Journal of Network Intelligence
Taiwan Ubiquitous Information
 ISSN 2414-8105 (Online)
Volume **7**, Number **1**, February **2022**

Icon-to-Icon Translation using Pixel-Level Weight Adversarial Networks

Yao-Ming Pan

School of Computer and Information Engineering
Xiamen University of Technology
Xiamen, 361024, China
pym@stu.xmut.edu.cn

Kai-Biao Lin*

School of Computer and Information Engineering
Xiamen University of Technology
Xiamen, 361024, China
Engineering Research Center of Big Data Application in Private Health Medicine
Fujian Provincial University
Putian, 351100, China
kblin@xmut.edu.cn

Chin-Ling Chen*

School of Information Engineering
Changchun Sci-Tech University
Changchun, 130600, China
Department of Computer Science and Information Engineering
Chaoyang University of Technology
Taichung, 41349, Taiwan
clc@mail.cyut.edu.tw

Corresponding Author: kblin@xmut.edu.cn and clc@mail.cyut.edu.tw

Received October 2021; revised December 2021

Abstract. *We investigate image-to-image translation as a solution to icon-to-icon translation problems. Icon-to-icon translation aims to translate the input icon to appear like the target domain's icons and simultaneously preserve the original icon's information (e.g., application logo, text and background color). We present a novel approach to translating icons to the version with the target style intelligently. By observing many icons with various styles, we propose a pixel-level weight matrix to calculating the weighted losses. These losses guide the generator to focus on the critical regions of the icon. People can clearly get the original icon's information from the proposed method's results, while the existing state-of-the-art image-to-image translation methods may fail to do this.*
Keywords: Icon-to-Icon translation, Image-to-Image translation, Icon synthesis

1. **Introduction.** Over the past decade, the Internet and the mobile Internet industry has come a long way. Merely providing new functions that consumers need is not enough to satisfy them nowadays. In recent years, there is an increasing need for personalization, which provides consumers with different products or services according to their preferences. The personalization of the user interface is very important to Internet users. It includes the personalization of icons, colors, backgrounds, etc. With the rapid

development of personalized icons, icons with various styles have sprung up to meet the diverse needs of users. As illustrated in Figure 1, various icons can be found in icon pack applications from mobile application stores.

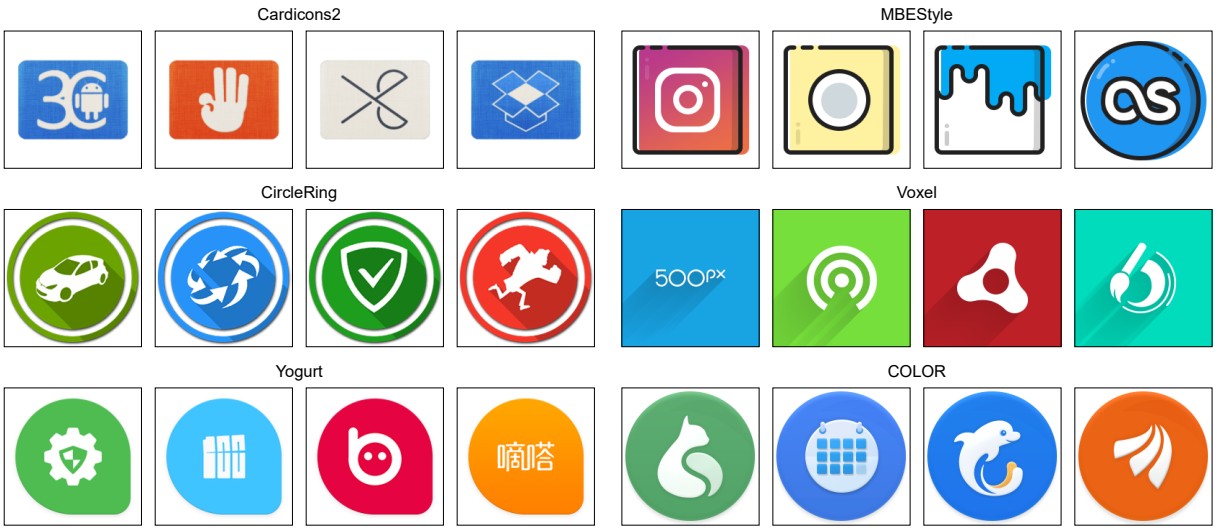

FIGURE 1. **Personalized icons.** Icons from datasets we collected from icon pack applications in mobile application stores. The title upon the icons refers to the icon pack that the icons come from.

