIST AI ²NAVER WEBTOON AI ³Korea University ⁴UNIST

{choyi0521, sy-yang, indigopyj, azamkhan, jchoo}@kaist.ac.kr

{junsoolee93, daesik.kim}@webtoonscorp.com

kjt7889@korea.ac.kr

haneollee@unist.ac.kr

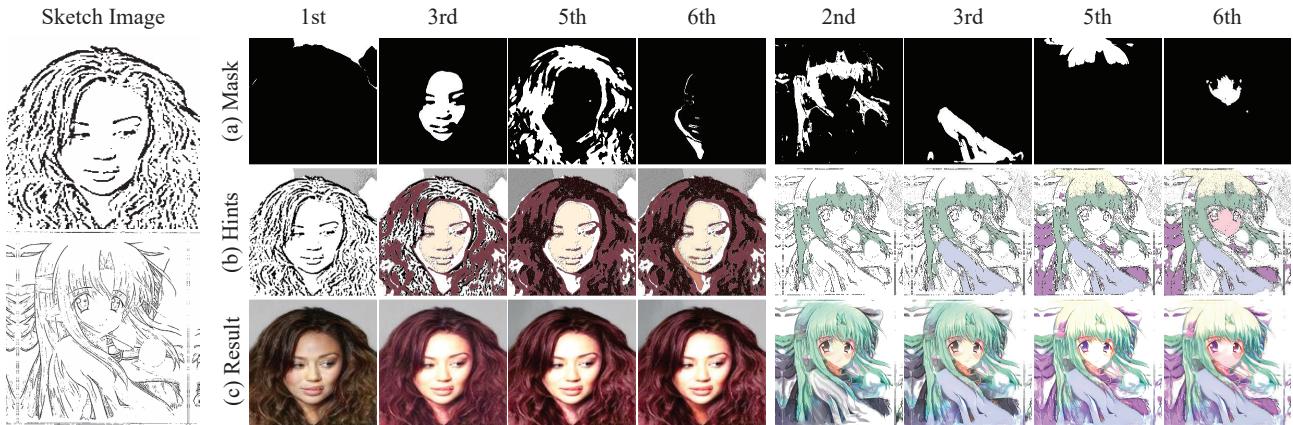


Figure 1: **Results of our proposed model on human faces and comics datasets.** Each column of (a)-(c) indicates the order of interactions as the i -th priority. (a) visualizes masked regions which our model guides at the i -th step. Given a region as (a), users select its representative color, and the region is filled with the selected color. (c) shows intermediate colorization results for given accumulated color hints as (b).

Abstract

Existing deep interactive colorization models have focused on ways to utilize various types of interactions, such as point-wise color hints, scribbles, or natural-language texts, as methods to reflect a user's intent at runtime. However, another approach, which actively informs the user of the most effective regions to give hints for sketch image colorization, has been under-explored. This paper proposes a novel model-guided deep interactive colorization framework that reduces the required amount of user interactions, by prioritizing the regions in a colorization model. Our method, called GuidingPainter, prioritizes these regions where the model most needs a color hint, rather than just relying on the user's manual decision on where to give a color hint. In our extensive experiments, we show that our approach outperforms existing interactive colorization

methods in terms of the conventional metrics, such as PSNR and FID, and reduces required amount of interactions.

1. Introduction

The colorization task in computer vision has received considerable attention recently, since it can be widely applied in content creation. Most content creation starts with drawn or sketch images, and these can be accomplished within a reasonable amount of time, but fully colorizing them is a labor-intensive task. For this reason, the ability to automatically colorize sketch images has significant potential values. However, automatic sketch image colorization is still challenging for the following reasons. (i) The information provided by an input sketch image is extremely limited compared to colored images or even gray-scale ones, and (ii) there can be multiple possible outcomes for a given sketch image without any conditional input, which tends to degrade the model performance and introduce bias toward

^{*}Equal contribution

the dominant colors in the dataset.

To alleviate these issues, conditional image colorization methods take partial hints in addition to the input image, and attempt to generate a realistic output image that reflects the context of the given hints. Several studies have leveraged *user-guided* interactions as a form of user-given conditions to the model, assuming that the users would provide a desired color value for a region as a type of point-wise color hint [36] or a scribble [24, 3]. Although these approaches have made remarkable progress, there still exist nontrivial limitations. First, existing approaches do not address the issue of estimating semantic regions which indicate how far the user-given color hints should be spread, and thus the colorization model tends to require lots of user hints to produce a desirable output. Second, for every interaction at test time, the users are still expected to provide a local-position information of color hint by pointing out the region of interest (RoI), which increases the user’s effort and time commitment. Lastly, since existing approaches typically obtain the color hints on randomized locations at training time, the discrepancies among intervention mechanisms for the training and the test phases need to be addressed.

In this work, we propose a novel *model-guided* framework for the interactive colorization of a sketch image, called GuidingPainter. A key idea behind our work is **to make a model actively seek for regions where color hints would be provided**, which can significantly improve the efficiency of interactive colorization process. To this end, GuidingPainter consists of two modules: active-guidance module and colorization module. Although colorization module works similar to previous methods, our main contribution is a hint generation mechanism in active-guidance module. The active-guidance module (Section 3.2-3.3) (i) divides the input image into multiple semantic regions and (ii) ranks them in decreasing order of estimated model gains when the region is colorized (Fig. 1(a)).

