Adversarial Colorization Of Icons Based On Structure And Color Conditions

Tsai-Ho Sun
National Chiao Tung University
Hsinchu, Taiwan
locha0519@gmail.com

Chien-Hsun Lai
National Chiao Tung University
Hsinchu, Taiwan
jxcode.tw@gmail.com

Sai-Keung Wong
National Chiao Tung University
Hsinchu, Taiwan
cswingo@cs.nctu.edu.tw

Yu-Shuen Wang
National Chiao Tung University
Hsinchu, Taiwan
yushuen@cs.nctu.edu.tw

ABSTRACT

We present a system to help designers create icons that are widely used in banners, signboards, billboards, homepages, and mobile apps. Designers are tasked with drawing contours, whereas our system colorizes contours in different styles. This goal is achieved by training a dual conditional generative adversarial network (GAN) on our collected icon dataset. One condition requires the generated image and the drawn contour to possess a similar contour, while the other anticipates the image and the referenced icon to be similar in color style. Accordingly, the generator takes a contour image and a man-made icon image to colorize the contour, and then the discriminators determine whether the result fulfills the two conditions. The trained network is able to colorize icons demanded by designers and greatly reduces their workload. For the evaluation, we compared our dual conditional GAN to several state-of-the-art techniques. Experiment results demonstrate that our network is over the previous networks. Finally, we will provide the source code, icon dataset, and trained network for public use.

CCS CONCEPTS

• **Information systems → Multimedia content creation**; • **Human-centered computing → Interactive systems and tools**.

KEYWORDS

Icon, colorization, generative adversarial networks

ACM Reference Format:
Tsai-Ho Sun, Chien-Hsun Lai, Sai-Keung Wong, and Yu-Shuen Wang. 2019. Adversarial Colorization Of Icons Based On Structure And Color Conditions. In *Proceedings of the 27th ACM International Conference on Multimedia (MM '19), October 21–25, 2019, Nice, France.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3343031.3351041

MM '19, October 21–25, 2019, Nice, France

ACM ISBN 978-1-4503-6889-6/19/10…$15.00
https://doi.org/10.1145/3343031.3351041

1 INTRODUCTION

Nowadays, icons are widely utilized in banners, signboards, billboards, homepages, and mobile apps. Effective icons are usually simple but distinguishable, so that users can quickly receive the intended information when seeing them at a small size or a long distance. Considering aesthetics and practical issues, designing an eye-catching icon is challenging. Designers have to carefully consider not only shapes and structures, but also colors, when they create icons for their customers. Moreover, icons are not self existent. When they appear on a signboard or a website with letters and backgrounds, the styles of these different components should be consistent, which makes icon design more challenging. The difficulties motivate us to build a system that can reduce designers' workload. Specifically, designers draw the contour of an icon, while our system is in charge of colorization.

Generative adversarial networks (GANs) [10] have been proven to be able to generate realistic images in many applications [4, 22, 24, 25, 30, 39], and could constitute a solution to help designers colorize icons. Specifically, a network takes a contour image drawn by the designers as input and then outputs the colorized icon image. Similar ideas have been adopted to colorize black-and-white Manga characters [5, 8, 14, 44] and achieved great success. To control the colorization process, additional inputs, such as stroke colors and style images, are fed into the network as well. The features extracted from both contour and style images will be fused and used for the colorization. However, training the above-mentioned networks to achieve icon colorization is inadequate because icons exhibit diverse styles and structures. While the discriminator is not strong enough to recognize man-made and machine-generated icons, its guidance to train the generator is inappropriate.

Observing that an icon can be well defined by color and structure conditions, we present a dual conditional GAN (Figure 1) to colorize icons. Rather than training a discriminator to recognize whether an icon is man-made or machine-generated, we train two discriminators to determine whether paired images are similar in structure and color style, respectively. In this way, the task assigned to each discriminator is simple and easy to accomplish. To specify the icon structure, we let users draw contours on a graphical interface. To condition the color style, a man-made icon image is selected. The two images are fed into our dual conditional GAN for icon colorization. Because it is not intuitive to apply a man-made icon to specify the color condition, in practice, we let users select a

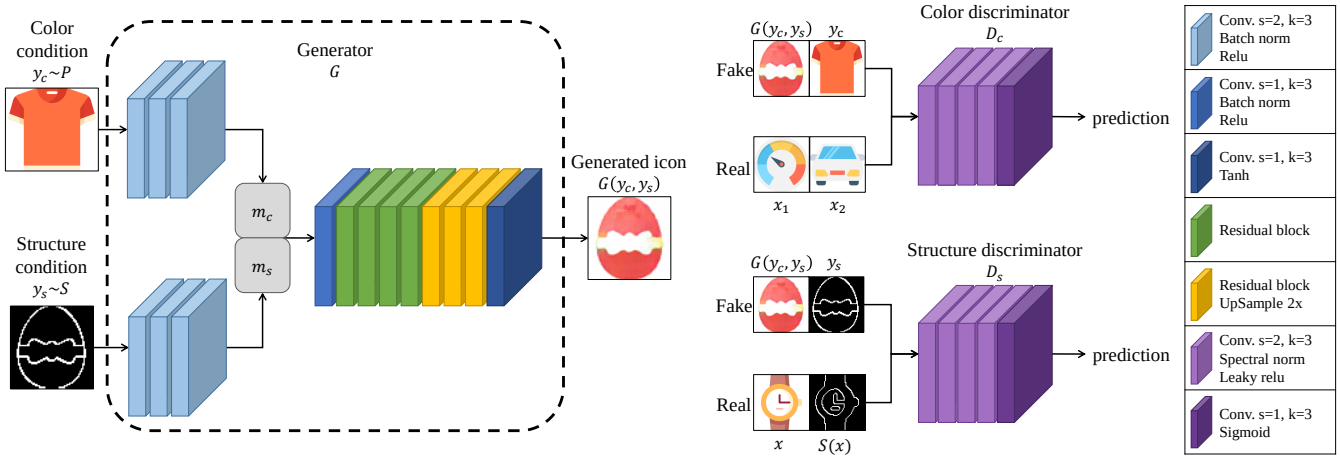

Figure 1: Our dual conditional generative adversarial network. The details of layers in different colors are on the right. k and s are the kernel size and the stride, respectively

style label when using our system to create icons. Then the system randomly selects man-made icons that match the style [20, 21] and feeds the icons to the network for colorization.

