Style-Structure Disentangled Features and Normalizing Flows for Diverse Icon Colorization

Yuan-kui Li Yun-Hsuan Lien Yu-Shuen Wang

National Yang Ming Chiao Tung University

Abstract

We present a colorization network that generates flat-color icons according to given sketches and semantic colorization styles. Our network contains a style-structure disentangled colorization module and a normalizing flow. The colorization module transforms a paired sketch image and style image into a flat-color icon. To enhance network generalization and the quality of icons, we present a pixel-wise decoder, a global style code, and a contour loss to reduce color gradients at flat regions and increase color discontinuity at boundaries. The normalizing flow maps Gaussian vectors to diverse style codes conditioned on the given semantic colorization label. This conditional sampling enables users to control attributes and obtain diverse colorization results. Compared to previous methods built upon conditional generative adversarial networks, our approach enjoys the advantages of both high image quality and diversity. To evaluate its effectiveness, we compared the flat-color icons generated by our approach and recent colorization and image-to-image translation methods on various conditions. Experiment results verify that our method outperforms state-of-the-arts qualitatively and quantitatively.

1. Introduction

Image colorization aims to generate color images based on grayscale references, such as monochrome photos and line arts. Most methods developed for colorizing monochrome photos are fully automatic because pixels with various intensities contain fruitful semantics. The networks can recognize objects and assign proper colors when colorizing images. Line arts, however, contain little semantics due to sparse structure lines. Moreover, colors in a line art may not have ground truths, and in some cases, they are not necessarily meaningful. Accordingly, previous line arts colorization methods require users to provide reference images or color hits to guide the generated results.

Icons and comics are two types of graphic designs that are widely used in communication. A step of creating them is colorization. To save designers' workload, methods take sketch images as inputs, which contain only black and white pixels for representing objects and backgrounds, and determine the color of each pixel. The methods strive to enhance color harmony and vividness and prevent colors from spreading the boundaries of adjacent objects. Although the colorization of icons and comics share several similarities, they are two different designs – structure lines are present in comics, but they are absent in icons. The structure of an icon appears because of color discontinuity. Hence, colorizing icons is challenging because methods have to consider where and what colors to assign and whether the change of colors exhibits clear and correct structure lines.

Training a conditional generative adversarial network (c-GAN) is a way to generate flat-color icons. Sun et al. [35] trained a c-GAN with two discriminators, which evaluate the structure and style of icons created by the generator. Although their generated results are visually appealing, the images frequently contain gradient colors and fail to present small features. In addition, it is known that c-GANs suffer from the diversity problem because the generator often degenerates into a deterministic function.

Inspired by StyleFlow [1], where specifying attributes to control a generator would degrade image qualities, we discard the framework of c-GAN. Instead, we train an encoder-decoder network to map paired sketch images and style images to flat-color icons using supervised learning. A pixel-wise decoder, a global style code, and a contour loss were introduced to help the network disentangle style and structure features, reduce color gradients at flat regions, and increase color discontinuities at boundaries. We also train a continuous normalizing flow [10] to sample diverse style codes conditioned on the given semantic style label [20]. Figure 2 shows the styles that users can choose when using our system. We concatenate the sampled style codes to the structure embedding and generate colorization results.

We apply our network to generate flat-color icons con-

ditioned on a variety of black-and-white sketch images and semantic colorization styles. Figures 5, 6, and 7, and our accompanying video show the results. To evaluate its effectiveness, we compare the flat-color icons generated by our approach and recent colorization and image-to-image translation methods. Experiment results demonstrate that our network outperforms current state-of-the-arts both qualitatively and quantitatively.

2. Related Work

Normalizing flows [29] is a class of generative models focusing on mapping a complex probability distribution to a simple distribution such as a Gaussian. The advantages of flow-based models are easy to sample, stable training, and accurate probability density estimation. Several studies in this field have proved that mapping complex image distributions to a Gaussian is practical [6, 7, 12, 18]. Compared to flow-based models, GANs [9] suffer from the problem of mode collapse, and the convergence criteria are unclear; autoregressive models [36] are slow during sampling; and variational autoencoders [19] make a strong assumption that the priors of data distributions are a Gaussian. Despite of the advantages, flow-based models are less expressive due to the requirement of invertibility and tractability. The layer designs [6, 7] in discrete flow-based models for fast computation of inverse Jacobian matrix worse the problem further. Recently, normalizing flows based on continuous time transforms [4, 10] were presented to ease the layer restriction and enjoy constant memory usage during training, although the expressiveness problem still remains.

Flow-based models are capable of controlling attributes by concatenating parameters to embeddings. Lugmayr et al. proposed SRFlow [27] to generates diverse high-resolution images conditioned on low-resolution ones. Abdal et al. [1] sampled latent vectors based on given attributes and fed the vectors to the StyleGAN [15] generator to synthesize high-quality images. Since flow-based models are less expressive than feedforward networks, in this study, we map random variables to style codes rather than images for colorization.

Image-to-Image Translation aims to map images from one domain to another. Most of the previous methods in this field are based on conditional generative adversarial networks (c-GANs) [28]. Isola et al. [14] trained the network on paired images in two domains to achieve image-to-image translation. While the paired images in certain applications could be difficult to obtain, follow-up methods adopt the share-latent space assumption [26] or the cycle consistency loss [17, 43] to lifted the restriction. There are also methods presented to disentangle structure and texture features when generating images [30, 37]. Since generators in c-GANs often degenerate into deterministic functions, methods such as BicycleGAN [42], MUNIT [13], DRIT [22], and DRIT++ [23] were introduced to overcome the diversity problem. In addition to full automation, several methods allowed users to additionally provide labels [21, 38], style images [25], or sentences [3] to control domain translation.