Since it is extremely expensive to obtain groundtruth for segmentation labels or even their prioritization, we explore a simple yet effective approach that identifies the meaningful regions in an order of their priority *without any manually annotated labels*. In our active guidance mechanism (Section 3.3), GuidingPainter can learn such regions by intentionally differentiating the frequency of usage for each channel obtained from the segmentation network. Also, we conduct a toy experiment (Section 4.5) to understand the mechanism, and to verify the validity of our approach. We propose several loss terms, e.g. smoothness loss and total variance loss, to improve colorization quality in our framework (Section 3.5), and analyze its effectiveness for both quantitatively and qualitatively (Section 4.6). Note that the only action required of users in our framework is to select one representative color for each region the model provides based on the estimated priorities (Fig. 1(b)). Afterwards, the

colorization network (Section 3.4) generates a high-quality colorized output by taking the given sketch image and the color hints (Fig. 1(c)).

In summary, our contributions are threefold:

- We propose a novel model-guided deep image colorization framework, which prioritizes regions of a sketch image in the order of the interest of the colorization model.
- GuidingPainter can learn to discover meaningful regions for colorization and arrange them in their priority just by using the groundtruth colorized image, without additional manual supervision.
- We demonstrate that our framework can be applied to a variety of datasets by comparing it against previous interactive colorization approaches in terms of various metrics, including our proposed evaluation protocol.

2. Related Work

2.1. Deep Image Colorization

Existing deep image colorization methods, which utilize deep neural networks for colorization, can be divided into automatic and conditional approaches, depending on whether conditions are involved or not. Automatic image colorization models [35, 25, 32, 1] take a gray-scale or sketch image as an input and generate a colorized image. CIC [35] proposed a fully automatic colorization model using convolutional neural networks (CNNs), and Su *et al.* [25] further improved the model by extracting the features of objects in the input image. Despite the substantial performances of automatic colorization models, a nontrivial amount of user intervention is still required in practice.

Conditional image colorization models attempt to resolve these limitations by taking reference images [14] or user interactions [36, 3, 34, 30, 33] as additional input. For example, Zhang *et al.* [36] allowed the users to input the point-wise color hint in real time, and AlacGAN [3] utilized stroke-based user hints by extracting semantic feature maps. Although these studies consider the results are improved by user hints, they generally require a large amount of user interactions.

2.2. Interactive Image Generation

Beyond the colorization task, user interaction is utilized in numerous computer vision tasks, such as image generation, and image segmentation. In image generation, research has been actively conducted to utilize various user interactions as additional input to GANs. A variety of GAN models employ image-related features from users to generate user-driven images [6, 15] and face images [22, 11, 27, 13, 26]. Several models generate and edit images via natural-language text [31, 20, 37, 2]. In image

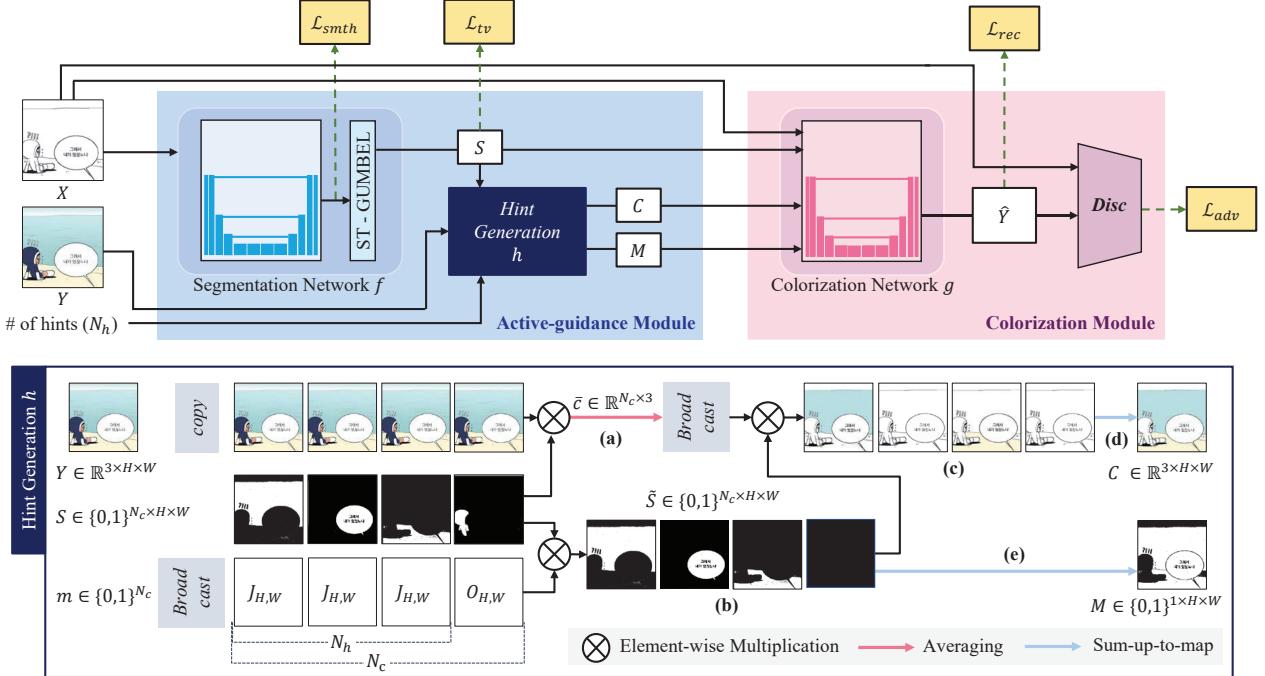


Figure 2: **Hint generation process of our proposed GuidingPainter model.** The segmentation network and the hint generation function renders colored hints (C) and condition masks (M). Based on the guidance results, our colorization network colorizes the sketch image. The example illustrates the hint generation process in the training phase where $N_h = 3$ and $N_c = 4$. First, the groundtruth image is copied as N_c times to consider each color segment at each interaction step. After element-wise multiplication with guided regions, (a) averages the color to decide representative colors for each guided region. To restrict the number of hints, we mask out the segments whose iteration step is larger than N_h . The masked results are (b). Based on (a) and (b), our module generates the colored condition for each segment as (c). In (d), we combine them into one partially-colorized image C . (e) operates as the same manner with (d) and generates the condition mask M .

segmentation, to improve the details of segmentation results, recent models have utilized dots [23, 18] and texts [8] from users. Although we surveyed a wide scope of interactive deep learning models beyond sketch image colorization, there is no directly related work with our approach, to the best of our knowledge. Therefore, the use of a deep learning-based guidance system for interactive process can be viewed as a promising but under-explored approach.