To evaluate the performance of our icon colorization method, we tested the system on several examples with diverse structures and color styles. Figures 4, 6, and 8, and our accompanying video present the results. In addition, we compared our dual conditional GAN to state-of-the-art techniques, including iGAN [46], CycleGAN [47], conditional image-to-image translation [42], ComiColorization [8], MUNIT [17] and Anime [44]. Experiment results demonstrate the effectiveness of our technique.

2 RELATED WORK

Icon Design. Although icons are widely used nowadays, creating visually appealing icons is not easy because many aspects, such as context, color, and structure, should be considered [12, 13, 15]. Extensive studies have conducted on issues about the message quality, metaphor, and styling of icons [16], and the effects of icon spacing and size [28]. Among the above-mentioned aspects, color is visually essential to make icons attractive, legible and viewer-friendly [26]. The optimal choice of color combinations and icon shapes can convey information both clearly and pleasantly.

Generative Adversarial Networks. GAN was first presented by Goodfellow *et al.* [10] and then widely used in realistic image generation [4, 18, 22, 24, 25, 30, 37–40]. The network typically contains a generator and one or multiple discriminators, which are trained iteratively and alternatively to surpass one another. In spite of tremendous advantages, training a GAN is challenging because of gradient vanish and stability problems. To facilitate network training, several methods, such as energy-based GANs [45], minibatch discrimination [36], Wasserstein GANs [1, 11], boundary equilibrium GANs [2], and spectral normalization [32], were presented.

Conditional GANs and Domain Transfer. Earlier versions of GANs are not controllable because the inputs are noise latent vectors. Afterward, to control the results, additional features or conditions are fed into the network. The features can be labels [22, 31, 41], images [18, 29], and sentences [3, 43]. Although results generated by the conditional GANs are impressive, most of them are supervised; and they demand a large number of labels and matching image pairs. Thus, unsupervised techniques were presented to map images from one domain to another [37, 42, 47]. The methods retain either latent codes or reconstructions during the domain transformation to achieve the goal. To avoid the mode collapse problem that frequently occurs in GANs, Zhu *et al.* [48] combined the conditional variational autoencoder GAN [23] and the conditional latent regressor GAN [6, 7] to generate diverse and realistic results.

Manga and Cartoon Colorization. In the past, coloring black-and-white Manga was achieved by considering hand crafted features, such as pattern-continuity and intensity-continuity [34]. While such features are difficult to define, deep neural networks were presented to achieve the goal by learning from data automatically. By providing a line art and several guided stoke colors, the system can be used to colorize Manga [5]. In addition to stroke colors, several methods let users provide reference images for guiding results. Among them, Furusawa *et al.* [8] adopted a convolutional encoder-decoder network with an additional discriminator to colorize black-and-white Manga pages. Hensman *et al.*'s method [14] required only one single reference image to train the conditional GAN [18], and used the network to colorize monochrome images. Zhang *et al.* [44] fused the high level features extracted from sketch and style images to generate the color version of a sketch. They also applied two guided decoders to prevent the residual U-net from skipping high level features.

The aforementioned networks have achieved great success on colorizing Manga characters. For our icon colorization task, however, they may fail because icons are not only diverse in color style but also in structure. Furthermore, we want to achieve that boundaries of regions are formed by color difference between the regions. Because man-made icons are difficult to define, a single discriminator cannot determine whether the generated result is meaningful. To tackle this problem, we presented a dual conditional

GAN that anticipates the generated results to fulfill structure and color conditions. Because the task assigned to each discriminator is simple, the network is easy to train and able to generate visually appealing results.

3 STRUCTURE AND COLOR OF AN ICON

Given a contour image and a man-made icon, our system strives to generate a result, in which the structure is similar to the contour and the color style is similar to the referenced man-made icon. To achieve this goal, we train a conditional generative adversarial network on an icon dataset[1] that contains 12,575 images. These icons contain multiple colors, with a white background, but without dark border lines. We apply image processing techniques to automatically extract contours and color styles from these icons for network training. No manual labeling tasks are needed.

3.1 Structure Condition

We represent the structure condition by a binary contour image. White and black pixels in the image indicate edge and non-edge regions, respectively. To obtain this contour image, the Canny edge detection algorithm is adopted. Intuitively, each icon and its corresponding contour can match, and otherwise cannot.

3.2 Color Condition

The color condition is specified by the referenced icon image. To determine whether two icons match in color style, we compute a 3D Lab color histogram ($8 \times 8 \times 8$) of each of them, and then measure their distance. In the pre-processing step, we apply the K-means ($K = 500$) clustering method to merge icons if their color histograms are close to each other. Icons in the same cluster are considered similar in color style.

4 NETWORK TRAINING

We train a dual conditional GAN to colorize icons. Figure 1 shows the network architecture and the details of each layer. The inputs of the network are a contour image and a referenced icon image, with a resolution $64 \times 64 \times 3$. The former and the latter inputs are the structure and the color conditions, respectively. They are encoded, concatenated together, and then fed into the generator. The generator then takes the code to generate a fake icon. After that, two discriminators are used to guide the generator creating results that can fulfill the input conditions. To train the structure discriminator, the contour image is combined with the corresponding icon and the generated icon to form the real and the fake pairs, respectively; and then they are fed into the structure discriminator. To train the color discriminator, the real pair is formed by two arbitrary icons that are classified in the same cluster, whereas the fake pair is formed by the referenced and the generated icon. It is worth noting that the purpose of this discriminator is to judge whether two icons are similar in color style. The real and the fake pairs can be dissimilar in structure.