Line Art Colorization transforms images containing only structure lines to full-color images. While structure lines are not as expressive as gray-scale images, and characters in manga are highly stylized, methods in this field often require users to provide color hints [5, 32, 34, 41], labels [16], or reference images [2, 8, 24, 35, 40] for the network to consider. Due to the lack of texture information, the works of [16, 32] applied two-step training methods to improve color vividness and properly colorize small features. Recently, Zhang et al. [41] presented a framework to colorize line arts with flat colors by computing the influence area of each scribble to avoid color leakage/contamination problems. Lee et al. [24] distorted the style image paired with the contour image and prevented the network from utilizing structure information in style images.

Colorizing an icon and a line art are different because the structure of an icon is represented by color discontinuity rather than solid lines. Sun et al. [35] presented the first icon colorization system. They trained a c-GAN with dual discriminators, which evaluate whether the generated flat-color icons fulfill structure and style constraints. Han et al. [11] followed the similar idea and used the given masks to prevent the network from mis-colorizing background regions. Since the above-mentioned methods are built upon c-GAN, they inherit its shortcomings in terms of quality and diversity. In this work, we train an encoder-decoder network to colorize icons. The style codes required by the colorization are sampled using conditional normalizing flows. This colorization strategy achieves both high-quality and diversity.

3. Background

Normalizing flows allow a bi-directional transformation of samples between two distributions. Let $p^*_{\mathbf{s}|\mathbf{c}}(\mathbf{s}|\mathbf{c})$ be an unknown conditional data distribution, and $p_{\mathbf{u}}(\mathbf{u})$ be a distribution that is easy for sampling and density estimation. A conditional normalizing flow aims to express the relationship of $\mathbf{u}$ and $\mathbf{s}|\mathbf{c}$ as:

$$\mathbf{u} = f_\theta(\mathbf{s};\mathbf{c}), \quad \mathbf{s} = f_\theta^{-1}(\mathbf{u};\mathbf{c}), \tag{1}$$

where $\mathbf{u} \sim p_{\mathbf{u}}(\mathbf{u})$, $\mathbf{s} \sim p^*_{\mathbf{s}|\mathbf{c}}(\mathbf{s}|\mathbf{c})$, f_θ is an invertible neural network parameterized by θ. In practice, $p_{\mathbf{u}}(\mathbf{u})$ can be a standard normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$.

Conditional Discrete Normalizing Flows (c-DNFs). The idea of DNFs is that the probability density of $\mathbf{s}$ can be explicitly computed using the change-of-variable rule:

$$p_{\mathbf{s}}(\mathbf{s};\mathbf{c},\theta) = p_{\mathbf{u}}\left(f_\theta(\mathbf{s};\mathbf{c})\right)\left|\det J_\theta(\mathbf{s};\mathbf{c})\right|, \tag{2}$$

where $p_{\mathbf{s}|\mathbf{c}}(\mathbf{s};\mathbf{c},\theta)$ is the approximation of a real distribution $p^*_{\mathbf{s}|\mathbf{c}}(\mathbf{s}|\mathbf{c})$, $J_\theta(\mathbf{s};\mathbf{c}) = \frac{\partial f_\theta}{\partial \mathbf{s}}(\mathbf{s};\mathbf{c})$, and $|\det J_\theta(\mathbf{s};\mathbf{c})|$ is

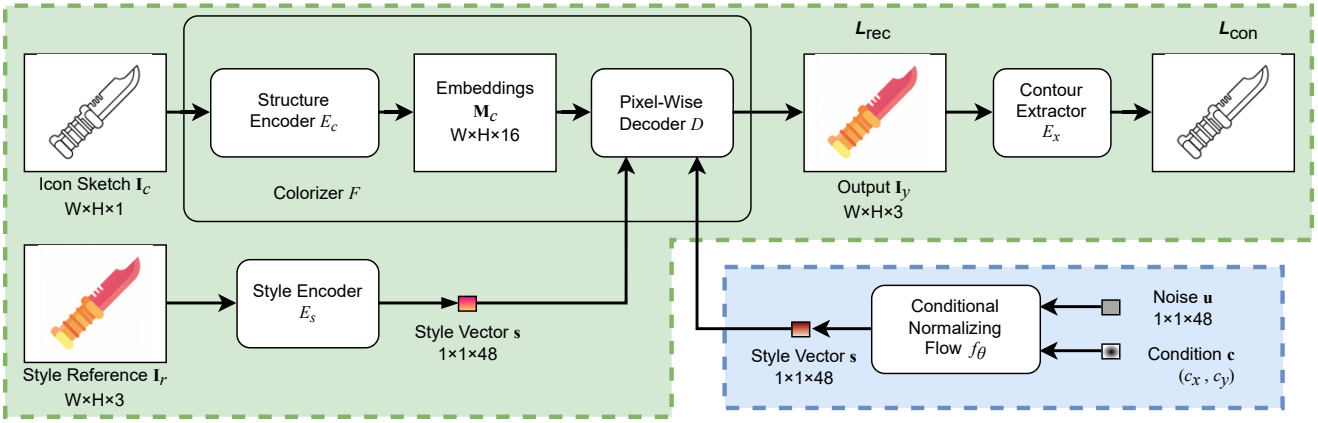

Figure 1. Our system contains a colorization network (green) and a continuous normalizing flow (blue). The former maps the paired sketch image and style image to a flat-color icon. The latter generates diverse style vectors conditioned on the given style for colorization.

the volume change caused by the transformation. Since f is trained to transform $p^*_{\mathbf{s}|\mathbf{c}}(\mathbf{s}|\mathbf{c})$ to $p_{\mathbf{u}}(\mathbf{u})$, θ can be computed by minimizing the negative log likelihood (NLL):

$$\begin{aligned}\mathcal{L}(\theta; \mathbf{s}, \mathbf{c}) &= -\log p_{\mathbf{s}|\mathbf{c}}(\mathbf{s}; \mathbf{c}, \theta) \\ &= -\log p_{\mathbf{u}}(f_\theta(\mathbf{s}; \mathbf{c})) - \log \left|\det J_\theta(\mathbf{s}; \mathbf{c})\right|.\end{aligned} \tag{3}$$