3. Proposed Approach

3.1. Problem Setting

The goal of the interactive colorization task is to train networks to generate a colored image $\hat{Y} \in \mathbb{R}^{3 \times H \times W}$ by taking as input a sketch image $X \in \mathbb{R}^{1 \times H \times W}$ along with user-provided partial hints U , where H and W indicate the height and width of the target image, respectively. The user-provided partial hints are defined as a pair $U = (C, M)$ where $C \in \mathbb{R}^{3 \times H \times W}$ is a sparse tensor with RGB values, and $M \in \{0, 1\}^{1 \times H \times W}$ is a binary mask indicating the region in which the color hints are provided. Our training framework consists of two networks and one function: seg-

mentation network $f(\cdot)$, colorization network g (Section 3.4), and a hint generation function called h (Section 3.3), which are trained in an end-to-end manner.

3.2. Segmentation Network

The purpose of segmentation network $f(\cdot)$ is to divide the sketch input X into several semantic regions which are expected to be painted in a single color, i.e.,

$$S = f(X; \theta_f), \quad (1)$$

where $S = (S_1, S_2, \dots, S_{N_c}) \in \{0, 1\}^{N_c \times H \times W}$, S_i is the i -th guided region, and N_c denotes the maximum number of hints. Specifically, f contains an encoder-decoder network with skip connections, based on U-Net [9] architecture, to preserve the spatial details of given objects.

Since each guided region will be painted with a single color, we have to segment the output of U-Net in a discrete form while taking advantages of end-to-end learning. To this end, after obtaining an output tensor $S_{logit} \in \mathbb{R}^{N_c \times H \times W}$ of U-Net, we discretize S_{logit} by applying straight-through (ST) gumbel estimator [10, 17] across channel dimensions to obtain S as a differentiable approximation. The result S satisfies $\sum_{i=1}^{N_c} S_i(j) = 1$ where $S_i(j)$

indicates the i -th scalar value of the j -th position vector, i.e., every pixel is contained in only one guided region. Here, $S_i(j) = 1$ indicates that the j -th pixel is contained in the i -th guided region while $S_i(j) = 0$ indicates that the pixel is not contained in the guided region.

3.3. Hint Generation

The hint generation function $\mathbf{h}(\cdot)$ is a non-parametric function that plays the role of simulating U based on S , a colored image Y , and the number of hints N_h , i.e.,

$$U = \mathbf{h}(S, Y, N_h). \quad (2)$$

To this end, we first randomly sample N_h from a bounded distribution which is similar to a geometric distribution formulated as

$$\mathcal{G}(N_h = i) = \begin{cases} (1-p)^i p & \text{if } i = 0, 1, \dots, N_c - 1 \\ (1-p)^{N_c} & \text{if } i = N_c, \end{cases} \quad (3)$$

where $p < 1$ is a hyperparameter indicating the probability that the user stops adding a hint on each trial. We set $N_c = 30$ and $p = 0.125$ for the following experiments.

Step1: building masked segments \tilde{S} . Given N_h , we construct a mask vector $m \in \{0, 1\}^{N_c}$ having each element with the following rule:

$$m_i = \begin{cases} 1 & \text{if } i \leq N_h \\ 0 & \text{otherwise,} \end{cases} \quad (4)$$

where m_i indicates the i -th scalar value of the vector m . Afterwards, we obtain a masked segment $\tilde{S} \in \mathbb{R}^{N_c \times H \times W}$ by element-wise multiplying the i -th element of m with the i -th channel of S as

$$\tilde{S}_i = m_i S_i, \quad (5)$$

where $S_i, \tilde{S}_i \in \mathbb{R}^{1 \times H \times W}$ denote the i -th channel of S and \tilde{S} , respectively.

Step2: building hint maps C . The goal of this step is to find the representative color value of the activated region in each segment \tilde{S}_i , and then to fill the corresponding region with this color. To this end, we calculate a mean RGB color $\bar{c}_i \in \mathbb{R}^3$ as

$$\bar{c}_i = \begin{cases} \frac{1}{N_p} \sum_j^{HW} S_i(j) \odot Y(j) & \text{if } 1 \leq N_p \\ 0 & \text{otherwise,} \end{cases} \quad (6)$$

where $N_p = \sum_j S_i(j)$ indicates the number of activated pixels of the i -th segment, \odot denotes an element-wise multiplication, i.e., the Hadamard product, after each element of S_i is broadcast to the RGB channels of Y , and both $S_i(j)$ and $Y(j)$ indicate the j -th position vector of each map. Finally, we obtain hint maps $C \in \mathbb{R}^{3 \times H \times W}$ as

$$C = \sum_{i=1}^{N_c} \bar{c}_i \tilde{S}_i, \quad (7)$$

where \bar{c}_i is repeated to the spatial axis as the form of $\tilde{S}_i \in \mathbb{R}^{1 \times H \times W}$ similar to Eq. (5) and \tilde{S}_i is broadcast to the channel axis as the form of $\bar{c}_i \in \mathbb{R}^3$ as in Eq. (6). In order to indicate the region of given hints, we simply obtain a condition mask $M \in \mathbb{R}^{1 \times H \times W}$ as

$$M = \sum_{i=1}^{N_c} \tilde{S}_i. \quad (8)$$

Eventually, the output of this module $U = C \oplus M \in \mathbb{R}^{4 \times H \times W}$ where \oplus indicates a channel-wise concatenation. Fig. 2 illustrates overall scheme of the hint generation process. At the inference time, we can create U similar to the hint generation process, but without an explicit groundtruth image. Note that a sketch image is all we need to produce \tilde{S} at the inference time. We can obtain C and M by assigning a color to each S_i for $i = 1, 2, \dots, N_h$.