[1] https://www.flaticon.com/

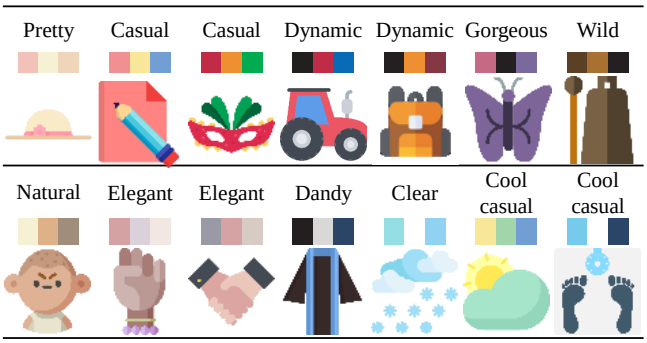

Figure 2: The semantic labels, corresponding color combinations, and the example icons.

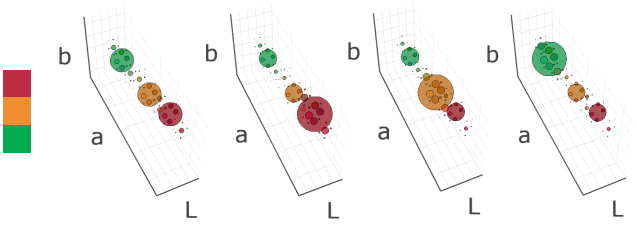

Figure 3: We transform a color combination (left) to four histograms (right) according to the ratios 1:1:1, 2:1:1, 1:2:1, and 1:1:2. The sphere size indicates the bin size. Note that the histograms have been smoothed by Gaussian blur.

4.1 Loss functions

The dual conditional GAN is trained by optimizing two adversarial losses. Let P and S be the distributions of man-made icons and the corresponding contours, respectively. We also let $x \in P$, and $S(x)$ be the contour of x. To fulfill the structure condition, the loss is defined as:

$$L_s(G, D_s) = \mathbb{E}_{y_c \sim P, y_s \sim S}[\log(1 - D_s(G(y_c, y_s), y_s))] + \mathbb{E}_{x \sim P}[\log D_s(x, S(x))], \quad (1)$$

where $G(y_c, y_s)$ is the result generated according to a reference icon y_c and a contour image y_s; and D_s is a conditional discriminator that determines whether the paired images have the same structure. To fulfill the color condition, we apply a similar strategy. Let $k(x)$ be the cluster index of x. We present the loss as:

$$L_c(G, D_c) = \mathbb{E}_{y_c \sim P, y_s \sim S}[\log(1 - D_c(G(y_c, y_s), y_c))] + \mathbb{E}_{x_1, x_2 \sim P, k(x_1)=k(x_2)}[\log D_c(x_1, x_2)], \quad (2)$$

where D_c is a conditional discriminator that determines whether two images are similar in colors. Recall that we apply the K-means clustering to group icons that are similar in color. The real pair, x_1 and x_2, are two arbitrary icons in the same cluster. Finally, the full objective function is