Icons can quickly convey simple information to people, and leave a deeper impression on people than words. However, it takes a lot of time for designers to design icons. To make the icon with the style that some users prefer, the designer has to redesign the original icon so that the new icon has the target style. In this paper, we present a method that can help designers to redesign icons more effectively.

Unlike designing a new icon, the process of translating an icon to another style often preserves the original icon's information (e.g., application logo, text and background color). Generative adversarial networks (GANs) [1, 2] have achieved impressive results in image synthesis. Recent work on image-to-image translation [3, 4], can generate an image that indistinguishable from the target domain's images based on an input image and retain the original image's content. We wonder whether there will be such a program: We input some icons, select the icon style that we prefer, then click "translate", the input icons will be translated to the target style. We can use icons with the target style quicker, and designers won't do these lengthy and tedious tasks anymore. In this paper, we present a method that can learn to translate icons intelligently. We have achieved compelling results on icons with various styles. We focus on icon-to-icon translation. In this setting, the input domain is icons with a consistent style, and the target domain is icons with a different consistent style. The translated icons should have a similar style to the target domain's icons. People can clearly get the original icon's information (e.g., application logo, text and background color) from the translated icon. Logo and texts in a icon, help us get what the icon represents. Background color in a icon, is meaningful, paired with the logo and texts.

Without the proper data, it's difficult to apply machine learning to solve a problem. Icons in existing icon datasets (e.g., Large Logo Dataset (LLD) [5]) don't have a consistent style. Therefore, we collected some icon datasets from icon pack applications in mobile application stores. Each icon dataset contains hundreds or thousands of icons with a consistent style.

To the best of our knowledge, we can't find any study about icon-to-icon translation. Icon-to-icon translation is a specific image-to-image translation. In this setting, the translated icon keeps the original icon's information (e.g., application logo, text and background color). Therefore, we first evaluated existing state-of-the-art image-to-image translation methods on the collected icon datasets. As shown in Figure 2, previous models can't achieve satisfying results. The translated icons have some flaws that make the icons can't be used directly. We denote the icon on row i column j as $icon_{ij}$. For $icon_{12}$, $icon_{13}$, $icon_{14}$, $icon_{22}$, $icon_{24}$, $icon_{33}$, $icon_{34}$, $icon_{37}$, $icon_{38}$, $icon_{43}$, $icon_{44}$, $icon_{47}$ and $icon_{48}$, icons aren't visually similar to the target domain's icons. For $icon_{13}$, $icon_{15}$, $icon_{16}$, $icon_{17}$, $icon_{18}$, $icon_{23}$, $icon_{24}$, $icon_{25}$, $icon_{26}$, $icon_{27}$, $icon_{28}$, $icon_{32}$, $icon_{33}$, $icon_{34}$, $icon_{35}$, $icon_{36}$, $icon_{37}$, $icon_{38}$, $icon_{42}$, $icon_{44}$, $icon_{45}$, $icon_{46}$, $icon_{47}$, $icon_{48}$, $icon_{52}$, $icon_{53}$, $icon_{54}$, $icon_{55}$, $icon_{56}$, $icon_{57}$, $icon_{58}$, $icon_{62}$, $icon_{63}$, $icon_{64}$, $icon_{65}$, $icon_{66}$, $icon_{67}$ and $icon_{68}$, the original icons' information (e.g., application logo, text and background color) changed too much in the translated icons. To sum up, the existing methods may not get compelling results on icon-to-icon translation.

To address the problems above, we investigate the difference between icon-to-icon translation and image-to-image translation. It is enough that image-to-image translation's results look real, while icon-to-icon translation's results should keep the original icon's information (e.g., application logo, text and background color). After observing many icons with various styles, we found that icons generally consist of three elements: the center content, the background, and the white space. To preserve the original icon's information, the center content and the background color should be changed as little as possible, while the white space is irrelevant. However, most previous image-to-image translation studies simply pay the same attention to each image's pixel when calculating reconstruction losses. We may not simply think that the reconstruction losses of the white space are as important as the center region. Recent work U-GAT-IT [8] and NICE [9] pay different attention to image's pixels when calculating losses using an attention module based on CAM [12]. However, as shown in Figure 2, they could not achieve satisfying results. The CAM is a object classifier that trained to learn the weight of the k-th feature map for the input domain. The CAM attention module pays more attention to the pixels that crucial to judge objects in images. These pixels are different from the pixels of center content and background color, so U-GAT-IT [8] and NICE [9] could not achieve satisfying results. We introduce a pixel-level weight matrix to make the cycle consistency loss and identity loss from the icon's white space smaller. We conducted extensive experiments on the collected icon datasets. We show that with the help of a pixel-level weight matrix, the model (Pixel-level Weight Generative Adversarial Networks, PWGAN) can generate notably better results compared to previous methods.