To understand how the hint generation module works, recall that N_h is randomly sampled from the bounded geometric distribution \mathcal{G} (Eq. (3)) per mini-batch at the training time. Since the probability that $i \leq N_h$ is higher than the probability that $j \leq N_h$ for $i < j$, S_i is more frequently activated than S_j during training the model. Hence, we can expect the following effects via this module: i) N_h affects in determining how many segments starting from the first channel of S as computed in Eq. (4-5); therefore, this mechanism encourages the segmentation network $f(\cdot)$ to locate relatively important and uncertain regions at the forward indexes of S . Section 4.5 shows this module behaves as our expectation. ii) We can provide more abundant information for the following colorization networks $g(\cdot)$ than previous approaches *without requiring additional labels at training time or even interactions at test time*, helping to generate better results even with fewer hints than baselines (Section 4.3).

3.4. Colorization Network

The colorization network $g(\cdot)$ aims to generate a colored image \hat{Y} by taking all the information obtained from the previous steps, i.e., a sketch image X , guided regions S , and partial hints U , as

$$\hat{Y} = g(X, S, U; \theta_g). \quad (9)$$

The reason for using the segments as input is to provide information about the color relationship, which the segmentation network infers. In order to capture the context of the input and to preserve the spatial information of the sketch image, our colorization networks also adopt the U-Net architecture, the same as in the segmentation network. We then apply a hyperbolic tangent activation function to normalize the output tensor of the U-Net.

3.5. Objective Functions

As shown in Fig. 2, our networks are trained using the combined following objective functions. For simplicity, \mathbf{G} denotes the generator of our approach which contains all the procedures, i.e., \mathbf{f} , \mathbf{h} , \mathbf{g} , mentioned above while \mathcal{D} denotes training datasets.

Smoothness loss. Although adjacent pixels in an image have similar RGB values, our segment guidance networks do not have an explicit mechanism to generate segments containing those locally continuous pixels. To improve the users' ability to interpret the segments, we introduce smoothness loss, as

$$\mathcal{L}_{smth} = \mathbb{E} \left[\sum_i^{HW} \sum_{j \in N_i} \|S_{logits}(i) - S_{logits}(j)\|_1 \right], \quad (10)$$

where N_i denotes a set of eight nearest neighbor pixels adjacent to the i -th pixel, and $S_{logit}(i)$ indicates the i -th position vector of S_{logit} .

Total variance loss. In our framework, the quality of segments from \mathbf{f} is important because the hints \mathbf{U} are built based on guided regions $S = \mathbf{f}(\mathbf{X})$. Although the \mathbf{f} can be indirectly trained by the colorization signal, we introduce a total variance loss in order to facilitate this objective directly, i.e.,

$$\mathcal{L}_{tv} = \mathbb{E}_{X, Y \sim \mathcal{D}} \left[\sum_{i=1}^{N_c} \|(Y - \bar{c}_i) \odot S_i\|_F^2 \right], \quad (11)$$

where $\|\cdot\|_F$ denotes a Frobenius norm. That is, \mathcal{L}_{tv} attempts to minimize the color variance across pixels in each segment, which helps pixels of similar color form into the same segment.

Reconstruction loss. Since both a sketch image \mathbf{X} and its corresponding partial hint \mathbf{U} are built from a groundtruth image \mathbf{Y} in the training phase, we can directly supervise our networks \mathbf{G} so that it can generate an output image close to the groundtruth \mathbf{Y} . Following the previous work, we select the \mathbb{L}_1 distance function as our reconstruction loss, i.e.,

$$\mathcal{L}_{rec} = \mathbb{E}_{X, Y \sim \mathcal{D}, N_h \sim \mathcal{G}} [\|\mathbf{G}(\mathbf{X}, N_h, \mathbf{Y}) - \mathbf{Y}\|_1]. \quad (12)$$

Adversarial loss. As shown in the image generation work, we adopt an adversarial training [4] strategy, in which our generator \mathbf{G} produces a natural output image enough to fool a discriminator \mathbf{D} , while \mathbf{D} attempts to classify whether the image is real or fake. During the image colorization task, the original contents of a sketch input should be preserved as much as possible. Therefore, we leverage the conditional adversarial [19] loss, written as

$$\begin{aligned} \mathcal{L}_{adv} = & \mathbb{E}_{X, Y \sim \mathcal{D}} [\log \mathbf{D}(Y, X)] \\ & + \mathbb{E}_{X, Y \sim \mathcal{D}, N_h \sim \mathcal{G}} [\log(1 - \mathbf{D}(\mathbf{G}(\mathbf{X}, N_h, \mathbf{Y}), X))]. \end{aligned} \quad (13)$$

Finally, our objective function is defined as

$$\begin{aligned} \min_{\mathbf{G}} \max_{\mathbf{D}} \mathcal{L} = & \lambda_{tv} \mathcal{L}_{tv} + \lambda_{smth} \mathcal{L}_{smth} \\ & + \lambda_{rec} \mathcal{L}_{rec} + \lambda_{adv} \mathcal{L}_{adv}, \end{aligned} \quad (14)$$

where each λ indicates the weighting factor for each loss term. We describe the implementation details in the supplementary material.