To achieve a tractable expression, methods in DNFs decompose a neural network f into invertible layers $f_i : R^d \rightarrow R^d$, where i is the layer index. Let f contain k layers, the transformations between $\mathbf{s}$ and $\mathbf{u}$ can be formulated as $\mathbf{u} = f_{k-1}(...f_1(f_0(\mathbf{s}; \mathbf{c})))$ and $\mathbf{s} = f_0^{-1}(f_1^{-1}(...f_{k-1}^{-1}(\mathbf{u}; \mathbf{c})))$. Also let θ_i be the parameter of layer i, and $\mathbf{h}_{i+1} = f_i(\mathbf{h}_i)$, where $\mathbf{h}_0 = \mathbf{s}$ and $\mathbf{h}_k = \mathbf{u}$. By applying the chain rule, the NLL objective in Equation 3 can be derived as

$$\mathcal{L}(\theta; \mathbf{s}, \mathbf{c}) = -\log p_{\mathbf{u}}(f_\theta(\mathbf{s}; \mathbf{c})) - \sum_{i=0}^{k-1} \log \left|\det J_{\theta_i}(\mathbf{s}; \mathbf{c})\right|,$$

$$\text{where} \quad J_{\theta_i}(\mathbf{s}; \mathbf{c}) = \frac{\partial f_{\theta_i}}{\partial \mathbf{h}_i}(\mathbf{h}_i; \mathbf{c}) \tag{4}$$

Conditional Continuous Normalizing Flows (c-CNFs). While c-DNFs transform data between $\mathbf{u}$ and $\mathbf{s}$ through discrete layers, c-CNFs parameterize the dynamics of data transformation over time, which can be expressed as an ordinary differential equation (ODE) [4]:

$$\frac{d\mathbf{u}_t}{dt} = g_\theta(t, \mathbf{c}, \mathbf{u}_t). \tag{5}$$

where $\mathbf{u}_t$ is the state at time t, $\mathbf{u}_{t_0} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and $\mathbf{c}$ is a given condition. The uniform Lipschitz continuity of g_θ in $\mathbf{u}_t$ ensures the invertibility [4, 10]. In other words, the state $\mathbf{u}_{t_0}$ $(= \mathbf{u})$ evolves to $\mathbf{u}_{t_1}$ $(= \mathbf{s})$ over time with the dynamics parameterized by g_θ. We thus compute $\mathbf{s}$ from $\mathbf{u}$ by integrating g_θ across time:

$$\mathbf{s} = \mathbf{u}_{t_0} + \int_{t_0}^{t_1} g_\theta(t, \mathbf{c}, \mathbf{u}_t)\, dt. \tag{6}$$

Similarly, to approximate the complex data distribution, c-CNFs are trained to minimize the NLL loss

$$\begin{aligned}\mathcal{L}(\theta; \mathbf{s}, \mathbf{c}) &= -\log p_{\mathbf{s}|\mathbf{c}}(\mathbf{s}; \mathbf{c}, \theta) \\ &= -\log p_{\mathbf{u}}(\mathbf{u}_{t_0}) + \int_{t_0}^{t_1} \mathbf{tr}\left(\frac{\partial g_\theta}{\partial \mathbf{u}_t}\right) dt.\end{aligned} \tag{7}$$

The forward and backward propagations of CNFs over time can be solved by an ODE solver. In our implementation, we use the adjoint method [31] to compute gradients.

Note that both CNF and DNF can map a normal distribution to approximate a data distribution. They are different in theory and network structures but can be controlled by both discrete and continuous attributes. We choose CNF because it can better approximate the distribution of style vectors.

4. Conditional Icon Colorization Network

4.1. System Overview

Let $\mathbf{I}_c$ be a sketch image and $\mathbf{s}$ be a style vector. The goal of our colorization network F is to generate a flat-color icon

$$\mathbf{I}_y = F(\mathbf{I}_c, \mathbf{s}). \tag{8}$$

The style vector $\mathbf{s}$ can be encoded from a style image $\mathbf{I}_r$ or sampled from a CNF conditioned on the given colorization style $\mathbf{c}$. In this study, $\mathbf{c} = (x, y)$ is a coordinate defined in the color image scale [20], as illustrated in Figure 2. It deserves noting that $\mathbf{s}$ and $\mathbf{c}$ in Equations 6, 7, and 8 are equivalent since we use a CNF to sample style vectors.

Figure 1 shows the framework of our system. It contains a colorization network that maps a sketch image $\mathbf{I}_c$ and a style image $\mathbf{I}_r$ to a flat-color icon $\mathbf{I}_y$. The network is trained by minimizing the reconstruction error and the contour loss. The other part of the framework is a c-CNF for sampling diverse style vectors conditioned on the given semantic style for colorization. We describe the details as follows.

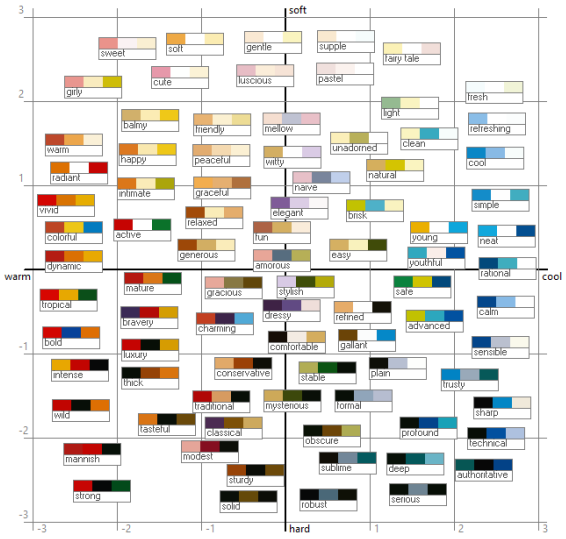

Figure 2. The color image scale [20] contains 85 semantic colorization styles. Each style is composed of three dominant colors. The x- and y- axes of this color image scale represent color temperature and hardness, respectively.