$$G^* = \arg\min_G \max_{D_s, D_c} (L_s(G, D_s) + L_c(G, D_c)). \quad (3)$$

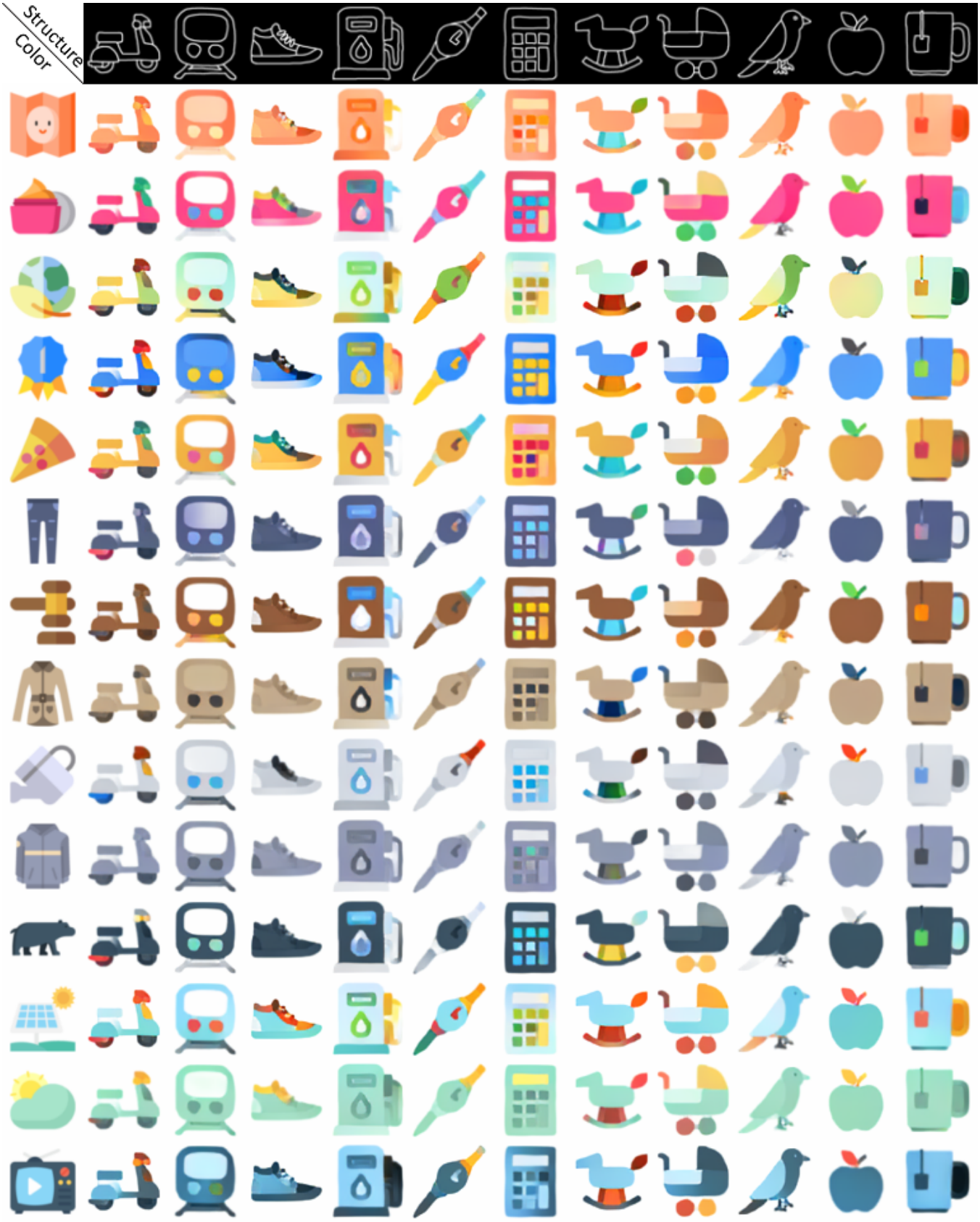

Figure 4: We applied our trained network to colorize icons in various styles. The first row shows contours, and the first column shows man-made icons that specify color conditions.

4.2 Training Details

We trained the dual conditional GAN by using the Adam optimizer [19] on a single NVIDIA GeForce 1080Ti. The learning rate was set to 10^{-4}; the hyper parameters were initialized by using Xavier initialization [9]; and the batch size was set to 64. In each epoch, the generator G, the color discriminator D_c, and the contour discriminator D_s were updated sequentially based on the stochastic gradient of G^*. We repeated the process 1,000 epochs until the loss was unable to decrease and the system started overfitting. It is worth noting that we did not use latent noise in the network.

4.3 Semantic Style Labels

Since applying a referenced icon to specify the color condition is not intuitive, we let users simply select a style label when using our system to create icons. Specifically, we consider the color psychology theory [20] and define the style of a man-made icon according to its color combination. In our current implementation, each color combination contains three major colors; and each combination refers to a style. For example, an icon that contains #C02C46, #ED8E32, and #01AC50 can be considered *casual*, while an icon that contains #94DEE2, #FFFFFF, and #91D2F1 can be considered *clear*. Figure 2 shows the illustration. Accordingly, when the users select a style label, our system randomly selects a number of man-made icons that fulfill the style, and then feeds the icons to the network as color conditions. We recommend readers to watch our accompanying video for this intuitive user interface, as user interactions are difficult to be visualized in still images.

To determine whether an icon image p matches a color combination (style) q, we compute their 3D Lab color histograms ($8 \times 8 \times 8$) for comparison. When determining the histogram of an icon H_p, white background pixels are not considered. In addition, because colors in adjacent bins can be visually similar, but are considered different, we apply 3D Gaussian blur to the histogram to reduce the difference between perception and statistics. Then, each bin is normalized by the total pixel number. Different to icon images, the definition of color combinations is rough. We generate four histograms H_{q1} - H_{q4} for each color combination based on the ratios (1:1:1, 2:1:1, 1:2:1, and 1:1:2) of three major colors, as illustrated in Figure 3. Finally, the icon is labelled as style q if its color histogram H_p is similar to either H_{q1} - H_{q4}.

5 RESULTS AND EVALUATIONS

Several contour images, which contain straight and curved lines, and small and large open areas, were tested on our system. As can be seen in Figure 4, these machine generated results were similar to man-made icons. For example, the results were in flat colors; only foreground objects were colorized; and most of the noticeable color boundaries could match the specified contours. The generated results are well elaborated and look like carefully designed icons. Interestingly, the results generated by considering the same reference icon were very similar in color style. This property is helpful to designers if they need to colorize a set of icons in a particular style. Considering that previous colorization methods may suffer from color leaking artifacts, we tested our system on several open contours. Figure 5 shows that the system is leak-proofing. Users are not expected to draw contours carefully when using it.

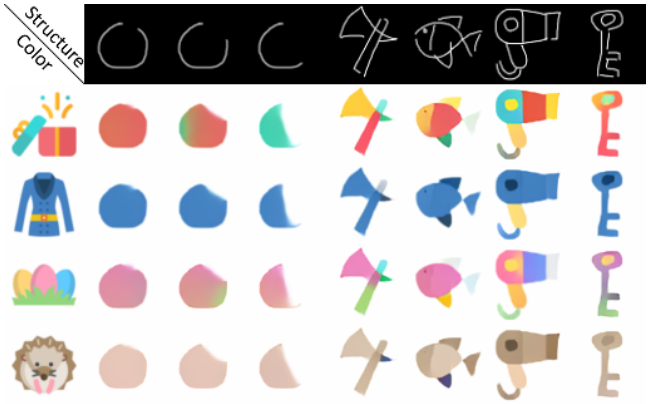

Figure 5: Our system can colorize icons that are conditioned by open contours without causing color leaking problems.

Figure 6: Step by step results. Our system generates results interactively while the input contour is incrementally enhanced. Each row shows the results based on the referenced icon image on the left. Notice that the colors were not changed considerably in subsequent steps once the referenced icon was selected.

The trained network can colorize icons in real-time by leveraging the GPU (graphics processing unit) resources. Hence, in our implementation, we feed the contour image and the referenced icon image into the network whenever a stroke is updated and then display the generated result immediately. Figure 6 shows the results for the input contours that are incrementally enhanced.