4. Experiments

4.1. Sketch Image Datasets

Yumi's Cells [21] is composed of 10K images from 509 episodes of a web cartoon, named *Yumi's Cells*, where a small number of characters appear repeatedly. Because it was published in a commercial industry, this dataset includes not only character objects but also non-character objects, e.g., text bubbles, letters, and background gradation. Therefore, we chose this dataset to evaluate the practical effectiveness of our model.

Tag2pix [12] consists of over 60K filtered large-scale anime illustrations from the Danbooru dataset [5]. While this dataset consists of images of a single character and a simply colored background, the diversity of each character in terms of pose and scale makes it challenging to generate plausible colored outputs. We chose this dataset to verify that our model reflects various user hints well.

CelebA [16] is a representative dataset which contains 203K human face images from diverse races. We chose it to evaluate our model on real-world images rather than artificial ones. We randomly divided each dataset into a training, a validation, and a test set with the ratio of 81:9:10 and resize all images to 256×256 . Referring to the recipe of Lee *et al.* (2020) [14], the sketch images were extracted using the XDoG [29] algorithm.

4.2. Evaluation Metrics

Peak signal to noise ratio (PSNR) has been broadly used as a pixel-level evaluation metric for measuring the distortion degree of the generated image in the colorization tasks [35, 9]. The metric is computed as the logarithmic quantity of the maximum possible pixel value of the image divided by the root mean squared error between a generated image and its groundtruth.

Frechét inception distance (FID). We used FID [7] as an evaluation metric for measuring the model performance by calculating the *Wasserstein-2* distance of feature space representations between the generated outputs and the real images. A low FID score means that the generated image is close to the real image distribution.

Number of required interactions (NRI). We propose a new evaluation metric to measure how many user interactions are required for the model to produce an image of a

Methods	Cond.	PSNR \uparrow			FID \downarrow			NRI \downarrow		
		Yumi	Tag2pix	CelebA	Yumi	Tag2pix	CelebA	Yumi	Tag2pix	CelebA
CIC	\times	15.17	13.99	17.01	137.35	167.88	79.05	-	-	-
Pix2Pix	\times	15.11	14.68	16.22	71.93	111.45	54.86	-	-	-
AlacGAN	\times	15.02	14.12	15.73	30.72	46.24	23.52	-	-	-
RTUG	\times	19.05	14.44	17.23	35.52	92.69	52.67	-	-	-
Ours	\times	18.63	15.19	16.53	34.07	55.31	42.46	-	-	-
AlacGAN	\checkmark	15.68	14.53	16.52	29.74	46.52	22.83	31.00	31.00	31.00
RTUG	\checkmark	20.10	16.36	19.16	30.26	63.58	44.45	13.82	14.79	11.64
Ours	\checkmark	20.88	17.55	20.24	24.46	43.18	16.43	11.08	11.39	6.98

Table 1: **Quantitative comparisons** in terms of PSNR, FID, and NRI (Section 4.2). For conditional cases, we compute the expected values of PSNR and FID when the number of synthesized hints follows \mathcal{G} .

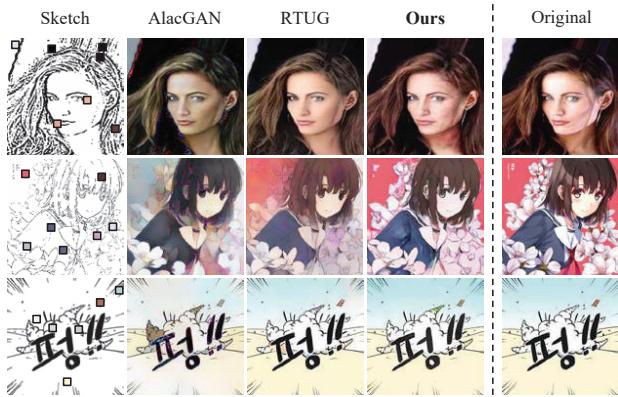


Figure 3: **Comparison to baselines on diverse datasets.** We compare our model with two conditional colorization baselines, AlacGAN and RTUG. From top to bottom, conditional results on CelebA, Tag2pix, and Yumi’s Cells datasets are presented.

certain quality. To this end, we count the number of hints needed by the model to reach a benchmark of PSNR. If the model cannot reach a certain level of accuracy even with the maximum number of hints, we compute the count as the maximum number of hints plus one. The benchmark can be set according to the user’s tolerance or the purpose of a framework. We set 20.5, 17.5, and 19.5 as the benchmarks for Yumi, Tag2pix, and CelebA datasets, respectively.

4.3. Comparisons against Colorization Baselines

We compare our GuidingPainter against diverse baseline models for the deep colorization tasks, including an image translation model Pix2Pix [9], an automatic colorization model CIC [35], a point-based interactive colorization model RTUG [36], and a scribble-based interactive colorization model AlacGAN [3]. Since our main focus is interactive colorization, we primarily analyze the performance of GuidingPainter using conditional cases. In order to analyze the colorization efficiency of conditional mod-

els, we compute NRI and the expected values of PSNR and FID when the number of color hints follows the distribution \mathcal{G} . The color hints are synthesized by their own method used for training, i.e., RTUG and AlacGAN provide hints in random location while our model provides color hints to the regions obtained by the active-guidance module, in order from the front channel (S_1, S_2, \dots). Table 1 presents the quantitative results of our model and other baselines on each dataset. Our model outperforms all of conditional baselines on all three metrics. This reveals that our model can generate various realistic images, reflecting the given conditions while reducing the interactions. Although our framework is mainly designed to colorize sketch images when conditions are given, our model shows the comparable performances across the automatic colorization setting. We also analyze the effectiveness of our guidance mechanism in situations where real users give hints in Section 4.4.