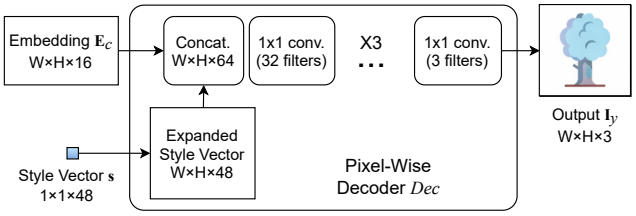

Figure 3. Pixel-wise decoder decodes each pixel embedding independently. Each 1×1 convolution layer, except for the last one, is followed by a pixel normalization and a ReLU.

4.2. Style-Structure Disentangled Colorization

Our colorization network contains a structure encoder E_c, a style encoder E_s, and a pixel-wise decoder D. The structure encoder E_c is an U-Net [33] that transforms an icon sketch $\mathbf{I}_c$ to an embedding $\mathbf{M}_c$. The style encoder E_s extracts a style vector $\mathbf{s}$ from the referenced style image $\mathbf{I}_r$. The two features are concatenated and then decoded to flat-color icons. We train the colorization network using the reconstruction loss. Specifically,

$$\mathcal{L}_{rec}(\mathbf{I}_c, \mathbf{I}_r) = \|F(\mathbf{I}_c, \mathbf{I}_r) - \mathbf{I}_r\|_F^2 , \tag{9}$$

where $\mathbf{I}_c$ and $\mathbf{I}_r$ are paired images.

We introduce two training strategies to improve the generalization of our colorization network. The main idea is to disentangle the structure encoder E_c and the style encoder E_s. To keep E_c focused only on structures, we augment the style image $\mathbf{I}_r$ by color shift. That is, we convert an image to the HSV color space and then randomly rotate the hue to

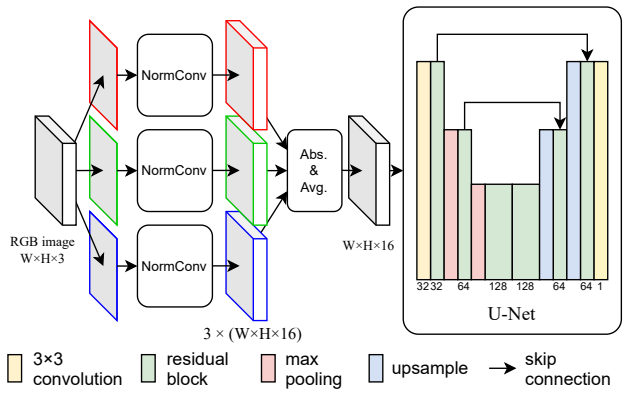

Figure 4. At the front of the contour extractor network is our presented *NormConv* layer, in which the sum of the parameters is zero. This constraint forces the network to determine the color gradient of every local region. The map containing gradient magnitudes is then fed into a U-Net to estimate the image contour.

generate its variants. Since an icon sketch $\mathbf{I}_c$ corresponds to many style images $\mathbf{I}_r$ in this case, the network can only obtain color information from $\mathbf{I}_r$ when colorizing $\mathbf{I}_c$. To prevent E_s from containing structure information, we constrain the style vector to be $1 \times 1 \times 48$. Afterward, the style vector $\mathbf{s}$ is expanded to the resolution of $W \times H \times 48$ for concatenating with the embedding $\mathbf{M}_c$. In addition, we apply the 1×1 convolution to decode the concatenated feature to a flat-color icon, as illustrated in Figure 3. Since each pixel in the embedding $\mathbf{M}_c$ is processed individually without considering its neighbors, it forces the structure encoder to take over the works related to structures, such as identifying closed or nearly-closed regions in a sketch image.

Remember that the structure of a flat-color icon is formed by color discontinuities. It indicates that neighboring pixels in the same region and in different regions should have identical and distinct colors, respectively. Because the intensity of a structure line is unknown and it depends on the styles used to colorize an icon, using a heuristic loss function, i.e., maximizing the color discontinuity at boundaries, to guide the colorization network is inapplicable. Therefore, we apply a contour extractor network E_x to evaluate whether the generated flat-color icons $\mathbf{I}_y$ fulfill the requirement. Specifically, we append E_x at the back of the colorization network to extract the sketch of $\mathbf{I}_y$, denoted as $\mathbf{I}'_c = Ex(\mathbf{I}_y)$, and expect the extracted $\mathbf{I}'_c$ possibly close to the input sketch $\mathbf{I}_c$. We formulate the loss function as:

$$\mathcal{L}_{con}(\mathbf{I}_c, \mathbf{I}_y) = \|\mathbf{I}'_c - \mathbf{I}_c\|_F^2 . \tag{10}$$

Figure 4 shows the network architecture of our contour extractor E_x. The front part of E_x is our presented NormConv layer, where the sum of the parameters in each kernel is zero. We apply this zero-sum constraint to force the layer to estimate a variety of color gradients at every local region.

We also treat RGB channels separately due to non-linear human visual perception. Afterward, the gradient magnitudes are fused and fed into a U-Net for contour estimation. Note that the contour extractor E_x is trained on real data. We freeze E_x when training the colorization network to prevent E_x from being corrupted by the generated results.

4.3. Super Resolution of Flat-Color Icons

Most computer vision methods demand to generate high-resolution images. So does our icon colorization. An intuitive idea to achieve the aim is to enlarge a network and inputs' and outputs' resolutions. Although with a higher computation cost and memory consumption, in practice, this strategy seldom succeeds due to training instability and convergence to local minimums. In this study, we achieve high-resolution flat-color icons by upsampling the low-resolution ones. It solves the problem neatly because it is unnecessary to synthesize unexisting fine details when upsampling flat-color icons. The only task that a network has to accomplish is retaining sharp boundaries. Therefore, we append an upsampling network SR at the back of output $\mathbf{I}_y$ to increase its resolution by $2\times$ and $4\times$, respectively. The network SR additionally takes a high-resolution sketch image, which can be easily obtained when designers are sketching an icon, to upsample the result. In our implementation, SR is a residual network. It first upsamples a low-resolution image by linear interpolation and then fine-tunes the pixel colors to minimize the reconstruction loss.