In this paper, we propose PWGAN to intelligently translate icons into versions with target styles and save the information of the original icons. PWGAN can help designers to redesign icons more effectively.

2. **Related Work. Generative adversarial networks (GANs)** The recent years have witnessed notable advances of GANs [13–15] for image generation. The idea of training generator and discriminator alternately to optimize adversarial loss in opposite direction is the key to GANs' success. This process enforces the generated images to be indistinguishable from the target domain's images. We employ an adversarial loss to encourage the output to be visually similar to the target domain's icons.

Image-to-image translation aims to learn the mapping from an input image domain to a target image domain. A milestone in paired image-to-image translation is "pix2pix" [3, 16], which maps an image from the input domain to the output domain

FIGURE 2. **Results.** We compare the method (PWGAN) with existing state-of-the-art methods(CycleGAN [6], CUT [7] and its faster and lighter version FastCUT, U-GAT-IT [8], NICE [9], UNIT [10] and MUNIT [11]). We denote the icon on row i column j as $icon_{ij}$. We show successful cases above the dotted lines. From top to bottom, the icons are from Cardicons2→MBEStyle task, CircleRing→Voxel task, and Yogurt→COLOR task. The final rows show typical failure cases.

using a conditional GAN. Paired training data are needed in "pix2pix" to calculate the reconstruction loss between the generated image and the corresponding image. However, paired training data can be expensive to obtain. Recent works attempted to learn image translation in an unpaired setting. In such cases, cycle-consistency [6, 17, 18] is a widely used approach to preserve the "content" of the input image, which calculates reconstruction loss based on a learned inverse mapping from the target to the input. In addition, UNIT [10] and MUNIT [11] propose to learn a shared "content" latent space instead. Recently, U-GAT-IT [8] incorporates an attention module and a learnable normalization into the translation model. Moreover, NICE-GAN [9] introduces a novel translation architecture, which reuses discriminators for encoding the images of the target domain. A recent approach [7] applies contrastive representation learning to enable one-sided unpaired translation. The method proposed builds on CycleGAN [6], which uses cycle-consistency

to enforce the correspondence between the input and output. Unlike the prior works above, we introduce a pixel-level weight matrix to calculate weighted cycle consistency loss and weighted identity loss in the model's objective.

Icon synthesis First attempt at icon synthesis was made in 2017 in the field of logo synthesis. Sage et al. [5] built a dataset of 600k+ logos crawled from the web. They used synthetic labels obtained through clustering to disentangle and stabilize GAN training on multi-modal datasets like the LLD. High diversity of plausible logos can be generated using their method. Recent work [19] trains a dual conditional generative adversarial network to colorize contour images using a referenced icon, and the generated icons and the referenced icon are similar in color style. An interesting work [20] translates preprocessed photos to black-and-white icons and LLD-style [5] icons using image-to-image translation models (e.g., CycleGAN [6]). Unlike the above works, our work focuses on icon-to-icon translation. The biggest difference between icon-to-icon translation and previous icon synthesis works is: The input icons have a consistent style and the target domain's icons have a consistent style on icon-to-icon translation.

3. **Methods.** Our goal is to translate the icons from input domain X to appear like the icons from output domain Y. We are given training samples $\{x_i\}_{i=1}^{N}$ where $x_i \in X$ and $\{y_j\}_{j=1}^{M}$ where $y_j \in Y$. For simplicity, we will omit the subscript i and j below. We denote the data distribution as $x \sim p(x)$ and $y \sim p(y)$. We adopt cycle-consistency to enforce the correspondence between the input and the output by learning two mappings $G : X \to Y$ and $F : Y \to X$. Accordingly, the framework includes two discriminators D_X and D_Y, where D_X distinguishes icons $\{x\}$ from translated icons $\{F(y)\}$; Likewise, D_Y discriminates between $\{y\}$ and $\{G(x)\}$. We introduce a pixel-level weight matrix to calculate weighted cycle consistency loss and weighted identity loss in the objective. The new losses above guide the generator to focus on the critical regions of the icon.