As shown in Fig. 3, our model colorizes each color within each segment by successfully reflecting both the location and the color of hints. The results show that ours is better than other conditional baselines. For a fair qualitative comparison, we equalize the number of hints given to each method and make the locations of the color hints for AlacGAN and RTUG similar to ours, by sampling the points in the regions that our segmentation network produces. The marks in the sketch image in Fig. 3 indicate where the hints are provided for RTUG. Compared with the conditional baselines on the animation dataset, our model reduces the color bleeding artifact, e.g., the second row in Fig. 3, and generates the continuous colors for each segment, e.g., hair in the first row, the sky and the ground in the third row in Fig. 3. This reveals that our model can distinguish the semantic regions of character and background and reflect the color hints into the corresponding regions. Especially, for the last two rows of Fig. 3, our model is superior to colorize the background region, while other baselines colorize the background across the edges or only part of the object. Technically, our approach can be applied to colorize not only a sketch image but also a gray-scale image.

	Yumi's Cells	TPI _↓ / QS _↑ Tag2pix	CelebA	CS _↑ total
RTUG	11.87 / 3.93	8.02 / 3.15	7.82 / 3.85	3.14
Ours	7.80 / 4.07	7.22 / 4.00	7.13 / 3.81	4.07

Table 2: **User study results on three different datasets.**

Time per interaction (TPI) is the average time (sec.) spent by a user before moving on to next interaction. Quality score (QS) is the overall quality of a colorized image. Convenience score (CS) denotes users’ convenience on the overall workflow. QS and CS are measured from one to five, and all scores were surveyed by users.

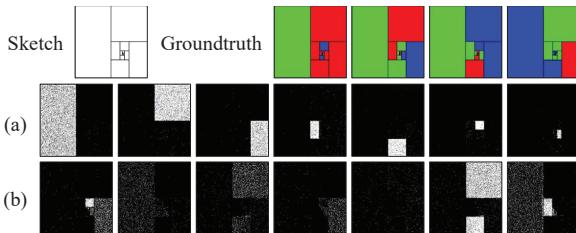


Figure 4: **Dark snail example.** A sketch image (top left) and a groundtruth image (top right) on dark snail. (a) are prioritization results of GuidingPainter, and (b) are when we fix $N_h = N_c$ of our model during the training time.

Additional results for qualitative comparison and grayscale colorization are in the supplementary materials.

4.4. User Study on Interactive Colorization Process

To validate the practical interactive process of our active-guidance mechanism, we develop a straightforward user interface (UI) that control peripheral variables except for our main algorithm. We conduct an in-depth user evaluation, in which users directly participate in the process of our framework. We then record various metrics to assess the practical usefulness of our method. We choose RTUG as our baseline interactive method since its interactive process is directly comparable to ours. As shown in Table 2, our model shows better time-per-interaction (TPI) scores with less qualitative degradation than RTUG model, confirming the superior *time efficiency* of our model. The total colorization time is decreased by 14.2% on average compared to RTUG. Furthermore, the improvement in the convenience score (CS) reveals that our approach clearly reduces the users’ workload. For more details, e.g., our UI design, see the supplementary material.

4.5. Effectiveness of Active-Guidance Mechanism

To understand the effects of our active-guidance mechanism described in Section 3.2-3.3, we design two sub-experiments as follows.

Dark Snail. The first one is a simulation to show that the

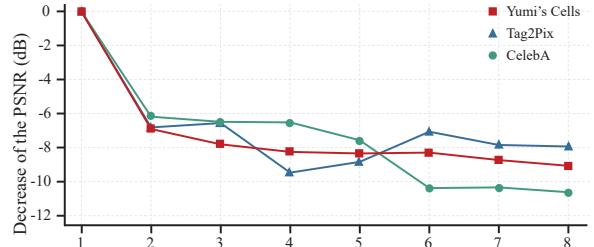


Figure 5: **PSNR scores of GuidingPainter decrease** when masking out all segment information except for the i -th hint.

proposed mechanism works as we expected by using the toy example named *Dark Snail*. As shown in the first row of Fig. 4, squares and rectangles are sequentially placed in a clockwise direction, and a groundtruth is generated at every mini-batch by having randomly sampled colors of red, green, and blue. In this setting, it is impossible for a model to estimate the exact color of each object unless each color hint is provided. Because the size of each rectangle is halved compared to the previous one, querying the largest region first is an optimal choice in terms of the information gain. In other words, this toy experiment is designed to confirm whether our model can (i) divide the semantic regions with the same color and (ii) ask for the color hints of objects in a descending order by their size. Fig. 4 (a) shows the guided regions obtained from a model that is trained by our original mechanism, GuidingPainter. Surprisingly, the original model tends to build the semantic segments, which are i) bounded by only one object and ii) placed in decreasing order based on the segment’s size, except for the 4-th case. Alternately, Fig. 4 (b) is retrieved from a modified version of our model that is trained by fixing $N_h = N_c$ during the training time, i.e., we simply turn off the most critical role of hint generation function. Fig. 4 (b) demonstrates that the modified model totally loses its guiding function, implying that the active-guidance mechanism plays a critical role in our framework.

Importance of highly ranked segments. For every dataset described in Section 4.1, we test how each segment provided by the active-guidance module affects the performance of colorization. To assess the importance of the i -th segment, we put the map of the i -th channel in front of remaining channels of S and then give a hint only at the first segment. Fig. 5 shows the tendency that the PSNR score decreases as a hint is given from the rear-ranked segment, which shows that the active-guidance module encourages to locate the important regions in the front channels of S .

While following the colorization order suggested by the model is an efficient way to reduce loss at training time, it is also possible to change the colorization order with additional learning. Detailed discussions on our approach, including the learning method for changing the order and limitations, are provided in the supplementary materials.