4.4. Conditional Style Sampling using Flows

The semantic style labels $\mathbf{c}$ are based on three dominant colors. However, it does not limit designers to use only three colors when colorization. The combination of dominant and non-dominant colors are complex and should be learned from icons that are well-designed. As a result, we model the distribution of style vectors using a c-CNF.

We determine the semantic style label $\mathbf{c}$ of each icon by considering its colors. To achieve this, we follow Sun et al.'s method [35] by transforming each icon image i into a $8 \times 8 \times 8$ color histogram $\mathcal{H}_i$ without considering background white pixels. We also generate a set of histograms $\mathcal{H}_s$ for each semantic style by setting the ratios of the three dominant colors to 1:1:1, 2:1:1, 1:2:1, 1:1:2, 1:2:2, 2:1:2, and 2:2:1. Then, we compute the distance between $\mathcal{H}_i$ and each of $\mathcal{H}_s$, and label the icon style if the shortest distance is smaller than a threshold δ. In our implementation, $\delta = 0.06$, and an icon can be assigned to multiple styles under this measurement. Because the numbers of icons in styles are imbalanced, we also synthesize fake icons to ensure that each style contains at least 1000 samples for training. To implement this idea, we randomly draw primitives, such as squares, circles, and triangles, on a canvas using the dominant colors of a style. The size, position, and the corresponding color of each primitive are randomly determined. We let the primitives overlap with each other to keep the diversity of synthesis.

We train the c-CNF by minimizing the NLL loss defined in Equation 7. The network of our c-CNF is composed of a moving batch normalization layer [39], then four concatsquash layers [10, 39], and again a batch normalization layer. Specifically, the concatsquash layer is defined as

$$CCS(t, \mathbf{c}, \mathbf{u}) = tanh((W_u\mathbf{u} + b_u) \times gate + bias),$$

where $gate = \sigma(W_{tt}t + W_{tc}\mathbf{c} + b_t)$, $bias = (W_{bt}t + W_{bc}\mathbf{c} + b_bt)$, $W_u, W_{tt}, W_{tc}, W_{bt}, W_{bc}, b_u, b_t, b_b$ are learnable parameters, and σ is a sigmoid activation function.

4.5. Implementation Details

We trained the colorization network and the c-CNF sequentially because we consider generating icons that can exhibit clear structures and contain flat-colors is more important than sampling style vectors. Specifically, we trained the colorization network using the reconstruction error and the contour loss for 600K iterations. The Adam optimizer was used to update the network parameters. The batch size, learning rate, and β_1 and β_2 used in the Adam were set to 64, 10^{-4}, 0.9, and 0.999, respectively. In addition, the width and height of $\mathbf{I}_c$, $\mathbf{I}_r$, and $\mathbf{I}_y$ were set to W=H=128. Regarding the c-CNF, we updated the network parameters by minimizing the NLL for 100k iterations. Similarly, the batch size, learning rate, β_1 and β_2 used in the Adam, and the tolerance used in the ODE solver were set to 64, 10^{-3}, 0.9, 0.999, and 10^{-5}, respectively.

5. Results and Discussions

We trained the presented network to colorize icons conditioned on a variety of styles. The style vectors were sampled using the c-CNF from $\mathcal{N}(\mathbf{0}, \tau \cdot \mathbf{I})$, where τ is a scalar known as temperature. We set $\tau = 0.4$ to sample style vectors. Figure 5 shows that our generated flat-color icons contain small features and sharp boundaries. In addition, given a pre-defined color style, our system can generate diverse flat-color icons that fulfill the requirement.

The supplemental video demonstrates that our system colorizes icons whenever the canvas is updated, and open boundaries frequently appear when sketching. Since artists seldom draw lines carefully at the early design stage, filling colors in closed regions extracted from structure lines is insufficient to colorize an icon. In addition, closed regions of an icon may belong to the background and should not be colorized. Results in Figure 6 (columns 1, 2, and 12) show that our system can generally prevent such a problem.

5.1. Comparison to State-of-the-Arts

We compared the results generated by our system and several baselines for evaluation. The baselines included