3.1. **Adversarial loss.** We employ adversarial losses [1] on both G and F. For the mapping $G : X \to Y$ and its discriminator D_Y, we express the objective as:

$$\begin{aligned}\mathcal{L}_{\mathrm{GAN}}(G, D_Y, X, Y) = {} & \mathbb{E}_{y\sim p(y)}[\log D_Y(y)] \\ & + \mathbb{E}_{x\sim p(x)}[\log(1 - D_Y(G(x)))]. \end{aligned} \tag{1}$$

G is encouraged to translate icons x to icons $G(x)$ that appear like the icons from domain Y, while D_Y distinguishes generated samples $G(x)$ from real samples y. G tries to minimize this loss, while D aims to maximize it, i.e., $\min_G \max_{D_Y} \mathcal{L}_{\mathrm{GAN}}(G, D_Y, X, Y)$. We apply a similar adversarial loss to the mapping $F : Y \to X$ and its discriminator D_X as well: i.e., $\min_F \max_{D_X} \mathcal{L}_{\mathrm{GAN}}(F, D_X, Y, X)$.

3.2. **Pixel-level weight matrix.** We want the generator to pay different attention to different regions of the icon. So, we propose a pixel-level weight matrix to calculate the weighted version of cycle consistency loss and identity loss. As shown in Figure 3, Figure 4 and Figure 5, we apply three types of center pixel-level weight matrices in the experiments. We use a pixel-level weight matrix to calculate the weighted loss between icon x and icon y as follows:

$$\ell_{\mathrm{weight}}(x, y) = \mathrm{mean}(|x - y| \circ M_{\mathrm{weight}}). \tag{2}$$

We define a term, center rate, which is the ratio of the icon's center content size to the icon size. The prefix "center" refers to the center version of the matrices we proposed. We adopt 192×192 (the sizes of the icons collected are all 192×192) pixel-level weight

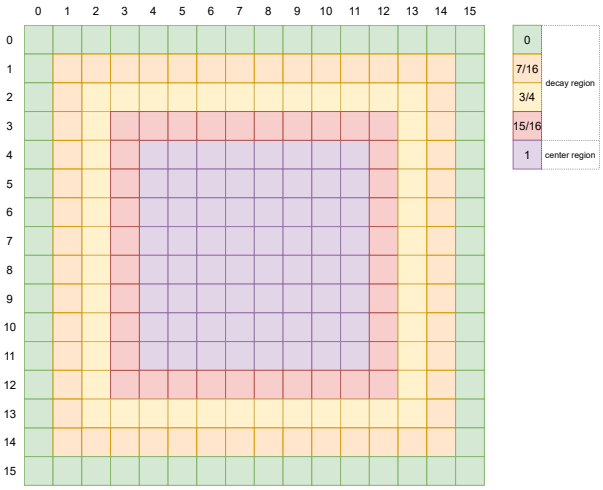

FIGURE 3. **Square center pixel-level weight matrix.** 16×16 square center pixel-level weight matrix with a 0.5 center rate.

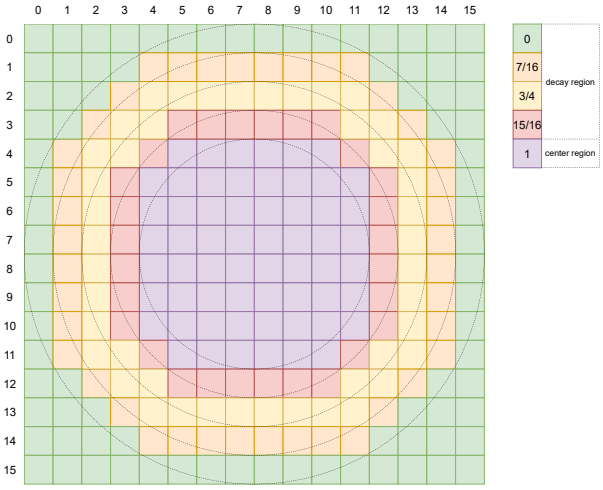

FIGURE 4. **Circle center pixel-level weight matrix.** 16×16 circle center pixel-level weight matrix with a 0.5 center rate.