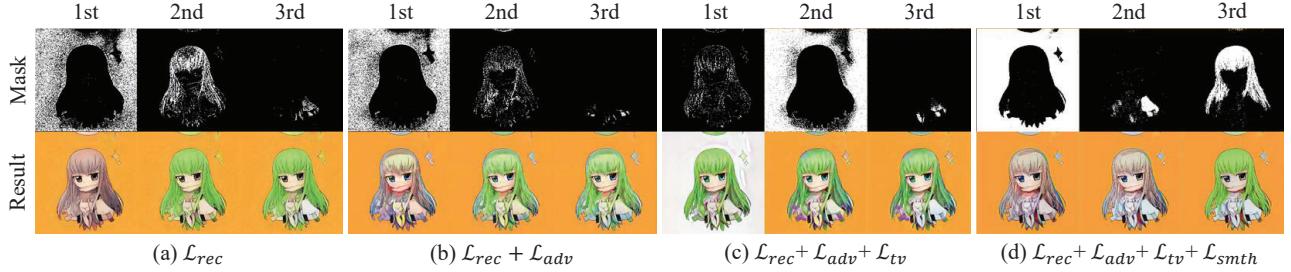


Figure 6: **Qualitative results of ablation study** for the losses (a) \mathcal{L}_{rec} , (b) $\mathcal{L}_{rec} + \mathcal{L}_{adv}$, (c) $\mathcal{L}_{rec} + \mathcal{L}_{adv} + \mathcal{L}_{tv}$, and (d) $\mathcal{L}_{rec} + \mathcal{L}_{adv} + \mathcal{L}_{tv} + \mathcal{L}_{smth}$. Each column of images indicates the priority of guided regions. The first row shows regions guided by the model in order of their priority, and the second row represents the intermediate colorization results.

	\mathcal{L}_{rec}	\mathcal{L}_{adv}	\mathcal{L}_{tv}	\mathcal{L}_{smth}	PSNR / FID / NRI		
					Yumi's Cells	Tag2pix	CelebA
(a)	✓				21.04 / 28.30 / 9.95	18.11 / 66.21 / 8.86	20.76 / 39.73 / 5.58
(b)	✓	✓			20.84 / 27.90 / 11.05	17.55 / 48.37 / 11.30	20.41 / 17.15 / 6.13
(c)	✓	✓	✓		20.72 / <u>27.81</u> / <u>10.39</u>	<u>17.58</u> / 47.32 / <u>10.48</u>	20.19 / <u>16.54</u> / 7.20
(d)	✓	✓	✓	✓	<u>20.88</u> / 24.46 / 11.08	17.55 / 43.18 / 11.39	20.24 / 16.43 / 6.98

Table 3: **Quantitative results of the ablation study** for the losses (a) \mathcal{L}_{rec} , (b) $\mathcal{L}_{rec} + \mathcal{L}_{adv}$, (c) $\mathcal{L}_{rec} + \mathcal{L}_{adv} + \mathcal{L}_{tv}$, and (d) $\mathcal{L}_{rec} + \mathcal{L}_{adv} + \mathcal{L}_{tv} + \mathcal{L}_{smth}$.

4.6. Effectiveness of Loss Functions

This section analyzes the effects of each loss function using both quantitative measurements and qualitative results. In this ablation study, we found a trade-off between the pixel-distance-based metric, i.e., PSNR, and the feature-distribution-based metric, i.e., FID, according to the combination of loss functions. Since \mathcal{L}_{rec} exactly matches up to the PSNR, Table 3 (a) shows the best score of the PSNR-related measurement. However, it does not perform well in terms of FID especially in the Tag2pix and CelebA datasets. This phenomenon can also be found in Fig. 6 (a). The character in the first colorization result tends to be painted with grayish color, and overall colorization results loss sharpness. After \mathcal{L}_{adv} is added, the FID scores in Table 3 (b) dramatically improve, along with the qualitative results in Fig. 6 (b), but PSNR-based scores slightly decrease. As discussed in a previous work [28], we guess that the PSNR score is not sufficient to measure how naturally a model can generate if only partial conditions are given. Although Fig. 6 (b) shows plausible images, the hair in all the output images are slightly stained. By adding \mathcal{L}_{tv} , these stains are removed, and the colors become clear, as shown in Fig. 6 (c). After adding \mathcal{L}_{smth} , the guided regions become significantly less sparse than before, and the strange colors on the sleeve of Fig. 6 (c)'s character disappear, as shown in Fig. 6 (d). Table 3 shows the FID score improves after adding \mathcal{L}_{adv} , \mathcal{L}_{tv} , and \mathcal{L}_{smth} one by one from \mathcal{L}_{rec} on all datasets. Despite the trade-off, we select (d) as our total loss function, considering the qualitative improvements and the balance

between the PSNR-based and FID metrics.

5. Conclusions

This work presents a novel interactive deep colorization framework, which enables the model to learn the priority regions of a sketch image that are most in need of color hints. Experimental results show that our framework improves the image quality of interactive colorization models, successfully reflecting the color hints with our active guidance mechanism. Importantly, our work demonstrates that GuidingPainter, without any manual supervision at all, can learn the ability to divide the semantic regions and rank them in decreasing order of priority by utilizing the colorization signal in an end-to-end manner. We expect that our approach can be used to synthesize hints for training other interactive colorization models. Developing a sophisticated UI which integrates our region prioritization algorithm with diverse techniques, such as region refinement, remains as our future work.

Acknowledgments This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) (No. 2019-0-00075, Artificial Intelligence Graduate School Program (KAIST) and No. 2021-0-01778, Development of human image synthesis and discrimination technology below the perceptual threshold) and the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No. NRF-2022R1A2B5B02001913).