5.1 Representations of a Color Condition

Several ways exist to represent the color condition of an icon. In addition to an image, we input a 3D Lab color histogram to the network and compared the results generated by these two different representations. Specifically, we duplicated the $8 \times 8 \times 8$ histogram to form an $8 \times 8 \times 256$ feature map. Therefore, on the generator part, the two feature maps determined from the contour image and the

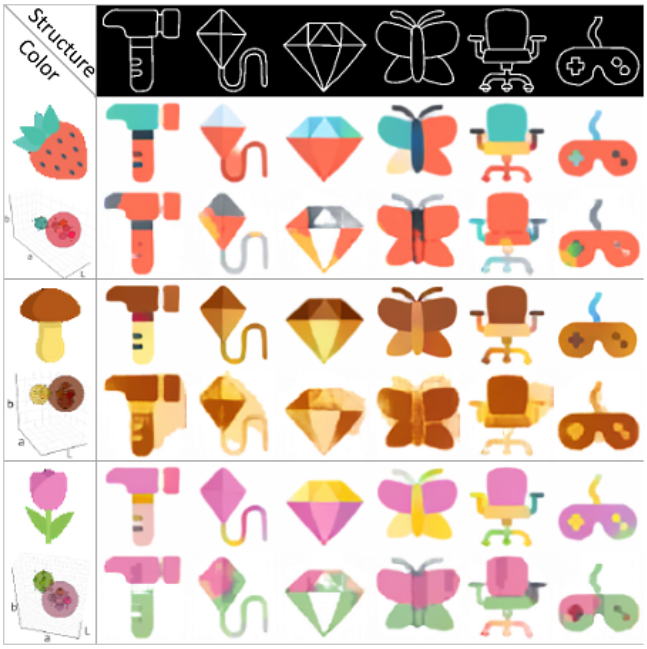

Figure 7: We applied different color conditions (i.e., image and histogram) to guide the network and compared their results.

histogram can be concatenated. Regarding the discriminator part, we encoded the generated icon from $64 \times 64 \times 3$ to $8 \times 8 \times 256$ by three 2D convolutions so as to concatenate with the feature map of a color histogram.

Figure 7 shows that both images and histograms are able to condition the color of the generated icons. However, the network can learn additional features if the color condition is represented by images. For example, a (nearly) closed region is in the same color, whereas adjacent regions are in different colors. In addition, we observed that the network adopts different strategies if the color and the structure conditions conflict with each other, which may occur when complexities of the contour image and the referenced icon image are considerably different. As shown in Figure 7, if the color condition is represented by a histogram, the generated icons may contain unnecessary edges/structures; if the condition is represented by an image, the generated and the conditioning icons may have different dominant colors.

5.2 Comparison to State-of-the-Art Techniques

We compared our method to the state-of-the-art techniques to evaluate its effectiveness. All the methods were trained on our collected icon dataset for the comparison.

iGAN. Interactive GAN (iGAN) [46] can produce samples that best satisfy user edits in real-time. The system is based on DCGAN [35], which optimizes the latent vector to generate results specified by the color and the shape of brush strokes. As can be seen in Figure 8, the results of iGAN are not satisfactory because low level features are noisy. In other words, although the generated icons to some extent fulfill the structure conditions, they are unsatisfactory.

Domain Transfer Methods. CycleGAN [47], CImg2Img [27], and MUNIT [17] are well known methods that can transform images from one domain to another. Thus, we were curious whether they were able to transform images in the contour domain to images in the color icon domain as well. The codes of these methods were obtained from the authors. As can be seen in Figure 8, given contour images, the results generated by their networks to some extent contain the features of icons, such as flat colors and simple lines. However, the results are not anticipated because their structures differ from the given contours. We suspect the reasons as follows. Given two domains X and Y, the goal of domain transfer is to transform samples in X to samples in Y, denoted as Y', and then back to samples in X, denoted as X'. Another pass is from Y to X and then back to Y. The relations between X and Y are learned by the network itself. Normally, since the network has to transform samples in Y' back to samples in X, the samples in Y' must contain some features in X to facilitate transformation. However, this does not mean that all features in Y' are all related to X. A part of them is not used. For example, in the icons generated by CycleGAN, CImg2Img, and MUNIT, although a part of the edges can be matched to the contour images, a part of them cannot. In general, the redundant features make the results deviate from our expectation. If the domain Y is narrow, such as a face dataset, the problem is not serious because the redundant features is suppressed by the discriminator. However, if the domain Y is wide, such as the icon dataset, the redundant features become noticeable. That a GAN can generate face images from a random latent vector, but may fail to generate icons (Figure 9), supports this assertion. Notice that the faces are recognizable, although there are many redundant features (artifacts).

Manga/Cartoon Colorization. The goal of our system is similar to Manga colorization. Both of them attempt to colorize contour images. Therefore, we compared our system to a famous online software called style2paints[2], which was an improved version of [44]. However, because the details of this online software was not disclosed and the network was trained on Manga images, the comparison may not be fair. We further implemented the original network [44] and trained the network on our collected icon dataset. Figure 8 shows the results. As can be seen, the interior structures of the icons generated by the method (Anime) are barely matched with the input structures. The network of [44] does not function well because the discriminator is trained to determine man-made and machine-generated icons. The domain is too wide and the task is too difficult. In addition, the training strategy makes the generator overfit easily. During the training phase, the input contour and style images are very similar in structure. However, during the testing phase, the two images are different.