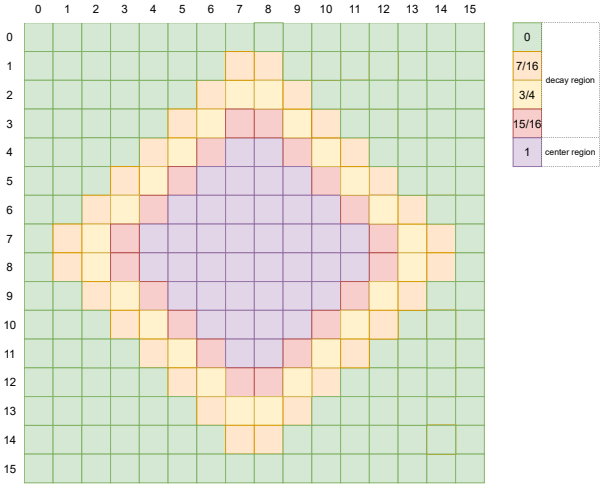

FIGURE 5. **Diamond center pixel-level weight matrix.** 16×16 diamond center pixel-level weight matrix with a 0.5 center rate.

matrices in the experiments. As illustrated in Figure 3, Figure 4 and Figure 5, the matrix consists of two regions: the center region and the decay region. The value of the center region is 1, as we think that each pixel in the center region has the same importance. The value of the decay region will decay from 1 to 0 quadratically since icons are two-dimensional. The further the pixel is away from the center region, the smaller the value is.

3.3. **Weighted cycle consistency loss.** To preserve the input icon's information (e.g., application logo, text and background color), we adopt cycle-consistency [6, 17, 18] constraint on the generator. For each icon x from domain X, the translation cycle should have enough ability to translate x back to the input icon, i.e., $x \to G(x) \to F(G(x)) \approx x$. Likewise, for each icon y from domain Y, $y \to F(y) \to G(F(y)) \approx y$ should be satisfied. We introduce a pixel-level weight matrix to calculate weighted cycle consistency loss, which guides the generator to focus on the critical regions of the icon. We express the loss as:

$$\begin{aligned}\mathcal{L}_{\text{weighted cyc}}(G, F) = {} & \mathbb{E}_{x\sim p_(x)}[\ell_{\text{weight}}(x, F(G(x)))] \\ & + \mathbb{E}_{y\sim p_(y)}[\ell_{\text{weight}}(G(F(y)), y)].\end{aligned} \tag{3}$$

3.4. **Weighted identity loss.** To enforce the generator to retain color composition between the input and the output, we adopt identity loss [21]. Identity loss encourages the generator to be an identity mapping when real icons are provided as the input to the generator. We apply the weighted identity loss, which is more reasonable on icon-to-icon translation. We express the objective as:

$$\begin{aligned}\mathcal{L}_{\text{weighted id}}(G, F) = {} & \mathbb{E}_{y\sim p(y)}[\ell_{\text{weight}}(G(y), y)] \\ & + \mathbb{E}_{x\sim p(x)}[\ell_{\text{weight}}(x, F(x))].\end{aligned} \tag{4}$$

3.5. **Full objective.** The full objective is:

$$\begin{aligned}\mathcal{L}(G, F, D_X, D_Y) = {} & \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) \\ & + \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) \\ & + \lambda_c \mathcal{L}_{\text{weighted cyc}}(G, F) \\ & + \lambda_i \mathcal{L}_{\text{weighted id}}(G, F),\end{aligned} \tag{5}$$

where λ_c and λ_i control the importance of cycle consistency loss and identity loss. Our goal is to solve:

$$G^*, F^* = \arg\min_{G,F} \max_{D_X, D_Y} \mathcal{L}(G, F, D_X, D_Y). \tag{6}$$

We optimize this full objective in opposite direction alternately to train generator and discriminator alternately. We train a better generator, then use it to train a better discriminator. After we get a better discriminator, we use it to train a better generator. This loop is a typical GAN training process.

4. **Experiments.**

4.1. **Implementation details.** To the best of our knowledge, we can't find any existing icon-to-icon translation method, so we compare the method (PWGAN) with various state-of-the-art image-to-image translation methods on Cardicons2→MBEStyle, CircleRing→Voxel and Yogurt→COLOR tasks. Some typical results are shown in Figure 2. All the methods are implemented using the author's code. The icons' sizes in the datasets are 192×192, so the input icons' sizes on the experiments are 192×192. For CycleGAN [6], CUT [7] and its faster and lighter version FastCUT, we use a learning rate of 0.0002 for both generator and discriminator during training in the experiments. The batch sizes on dataset Cardicons2→MBEStyle, CircleRing→Voxel and Yogurt→COLOR are 8. For U-GAT-IT [8], NICE [9], UNIT [10] and MUNIT [11], we use a learning rate of 0.0001 for both generator and discriminator during training in the experiments. The batch size for U-GAT-IT [8], NICE [9], UNIT [10] and MUNIT [11] are 1, 1, 8, and 8, respectively.