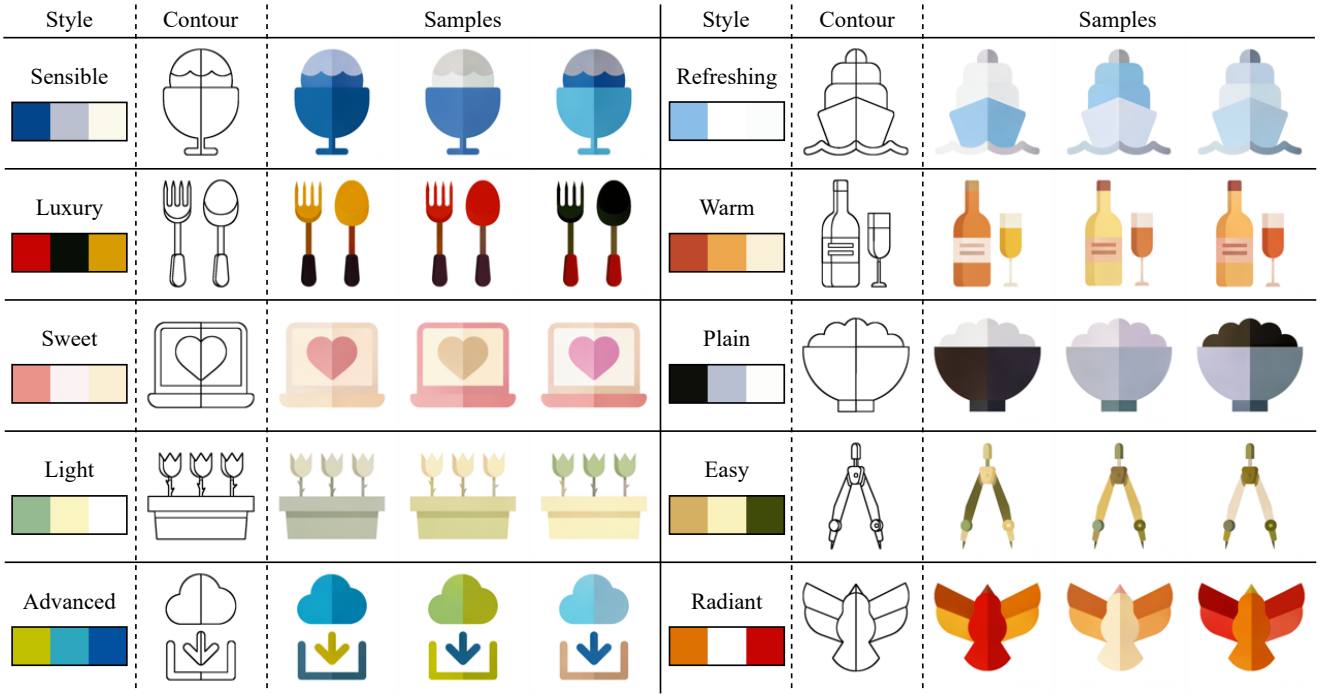

Figure 5. Our system generates flat-color icons based on the given sketch image and the colorization style. The resolution of these icons is 128×128. Note that the results are diverse under the condition of an identical style label.

Anime [40], Comi [8], MUNIT [13], ASCFT [24], and AdvIcon [35]. The implementations of these baselines were obtained from the authors' websites. We trained all of the methods on the dataset released by Sun et al. [35], which contains 12,575 images. In the experiment, 90% of the samples were randomly selected for training, and the remaining 10% were for testing. Since all of the baselines are reference-based colorization methods, we apply our style encoder $\mathbf{E}_s$ to obtain style vectors for coloring icons. We also resize all generated results to the resolution of 128×128 for comparison.

The results in Figure 6 show that all of the baselines produced noticeable gradient colors when colorization. The artifacts frequently appear at small and thin areas. They break the icon structures and in certain cases make the colorized icons unrecognizable. In contrast, the icons generated by our method are flat-color and exhibit clear structure lines.

In addition to visual comparison, we quantitatively evaluated the generated results using the following measures.

Structure distance. Since $\mathbf{I}_y$ is conditioned on the input sketch $\mathbf{I}_c$, we expect that $\mathbf{I}_y$ and $\mathbf{I}_c$ have similar structures. To estimate the structure distance, we first apply the Canny edge detection to extract contour images $\mathbf{I}_{c'}$ from the generated icons $\mathbf{I}_y$. Then, for each edge pixel in $\mathbf{I}_{c'}$, we search for the closest edge pixel in $\mathbf{I}_c$ and accumulate the deviation of these two pixels to obtain the structure distance $D_{c' \to c}$. Considering that the generated icons $\mathbf{I}_y$ may miss certain features, we also compute the distance $D_{c \to c'}$. The final structure distance of $\mathbf{I}_y$ and $\mathbf{I}_c$ is defined as $D_{c' \to c} + D_{c \to c'}$.

Color distance. Similar to the structure distance, we expect that the dominant colors of $\mathbf{I}_y$ and $\mathbf{I}_s$ are alike. Specifically, we first converted each generated icon to the Lab color space and then computed the corresponding $8 \times 8 \times 8$ color histogram. The value of each bin indicates the frequency of a color. Background white pixels were not considered when computing the histogram. Therefore, given two images $\mathbf{I}_y$ and $\mathbf{I}_s$, we computed the Jensen-Shannon divergence of their histograms and obtained the color distance.

Flatness. We expect each closed region to be in an identical color. To measure whether the generated icons fulfill this requirement, we detect closed regions in each icon sketch $\mathbf{I}_c$ and estimate the color variation of pixels in each region. A simple method can achieve detection because open boundaries do not exist in the collected icons. Specifically, we compute

$$\frac{1}{N} \sum_{m=1}^{M} \sum_{n=1}^{N_m} \left\| \mathbf{p}_m^n - \bar{\mathbf{p}}_m \right\|_1 , \tag{11}$$

where M is the number of the closed regions, N_m is the number of pixels in region m, $N = \sum_{m=1}^{M} N_m$, $\mathbf{p}$ indicates the pixel color, and $\bar{\mathbf{p}}_m$ is the the mean color of region m.

Fréchet Inception Distance (FID). FID has widely been used to measure the similarity of visual features be-

Figure 6. We compared the icons colorized by using our method and the baselines. Experiment results indicate that our method outperformed the baselines because of clear structures and unnoticeable color gradients at flat regions.

Method	Structure	Color	Flatness	FID
Anime	0.53 ± 0.41	0.39 ± 0.12	1.78 ± 1.54	52.57
Comi	1.12 ± 0.65	0.37 ± 0.10	2.10 ± 1.75	93.03
MUNIT	1.96 ± 1.07	0.23 ± 0.09	2.67 ± 2.00	58.42
ASCFT	0.39 ± 0.27	0.42 ± 0.11	1.49 ± 1.24	63.02
AdvIcon	0.30 ± 0.40	0.37 ± 0.11	1.52 ± 1.53	40.86
Ours	$\mathbf{0.12 \pm 0.20}$	$\mathbf{0.22 \pm 0.07}$	$\mathbf{0.62 \pm 0.73}$	**27.96**

Table 1. We evaluate the quality of the generated icons by measuring the structure distance, color distance, flatness, and FID scores. The means and standard deviations are listed. The lower values indicate the better results. The best results are in bold face.

tween real and generated images. We applied this measure to evaluate whether the generated icons were realistic.

The numbers in Table 1 indicate that our method outperformed all of the baselines in terms of structure distance, color distance, flatness, and FID. This is not surprising because our method can effectively reduce color gradients at flat regions and enhance color discontinuity at boundary.