References

- [1] Yun Cao, Zhiming Zhou, Weinan Zhang, and Yong Yu. Unsupervised diverse colorization via generative adversarial networks. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 151–166. Springer, 2017.
- [2] Yu Cheng, Zhe Gan, Yitong Li, Jingjing Liu, and Jianfeng Gao. Sequential attention gan for interactive image editing, 2020.
- [3] Yuanzheng Ci, Xinzhu Ma, Zhihui Wang, Haojie Li, and Zhongxuan Luo. User-guided deep anime line art colorization with conditional adversarial networks. In *Proceedings of the 26th ACM international conference on Multimedia*, pages 1536–1544, 2018.
- [4] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.
- [5] Aaron Gokaslan Gwern Branwen. Danbooru2017: A large-scale crowdsourced and tagged anime illustration dataset. <https://www.gwern.net/Danbooru2017>, 2018. [Online; accessed 22-03-2018].
- [6] Eric Heim. Constrained generative adversarial networks for interactive image generation, 2019.
- [7] 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. In *NIPS*, 2017.
- [8] Ronghang Hu, Marcus Rohrbach, and Trevor Darrell. Segmentation from natural language expressions, 2016.
- [9] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017.
- [10] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *ICLR*, 2017.
- [11] Youngjoo Jo and Jongyoul Park. Sc-fegan: Face editing generative adversarial network with user’s sketch and color, 2019.
- [12] Hyunsu Kim, Ho Young Jhoo, Eunhyeok Park, and Sungjoo Yoo. Tag2pix: Line art colorization using text tag with secat and changing loss. In *ICCV*, 2019.
- [13] Cheng-Han Lee, Ziwei Liu, Lingyun Wu, and Ping Luo. Maskgan: Towards diverse and interactive facial image manipulation, 2020.
- [14] Junsoo Lee, Eungyeup Kim, Yunsung Lee, Dongjun Kim, Jaehyuk Chang, and Jaegul Choo. Reference-based sketch image colorization using augmented-self reference and dense semantic correspondence, 2020.
- [15] Yao Li, Xianggang Yu, Xiaoguang Han, Nianjuan Jiang, Kui Jia, and Jiangbo Lu. A deep learning based interactive sketching system for fashion images design, 2020.
- [16] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *ICCV*, 2015.
- [17] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *ICLR*, 2017.
- [18] Soumajit Majumder and Angela Yao. Localized interactive instance segmentation, 2020.
- [19] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.
- [20] Seonghyeon Nam, Yunji Kim, and Seon Joo Kim. Text-adaptive generative adversarial networks: Manipulating images with natural language, 2018.
- [21] NaverWebtoon. Yumi’s cells. <https://comic.naver.com/webtoon/list.nhn?titleId=651673>, 2019. [Online; accessed 22-11-2019].
- [22] Tiziano Portenier, Qiyang Hu, Attila Szabó, Siavash Arjomand Bigdeli, Paolo Favaro, and Matthias Zwicker. Faceshop: Deep sketch-based face image editing, 2018.
- [23] Holger R Roth, Dong Yang, Ziyue Xu, Xiaosong Wang, and Daguang Xu. Going to extremes: Weakly supervised medical image segmentation, 2020.
- [24] Patsorn Sangkloy, Jingwan Lu, Chen Fang, Fisher Yu, and James Hays. Scribbler: Controlling deep image synthesis with sketch and color. In *CVPR*, 2017.
- [25] Jheng-Wei Su, Hung-Kuo Chu, and Jia-Bin Huang. Instance-aware image colorization. In *CVPR*, 2020.
- [26] Zhentao Tan, Menglei Chai, Dongdong Chen, Jing Liao, Qi Chu, Lu Yuan, Sergey Tulyakov, and Nenghai Yu. Michigan: Multi-input-conditioned hair image generation for portrait editing, 2020.
- [27] Hao Tang, Song Bai, Philip H. S. Torr, and Nicu Sebe. Bipartite graph reasoning gans for person image generation, 2020.
- [28] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment. In *The Thirly-Seventh Asilomar Conference on Signals, Systems & Computers*, 2003, volume 2, pages 1398–1402. Ieee, 2003.
- [29] Holger Winnemöller, Jan Eric Kyprianidis, and Sven C Olsen. Xdog: an extended difference-of-gaussians compendium including advanced image stylization. *Computers & Graphics*, 36(6):740–753, 2012.
- [30] Yi Xiao, Peiyao Zhou, and Yan Zheng. Interactive deep colorization with simultaneous global and local inputs, 2018.
- [31] Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, and Xiaodong He. Attngan: Fine-grained text to image generation with attentional generative adversarial networks, 2017.
- [32] Seungjoo Yoo, Hyojin Bahng, Sunghyo Chung, Junsoo Lee, Jaehyuk Chang, and Jaegul Choo. Coloring with limited data: Few-shot colorization via memory augmented networks. In *CVPR*, 2019.
- [33] Lvmin Zhang, Chengze Li, Edgar Simo-Serra, Yi Ji, Tien-Tsin Wong, and Chunping Liu. User-guided line art flat filling with split filling mechanism. In *CVPR*, 2021.
- [34] Lvmin Zhang, Chengze Li, Tien-Tsin Wong, Yi Ji, and Chunping Liu. Two-stage sketch colorization. *ACM Transactions on Graphics (TOG)*, 37(6):1–14, 2018.
- [35] Richard Zhang, Phillip Isola, and Alexei A. Efros. Colorful image colorization. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *ECCV*, 2016.
- [36] Richard Zhang, Jun-Yan Zhu, Phillip Isola, Xinyang Geng, Angela S Lin, Tianhe Yu, and Alexei A Efros. Real-time user-guided image colorization with learned deep priors. *ACM Transactions on Graphics (TOG)*, 36(4):1–11, 2017.

[37] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A. Efros. Generative visual manipulation on the natural image manifold, 2018.