We also compared our system to Comicolorization [8]. The codes were obtained from the authors. In the beginning, we failed to train the network by following their procedure because the discriminator cannot recognize icons. Hence, we reduced the weight of the adversarial loss to 0.05 and then succeeded. In other words, the network was trained mostly based on the reconstruction loss. Since the supervised learning tends to fit the whole data distribution, the generated icons often contain blurring artifacts. The structures of

[2]https://github.com/lllyasviel/style2paints

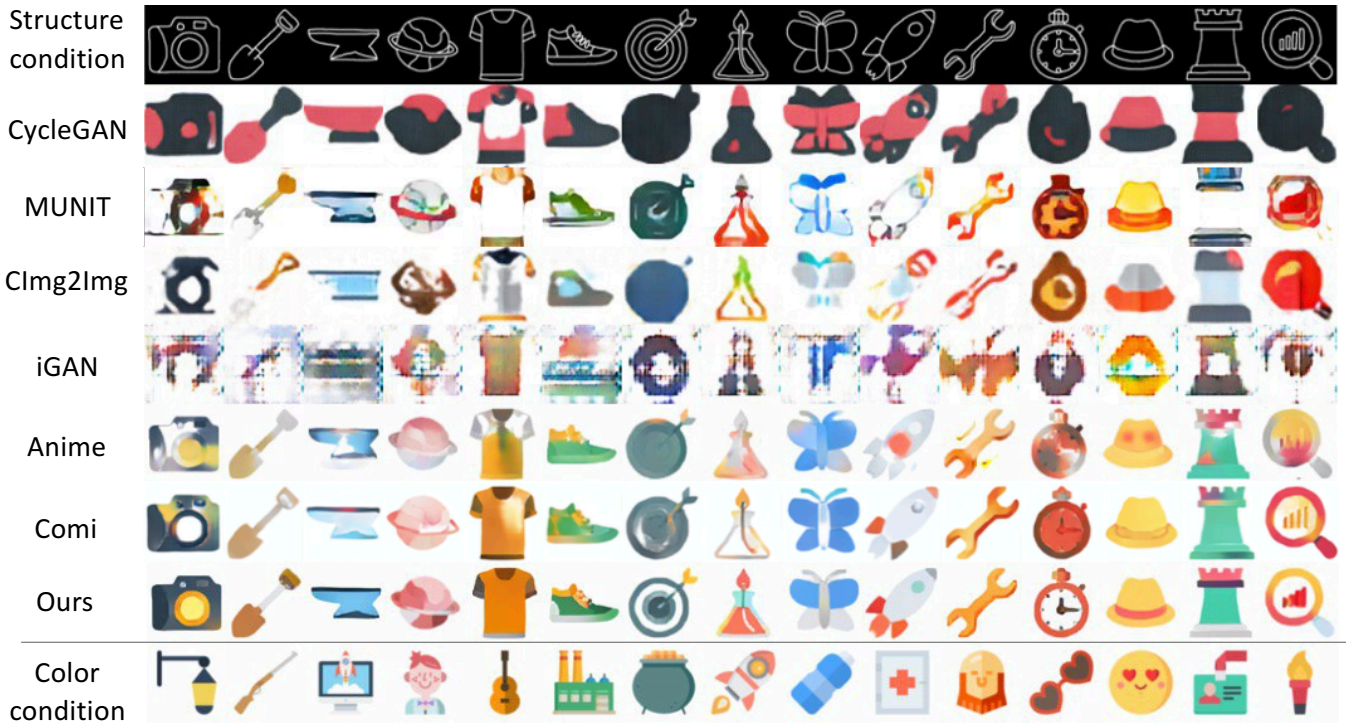

Figure 8: We compared our system to the current state-of-the-art techniques. From top to bottom rows are the contour images, the icons generated by CycleGAN [47], MUNIT [17], CImg2Img [27], iGAN [46], Anime [44], Comi [8], our system, and the icons referenced by Anime and our system.

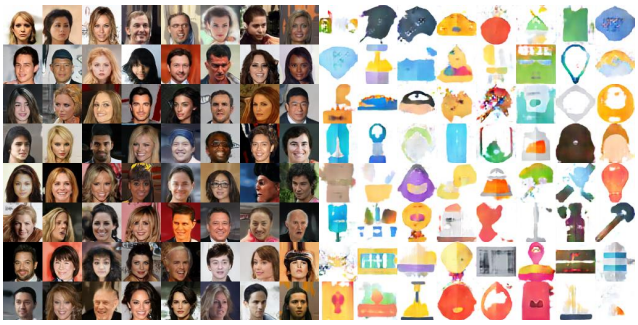

Figure 9: We trained a SNGAN [33] on celeba and our collected icon datasets, respectively. It fails to generate icons (right) due to complex structures and styles.

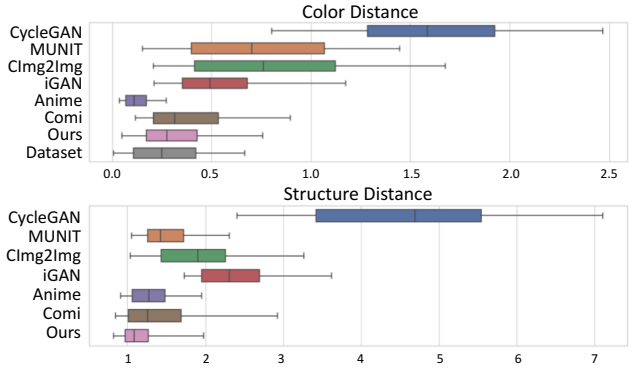

Figure 10: Fulfillment of color and structure conditions among the previous and our methods. The five-number summary (from left to right on the box and whisker plots) consists of the 5^{th}, 25^{th}, 50^{th}, 75^{th}, and 95^{th} percentiles.

their generated icons (Comi) are not as sharp/clear as the structures of ours (Figure 8).

5.3 Objective Evaluations

In addition to visual comparison, we quantitatively evaluated the generated results. Color distances and structure distances between the generated icons and the conditions were computed. To measure the fulfillment of color condition, we computed the Jensen-Shannon divergence of 3D Lab color histograms for the evaluation. To measure the fulfillment of structure condition, we first applied the Canny edge detection method to the generated icon image. Afterward, bi-directional search of the closest edge pixels is adopted to compute the distance between the generated and the conditioned contours. Specifically, for each edge pixel p in one image, we

searched the edge pixel q in the other that is closest to p, and computed the mean distance D_{pq}. The two images were then switched for estimating the mean distance D_{qp}. In other words, we measured the fulfillment of structure condition by $\frac{1}{2}(D_{pq} + D_{qp})$.