5.2. Ablation Studies

We conducted an ablation study to demonstrate the effectiveness of our pixel-wise decoder, contour extractor, and NormConv layer. The results in Figure 7 show that the CNN decoder memorized structure information in the style images and failed to generate icons fulfilling the sketch condition. The results also indicate that our contour extractor and NormConv layer can considerably reduce color gradients and enhance object structures when colorization.

Anime	Comi	MUNIT	ASCFT	AdvIcon	Ours
94 ± 4.1	155 ± 9.4	151 ± 8.1	141 ± 6.0	192 ± 9.8	905 ± 17.2

Table 2. The user study result obtained from 126 participants. We show the mean and standard deviations of the points of each method over 13 questions. The higher it is, the better. The results of each question can be obtained from our supplemental material.

5.3. Style Interpolation

Users may want to interpolate styles when colorizing icons. There are two ways to achieve style interpolation. The first is to interpolate coordinates defined in the color image scale (Figure 2), and the second is to interpolate style vectors learned from real data. Figure 8 compares the results interpolated in these two coordinate systems. An identical Gaussian vector was used to generate the results. We refer readers to our supplemental material for more results.

5.4. User Study

We have conducted a user study for subjective evaluation. Specifically, we chose the icons colorized using base-

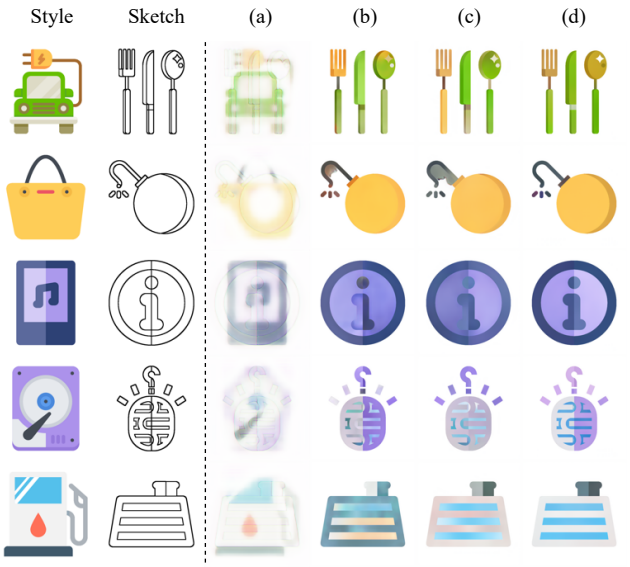

Figure 7. Ablation study. (a) A CNN decoder identical to the work of [35] was used to colorize icons. (b) Our network was trained without using the contour extractor E_x. (c) The NormConv layer was removed from the contour extractor. (d) Our results.

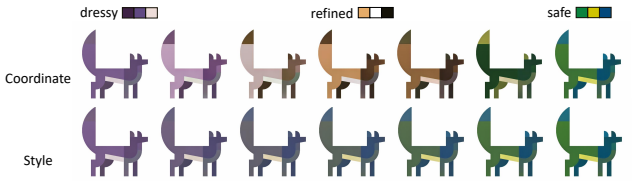

Figure 8. The refined style appears when coordinates in the color image scale are interpolated since it is in between the dressy and the safe styles. The refined style would disappear if style vectors are interpolated.

lines and our method shown in Figure 6 and created a web page for participants to select their most preferred result. The page starts with a brief tutorial and then 13 questions. Each question shows a sketch, a referenced style image, and the icons colorized using different methods. Each method would get one point if its colorized icon was preferred the most. The orders of the questions and the icons were randomly determined. We posted the page on a social media for participants to answer. Table 2 shows the study results.

5.5. Limitations

Although our generated flat-color icons are visually appealing, there is still space for improvement. Specifically, the network may (1) fill gradient colors in large regions, (2) mis-colorize foreground and background regions if the sketch is semantically ambiguous, and (3) use perceptually less distinguishable colors in nearby regions. Figure 9 shows several failure cases. In addition, since we apply

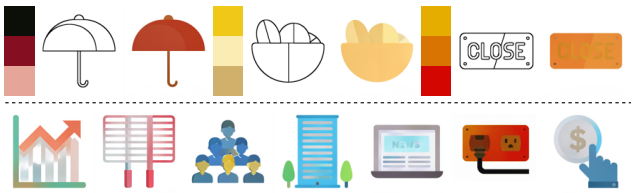

Figure 9. Failure results generated by our network. (Top) Nearby regions are in perceptually less distinguishable colors. (Bottom) Gradient colors and mis-colored areas appear.

a normalizing flow to sample style vectors, it requires users to specify a temperature when using our system. The high temperature enables the normalizing flow to sample diverse style vectors with the price of deviating from the given style condition. The low temperature is in the opposite situation. In the future, we will explore the strategy to achieve both high diversity while faithfully fulfilling the style condition.

6. Conclusions

We have presented an icon colorization system that is composed of an encoder-decoder network and a conditional normalizing flow. We design novel network architectures, including a pixel decoder, a NormConv layer, and a contour extractor, to generate flat-color icons that exhibit clear structures. The conditional normalizing flow enables the colorization network to generate diverse results conditioned on the given style. Experiment results and objective evaluations demonstrate the effectiveness of our system. We point out that this icon colorization system is beneficial to designers because they can focus on only sketching shapes and structures when creating icons. Our approach is in charge of the colorization task and provides users with diverse results, which may further inspire designers and lead to a virtuous cycle. In addition, the presented network architectures and training strategies can be beneficial to manga colorization. We will also extend our network to colorize cartoon and retain temporal coherence among frames in the future.

Acknowledgment

We thank the anonymous reviewers for their constructive comments. We are grateful to Wen-Hsiao Peng, Wei-Chen Chiu, and Hung-Kuo Chu for insightful discussions, and to all participants who joined the user study. This work was supported by the Ministry of Science and Technology, Taiwan (110-2221-E-A49 -062 - and 109-2221-E-009 -097 -).