In the experiments of PWGAN, we follow the setting of CycleGAN [6], except that the cycle consistency loss and the identity loss are replaced with the weighted version losses. We use a learning rate of 0.0002 for both generator and discriminator during training in the experiments, and batch sizes on the datasets are 8. As shown in Table 4, Table 5 and Table 6, we trained PWGAN with different matrices and center rates. We found PWGAN with square matrix and 0.618 center rate gets better results than other cases, the results of PWGAN are from this case in Figure 2, Table 3 and Table 7.

4.2. **Baselines. CycleGAN [6]** This method learns generator $G : X \to Y$ and $F : Y \to X$ using adversarial losses. It uses $\|x - F(G(x))\|_1$ as cycle-consistency loss to encourage $x \to G(x) \to F(G(x)) \approx x$.

CUT [7] Taesung et al. [7] uses an adversarial loss to train a translation from X to Y. They apply contrastive representation learning to enforce content correspondence between an input image and its translated image. FastCUT can be thought as a faster and lighter version of CycleGAN, it's simplified from CUT.

U-GAT-IT [8] Unsupervised image-to-image translation model incorporates an attention module and a learnable normalization. This attention module is based on CAM [12].

NICE [9] Using the framework of U-GAT-IT [8], Runfa et al. [9] reuse discriminators for encoding the images of the target domain to regularize the original overfitting model.

UNIT [10] Like CycleGAN [6], Liu et al. [10] share a content latent space between generator $G : X \to Y$ and $F : Y \to X$ to encourage content correspondence between an input image and its translated image.

MUNIT [11] Using the framework of UNIT [10], this method extends unsupervised image-to-image translation to the multimodal case.

4.3. **Dataset.** We evaluated the performance of each method on the datasets collected from icon pack applications. There are 2695 icons in the Cardicons2 [22] icon set, 709 icons in the MBEStyle [23] icon set, 5827 icons in the CircleRing [24] icon set, 4884 icons in the Voxel [25] icon set, 1170 icons in the Yogurt [26] icon set, 1510 icons in the COLOR [27] icon set. We use 80% of the data for training, 10% for validating, and 10% for testing.

4.4. **Quantitative evaluation.** The translated icons should have a similar style to the target domain's icons. There is a widely-used Fréchet Inception Distance (FID) [28] metric that might measure the completion of this goal. FID estimates the distribution of real and generated images in the feature space of the Inception network [29] and computes the Fréchet distance between them. FIDs of the methods are shown in Table 1. Lower is better.

People can clearly get the original icon's information (e.g., application logo, text and background color) from the translated icon. To the best of our knowledge, we can't

TABLE 1. **FIDs for the methods.** Fréchet Inception Distance (FID) for PWGAN and existing methods. C→M refers to Cardicons2→MBEStyle task, C→V refers to CircleRing→Voxel task, Y→C refers to Yogurt→COLOR task.

Method	C→M	C→V	Y→C
PWGAN	112.29	73.28	113.01
CycleGAN	150.47	52.65	122.17
CUT	99.32	150.48	137.49
FastCUT	150.11	163.24	158.19
U-GAT-IT	75.23	106.51	107.13
NICE	111.27	163.64	107.21
UNIT	126.91	126.74	189.28
MUNIT	94.26	87.76	155.52
average value	114.98	115.54	136.25

find any quantitative indicator that can measure this goal. Therefore, we design a circle weighted quadratic loss to estimate the completion of this goal. We express the loss as:

$$\ell_{\text{circle}}(x, y) = \text{mean}((x - y) \circ (x - y) \circ M_{\text{circle}}). \tag{7}$$

x denotes a input icon, y denotes its translated icon, M_{circle} is a 192×192 circle center pixel-level weight matrix with 0.5 center rate. We use circle rather than square or diamond matrix with 0.5 center rate, because logo, text in icons often locate in the center region of this matrix. Square and diamond matrices usually cover more background color area in icons than circle matrix. This makes the background color difference between x and y has a improper big influence than logo and text difference. Therefore, we use circle matrix to calculate the loss. We use quadratic loss to penalize big change between x and y. The methods' circle weighted quadratic losses are shown in Table 2. Lower is better.

TABLE 2. **Circle weighted quadratic losses for the methods.** Circle weighted quadratic losses for PWGAN and existing methods. C→M refers to Cardicons2→MBEStyle task, C→V refers to CircleRing→Voxel task, Y→C refers to Yogurt→COLOR task.