The box and whisker plots in Figure 10 show the evaluation results. Clearly, CycleGAN, MUNIT, CImg2Img and iGAN could not fulfill the color condition because of the monotonic or noisy colors. It is worth noting that the icons generated by Anime were the most similar to the referenced icons in colors. By comparing the statistic and the results shown in Figure 8, we found that Anime tends to copy colors from one image to another when colorizing an icon. However, this strategy is inappropriate because the contour image and the referenced icon were different in structure. The number and the sizes of contours cause the conflict. To verify this inference, we randomly picked two man-made icons that were classified in the same cluster (Section 3.2) and computed their mean color distance. The statistic shows that man-made and our colorized icons were well matched. In contrast to the color distance, the shorter contour distance is the better because ideally we expect the generated icon and the contour image to be the same in structure. Again, the distances show that icons generated by CycleGAN, MUNIT, CImg2Img, and iGAN were dissimilar to the specified contours. Anime and Comi could fulfill the overall structure condition. However, our method did a better job in colorizing details.

5.4 Subjective Evaluations

We conducted a user study with 111 participants to evaluate the results generated by CyleGAN, MUNIT, CImg2Img, iGAN, Anime, Comi, and ours. Specifically, we created a questionnaire and posted it in the Internet for anonymous participants to answer. They were asked to compare and to rate icons colorized by different methods. The best to the worst icons were rated by 5 to 1, respectively. To achieve a fair comparison, the icons that were conditioned by the same color and the same structure were listed in a page. In addition, the order of the methods was randomly assigned to prevent bias.

The mean scores rated by the participants to our method, Comi, Anime, MUNIT, CycleGAN, CImg2Img, and iGAN were 3.65 (SD = 1.12), 3.27 (SD = 1.20), 2.67 (SD = 1.14), 2.07 (SD = 1.02), 1.66 (SD = 0.80), 1.57 (SD = 0.72), and 1.25 (SD = 0.64), respectively. As indicated, our method was rated the best. To understand whether the result had statistical significance, we ran a one-way ANOVA to analyze/compare the scores of the previous methods and ours. The results confirmed that our system was qualitatively better than the other methods ($p < 0.01$ for all of the comparisons).

5.5 Limitations

Generating results that are always satisfactory in semantics is difficult. Figure 11 shows several failure examples. Specifically, some icons, such as an apple, a lion, or a tree, have their own colors. Our system is likely to generate results that do not match the target semantics (e.g., colorizing an apple with blue) because it only considers whether the generated and the reference icons have the same color style. We also find that several external regions, such as the holes formed by the steps of a ladder and the bicycle frame, are mis-colorized. In addition, the structure and color conditions

Figure 11: Several failure examples generated by our system. From top to bottom are the contours, results, and the reference icons.

can conflict when the complexities of a contour image and a referenced icon are different. If a simple contour and a diverse-color icon are fed into the generator, the generated result may contain additional boundaries or gradient colors that should not appear in icons. Regarding the combination of complex contour and monotonous color, the generated results are of two types: 1) they can be as monotonous as the reference icon; or 2) they contain colors that are not in the reference icon.

6 CONCLUSIONS AND FUTURE WORKS

We have presented an interactive system to help designers create icons. This is a system that allows both humans and machines to cooperate and explore creative designs. Specifically, designers draw contours to specify the structure of an icon; then the system colorizes the contours according to the color conditions. This goal cannot be achieved by previous methods due to various structures and styles of man-made icons. Training a discriminator to recognize a man-made or a machine-generated icon is very challenging. Therefore, we divide the icon recognition task into two sub-tasks and apply a dual conditional GAN to solve the problem. Specifically, the two discriminators determine whether the structure and the color of paired images are well matched, respectively. While the generator successfully cheats these two discriminators, it can generate icons demanded by users.

The color condition of our current system is represented by an image. To improve usability, we let users specify a style label when using our system to create icons. Man-made icons that fulfill the label will be randomly selected and then fed into the network as color condition. In other words, the network has no idea about styles. Therefore, in future, we plan to train a network that can take semantic labels as input when colorizing icons. Considering that style labels are implicit, we believe that the strategy also has the potential to solve the conflict of structure and color conditions.

ACKNOWLEDGEMENTS

We thank anonymous reviewers for their insightful comments and suggestions. We are also grateful to Prof. Chun-Cheng Hsu for the valuable discussions, and all the participants who joined the user study. This work is partially supported by the Ministry of Science and Technology, Taiwan, under Grant No. 105-2221-E-009 -135 -MY3 and 107-2221-E-009 -131 -MY3.

REFERENCES

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein Generative Adversarial Networks. *Proceedings of the 34th International Conference on Machine Learning* 70 (2017), 214–223.

[2] David Berthelot, Thomas Schumm, and Luke Metz. 2017. BEGAN: boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717* (2017).

[3] Navaneeth Bodla, Gang Hua, and Rama Chellappa. 2018. Semi-supervised FusedGAN for Conditional Image Generation. In *ECCV*. https://doi.org/10.1145/2897824.2925974

[4] Yun Cao, Zhiming Zhou, Weinan Zhang, and Yong Yu. 2017. Unsupervised diverse colorization via generative adversarial networks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 151–166.

[5] Yuanzheng Ci, Xinzhu Ma, Zhihui Wang, Haojie Li, and Zhongxuan Luo. 2018. User-Guided Deep Anime Line Art Colorization with Conditional Adversarial Networks. In *Proceedings of ACM Multimedia*. 1–6.

[6] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. 2017. Adversarial feature learning. *ICLR*.