References

[1] Rameen Abdal, Peihao Zhu, Niloy J Mitra, and Peter Wonka. Styleflow: Attribute-conditioned exploration of stylegan-generated images using conditional continuous normalizing

flows. *ACM Transactions on Graphics (TOG)*, 40(3):1–21, 2021. 1, 2

[2] Kenta Akita, Yuki Morimoto, and Reiji Tsuruno. Colorization of line drawings with empty pupils. In *Computer Graphics Forum*, volume 39, pages 601–610. Wiley Online Library, 2020. 2

[3] Navaneeth Bodla, Gang Hua, and Rama Chellappa. Semi-supervised fusedgan for conditional image generation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 669–683, 2018. 2

[4] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. *arXiv preprint arXiv:1806.07366*, 2018. 2, 3

[5] 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. 2

[6] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014. 2

[7] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016. 2

[8] Chie Furusawa, Kazuyuki Hiroshiba, Keisuke Ogaki, and Yuri Odagiri. Comicolorization: semi-automatic manga colorization. In *SIGGRAPH Asia 2017 Technical Briefs*, pages 1–4. 2017. 2, 6

[9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014. 2

[10] Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, 2018. 1, 2, 3, 5

[11] Qin-Ru Han, Wen-Zhe Zhu, and Qing Zhu. Icon colorization based on triple conditional generative adversarial networks. In *International Conference on Visual Communications and Image Processing (VCIP)*, pages 391–394, 2020. 2

[12] Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. Flow++: Improving flow-based generative models with variational dequantization and architecture design. In *International Conference on Machine Learning*, pages 2722–2730. PMLR, 2019. 2

[13] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 172–189, 2018. 2, 6

[14] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017. 2

[15] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019. 2

[16] Hyunsu Kim, Ho Young Jhoo, Eunhyeok Park, and Sungjoo Yoo. Tag2pix: Line art colorization using text tag with secat and changing loss. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9056–9065, 2019. 2

[17] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim. Learning to discover cross-domain relations with generative adversarial networks. In *International Conference on Machine Learning*, pages 1857–1865. PMLR, 2017. 2

[18] Diederik P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *arXiv preprint arXiv:1807.03039*, 2018. 2

[19] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 2

[20] Shigenobu Kobayashi. Color image scale. *Kodansha Amer Inc.*, 1992. 1, 3, 4

[21] Guillaume Lample, Neil Zeghidour, Nicolas Usunier, Antoine Bordes, Ludovic Denoyer, and Marc'Aurelio Ranzato. Fader networks: Manipulating images by sliding attributes. *arXiv preprint arXiv:1706.00409*, 2017. 2

[22] Hsin-Ying Lee, Hung-Yu Tseng, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang. Diverse image-to-image translation via disentangled representations. In *Proceedings of the European conference on computer vision (ECCV)*, pages 35–51, 2018. 2

[23] Hsin-Ying Lee, Hung-Yu Tseng, Qi Mao, Jia-Bin Huang, Yu-Ding Lu, Maneesh Singh, and Ming-Hsuan Yang. Drit++: Diverse image-to-image translation via disentangled representations. *International Journal of Computer Vision*, 128(10):2402–2417, 2020. 2

[24] 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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5801–5810, 2020. 2, 6

[25] Jianxin Lin, Yingce Xia, Tao Qin, Zhibo Chen, and Tie-Yan Liu. Conditional image-to-image translation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5524–5532, 2018. 2

[26] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In *Advances in neural information processing systems*, pages 700–708, 2017. 2

[27] Andreas Lugmayr, Martin Danelljan, Luc Van Gool, and Radu Timofte. Srflow: Learning the super-resolution space with normalizing flow. In *European Conference on Computer Vision*, pages 715–732. Springer, 2020. 2

[28] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014. 2

[29] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *arXiv preprint arXiv:1912.02762*, 2019. 2

[30] Taesung Park, Jun-Yan Zhu, Oliver Wang, Jingwan Lu, Eli Shechtman, Alexei Efros, and Richard Zhang. Swapping autoencoder for deep image manipulation. *Advances in Neural Information Processing Systems*, 33:7198–7211, 2020. 2

[31] Lev Semenovich Pontryagin, EF Mishchenko, VG Boltyanskii, and RV Gamkrelidze. The mathematical theory of optimal processes. 1962. 3

[32] Hui Ren, Jia Li, and Nan Gao. Two-stage sketch colorization with color parsing. *IEEE Access*, 8:44599–44610, 2019. 2

[33] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 4

[34] Patsorn Sangkloy, Jingwan Lu, Chen Fang, Fisher Yu, and James Hays. Scribbler: Controlling deep image synthesis with sketch and color. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5400–5409, 2017. 2

[35] Tsai-Ho Sun, Chien-Hsun Lai, Sai-Keung Wong, and Yu-Shuen Wang. Adversarial colorization of icons based on contour and color conditions. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 683–691, 2019. 1, 2, 5, 6, 8

[36] Benigno Uria, Marc-Alexandre Côté, Karol Gregor, Iain Murray, and Hugo Larochelle. Neural autoregressive distribution estimation. *The Journal of Machine Learning Research*, 17(1):7184–7220, 2016. 2

[37] Xiaolong Wang and Abhinav Gupta. Generative image modeling using style and structure adversarial networks. In *European conference on computer vision*, pages 318–335. Springer, 2016. 2

[38] Xinchen Yan, Jimei Yang, Kihyuk Sohn, and Honglak Lee. Attribute2image: Conditional image generation from visual attributes. In *European Conference on Computer Vision*, pages 776–791. Springer, 2016. 2

[39] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. Pointflow: 3d point cloud generation with continuous normalizing flows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4541–4550, 2019. 5

[40] Lvmin Zhang, Yi Ji, Xin Lin, and Chunping Liu. Style transfer for anime sketches with enhanced residual u-net and auxiliary classifier gan. In *2017 4th IAPR Asian Conference on Pattern Recognition (ACPR)*, pages 506–511. IEEE, 2017. 2, 6

[41] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9889–9898, 2021. 2

[42] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A. Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. *CoRR*, abs/1711.11586, 2017. 2

[43] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017. 2