Method	C→M	C→V	Y→C
PWGAN	3140.81	2521.62	1128.01
CycleGAN	2934.95	6807.01	1308.38
CUT	4269.31	4470.34	3262.54
FastCUT	4886.39	4472.35	5073.87
U-GAT-IT	5003.28	7295.34	1476.40
NICE	3520.67	6949.09	1386.36
UNIT	4048.08	5341.62	2474.04
MUNIT	6360.17	5670.48	5104.40
average value	4270.46	5440.98	2651.75

To comprehensively evaluate the methods, we standardize their FIDs and circle weighted quadratic losses then sum them up to calculate icon-to-icon translation indices. We define the index as:

$$i_{\text{icon-to-icon}} = \frac{\text{FID} - \overline{\text{FID}}}{\overline{\text{FID}}} + \frac{\ell_{\text{circle}} - \overline{\ell_{\text{circle}}}}{\overline{\ell_{\text{circle}}}}. \tag{8}$$

$\overline{\text{FID}}$ denotes average FIDs in Table 1, $\overline{\ell_{\text{circle}}}$ denotes average circle weighted quadratic losses in Table 2. Every task has its corresponding $\overline{\text{FID}}$ and $\overline{\ell_{\text{circle}}}$. Icon-to-icon translation indices are shown in Table 3. Lower is better. PWGAN has better results on all the tasks. On quantitative evaluation, PWGAN with pixel-level weight matrix performs better than the existing methods.

TABLE 3. **Quantitative results.** Icon-to-icon translation indices for PWGAN and existing methods. C→M refers to Cardicons2→MBEStyle task, C→V refers to CircleRing→Voxel task, Y→C refers to Yogurt→COLOR task.

Method	C→M	C→V	Y→C
PWGAN	**-0.29**	**-0.90**	**-0.75**
CycleGAN	0.00	-0.29	-0.61
CUT	-0.14	0.12	0.24
FastCUT	0.45	0.23	1.07
U-GAT-IT	-0.17	0.26	-0.66
NICE	-0.21	0.69	-0.69
UNIT	0.05	0.08	0.32
MUNIT	0.31	-0.20	1.07

As illustrated in Table 4, Table 5 and Table 6, we trained PWGAN with different matrices and center rates. The models with circle and square matrices consistently perform better than the existing methods, while the models with diamond matrices sometimes don't. Circle and square matrices are more reasonable than diamond matrices on icon-to-icon translation. By manually comparing the above cases' results, we found PWGAN with square matrix and 0.618 center rate gets better results than other cases on all the tasks. However, the icon-to-icon translation indices of PWGAN with square matrix and 0.618 center rate aren't better than some PWGAN models' indices. The designs of circle weighted quadratic loss and icon-to-icon translation index have possibility of improvement. Icon-to-icon translation index might measure icon-to-icon translation's quality in a way, but can't measure it perfectly.

4.5. **Qualitative evaluation.** To evaluate the quality of translated icons, we carry out a user study. We recruit 167 volunteers for the user study. The participants include 67 elementary school students between the ages of 11 and 13, 14 students between the ages of 18 and 21, 37 workers and students between the ages of 22 and 31, 1 teacher between the ages of 32 and 45, and 48 middle-aged people between the ages of 46 and 58. For each translation task, we provide 10 icons from the target domain for the participants. We give the original icon first, then the participants follow the rule (the translated icon appears like the target domain's icons and simultaneously preserves the original icon's information (e.g., application logo, text and background color)) to select the best icon among the icons generated from different methods including the proposed method. As illustrated in Figure 6 and Table 7, most participants prefer the icons translated by PWGAN. As shown in Figure 2, the icons translated by PWGAN appear like the target domain's icons and simultaneously preserve the original icon's information (e.g., application logo, text

TABLE 4. **Quantitative results for PWGAN on Cardicons2→MBEStyle task.** Icon-to-icon translation indices for PWGAN with different matrices and center rates on Cardicons2→MBEStyle task. Lower is better.

center rate	circle	diamond	square
0.1	-0.26	-0.10	-0.22
0.2	-0.51	-0.38	-0.21
0.3	-0.22	-0.26	-0.24
0.4	-0.27	0.44	-0.40
0.5	-0.37	-0.28	-0.26
0.6	-0.33	-0.22	-0.48
0.618	-0.31	-0.12	-0.29
0.7	-0.24	-0.33	-0.23
0.8	-0.37	-0.13	-0.41
0.9	-0.51	-0.03	-0.40

TABLE 5. **Quantitative results for PWGAN on CircleRing→Voxel task.** Icon-to-icon translation indices for PWGAN with different matrices and center rates on CircleRing→Voxel task. Lower is better.