[7] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. 2017. Adversarially learned inference. In *International Conference on Learning Representations*.

[8] Chie Furusawa, Kazuyuki Hiroshiba, Keisuke Ogaki, and Yuri Odagiri. 2017. Comicolorization: semi-automatic manga colorization. In *SIGGRAPH Asia 2017 Technical Briefs*. ACM, 12.

[9] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 249–256.

[10] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.

[11] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. 2017. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*. 5767–5777.

[12] D Hawthorn. 2000. Possible implications of aging for interface designers. *Interacting with Computers* 12, 5 (2000), 507 – 528. https://doi.org/10.1016/S0953-5438(99)00021-1

[13] Steven Heim. 2007. *The resonant interface: HCI foundations for interaction design*. Addison-Wesley Longman Publishing Co., Inc.

[14] Paulina Hensman and Kiyoharu Aizawa. 2017. cGAN-based Manga Colorization Using a Single Training Image. In *Document Analysis and Recognition (ICDAR), 2017 14th IAPR International Conference on*, Vol. 3. IEEE, 72–77.

[15] William K Horton. 1994. *The icon book: Visual symbols for computer systems and documentation*. John Wiley & Sons, Inc.

[16] Shih-Miao Huang, Kong-King Shieh, and Chai-Fen Chi. 2002. Factors affecting the design of computer icons. *International Journal of Industrial Ergonomics* 29, 4 (2002), 211 – 218. https://doi.org/10.1016/S0169-8141(01)00064-6

[17] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. 2018. Multimodal unsupervised image-to-image translation. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 172–189.

[18] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. 2017. Image-to-Image Translation with Conditional Adversarial Networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), 5967–5976.

[19] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[20] Shigenobu Kobayashi. 1992. *Color Image Scale*. Kodansha Amer Inc.

[21] J. Krause. 2002. *Color Index: Over 1,100 Color Combinations, CMYK and RGB Formulas, for Print and Web Media*. F&W Publications, Inc.

[22] Guillaume Lample, Neil Zeghidour, Nicolas Usunier, Antoine Bordes, Ludovic Denoyer, et al. 2017. Fader networks: Manipulating images by sliding attributes. In *Advances in Neural Information Processing Systems*. 5967–5976.

[23] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. 2016. Autoencoding Beyond Pixels Using a Learned Similarity Metric. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*. 1558–1566.

[24] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew P Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. 2017. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network.. In *CVPR*, Vol. 2. 4.

[25] Chuan Li and Michael Wand. 2016. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *European Conference on Computer Vision*. Springer, 702–716.

[26] Yan-Peng Lim and Peter Charles Woods. 2010. *Visual Information Communication*. Springer, Chapter: Experimental Color in Computer Icons, 149–158.

[27] Jianxin Lin, Yingce Xia, Tao Qin, Zhibo Chen, and Tie-Yan Liu. 2018. Conditional Image-to-Image Translation. In *CVPR*. 5524–5532.

[28] Tomas Lindberg and Risto Näsänen. 2003. The effect of icon spacing and size on the speed of icon processing in the human visual system. *Displays* 24, 3 (2003), 111–120.

[29] Yifan Liu, Zengchang Qin, Tao Wan, and Zhenbo Luo. 2018. Auto-painter: Cartoon image generation from sketch by using conditional Wasserstein generative adversarial networks. *Neurocomputing* 311 (2018), 78–87.

[30] Michael Mathieu, Camille Couprie, and Yann LeCun. 2016. Deep multi-scale video prediction beyond mean square error. In *ICLR*.

[31] Mehdi Mirza and Simon Osindero. 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784* (2014).

[32] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. 2018. Spectral Normalization for Generative Adversarial Networks. In *ICLR*.

[33] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. 2018. Spectral normalization for generative adversarial networks. In *ICLR*.

[34] Yingge Qu, Tien-Tsin Wong, and Pheng-Ann Heng. 2006. Manga Colorization. *ACM Trans. Graph.* 25, 3 (July 2006), 1214–1220. https://doi.org/10.1145/1141911.1142017

[35] Alec Radford, Luke Metz, and Soumith Chintala. 2016. Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR*.

[36] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*. 2234–2242.

[37] Yaniv Taigman, Adam Polyak, and Lior Wolf. 2017. Unsupervised cross-domain image generation. In *ICLR*.

[38] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. 2018. Mocogan: Decomposing motion and content for video generation. In *CVPR*.

[39] Xiaolong Wang and Abhinav Gupta. 2016. Generative image modeling using style and structure adversarial networks. In *European Conference on Computer Vision*. Springer, 318–335.

[40] Wei Xiong, Wenhan Luo, Lin Ma, Wei Liu, and Jiebo Luo. 2018. Learning to generate time-lapse videos using multi-stage dynamic generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2364–2373.

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

[42] Zili Yi, Hao (Richard) Zhang, Ping Tan, and Minglun Gong. 2017. DualGAN: Unsupervised Dual Learning for Image-to-Image Translation.. In *ICCV*. 2868–2876.

[43] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris Metaxas. 2017. StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks. In *ICCV*.

[44] Lvmin Zhang, Yi Ji, and Xin Lin. 2017. Style transfer for anime sketches with enhanced residual U-net and auxiliary classifier gan. *arXiv preprint arXiv:1706.03319* (2017).

[45] Junbo Zhao, Michael Mathieu, and Yann LeCun. 2016. Energy-based generative adversarial network. *arXiv preprint arXiv:1609.03126* (2016).

[46] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros. 2016. Generative visual manipulation on the natural image manifold. In *European Conference on Computer Vision*. Springer, 597–613.

[47] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. 2017. Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), 2242–2251.

[48] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A Efros, Oliver Wang, and Eli Shechtman. 2017. Toward multimodal image-to-image translation. In *Advances in Neural Information Processing Systems*. 465–476.