center rate	circle	diamond	square
0.1	-0.33	-0.10	-1.09
0.2	-0.46	0.57	-1.01
0.3	-0.30	0.18	-0.67
0.4	-0.48	0.18	-1.20
0.5	-0.31	-0.10	-0.32
0.6	-1.17	-0.40	-1.14
0.618	-1.18	-0.18	-0.90
0.7	-0.44	-0.08	-1.16
0.8	-0.62	-0.35	-0.45
0.9	-0.83	0.30	-1.24

TABLE 6. **Quantitative results for PWGAN on Yogurt→COLOR task.** Icon-to-icon translation indices for PWGAN with different matrices and center rates on Yogurt→COLOR task. Lower is better.

center rate	circle	diamond	square
0.1	-0.73	-0.56	-0.81
0.2	-0.70	-0.56	-0.72
0.3	-0.72	-0.76	-0.76
0.4	-0.79	-0.71	-0.75
0.5	-0.71	-0.51	-0.71
0.6	-0.82	-0.35	-0.82
0.618	-0.77	-0.36	-0.75
0.7	-0.76	-0.87	-0.70
0.8	-0.77	-0.62	-0.69
0.9	-0.78	-0.77	-0.81

and background color). Through the results of qualitative evaluation, we show the crucial role that the pixel-level weight matrix plays in achieving compelling results.

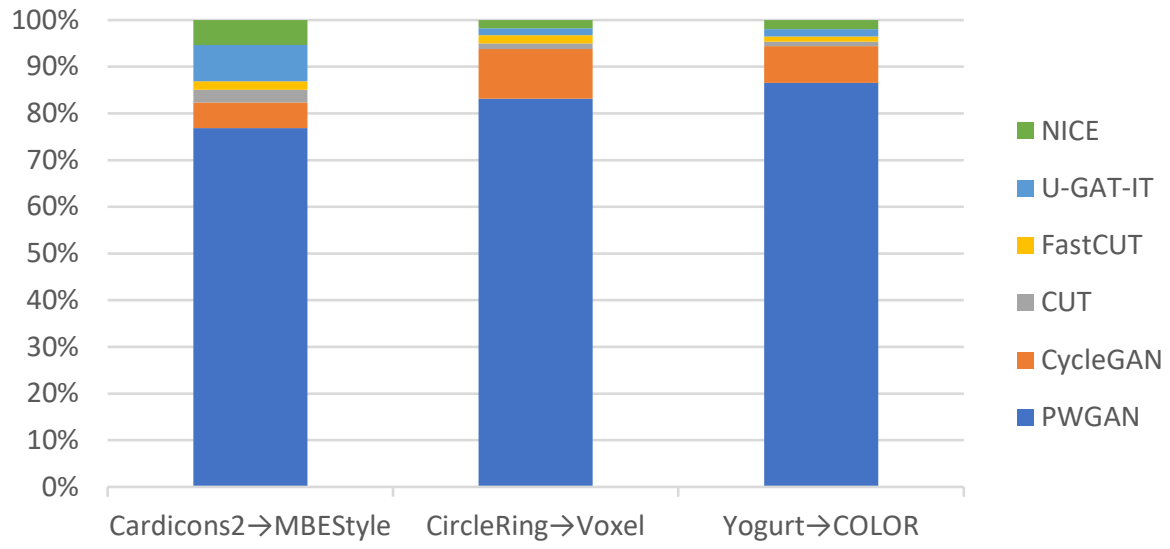

FIGURE 6. **Qualitative results graph.** The length of the bar indicates the percentage of preference on the corresponding translation task. None of the participants prefer the icons generated by UNIT and MUNIT on the tasks.

TABLE 7. **Qualitative results table.** The percentage of preference on the corresponding translation task. C→M refers to Cardicons2→MBEStyle task, C→V refers to CircleRing→Voxel task, Y→C refers to Yogurt→COLOR task. None of the participants prefer the icons generated by UNIT and MUNIT on the tasks.

Method	C→M	C→V	Y→C
PWGAN	**76.83**	**83.17**	**86.53**
CycleGAN	5.51	10.60	7.84
CUT	2.69	1.20	1.02
FastCUT	1.86	1.80	1.14
U-GAT-IT	7.78	1.44	1.56
NICE	5.33	1.80	1.92

5. **Limitations and Discussion.** Although PWGAN can achieve compelling results in many tasks, the results are far from uniformly practical. When the color distribution of icons is complex, PWGAN may gets unsatisfying results. Figure 2 shows several typical failure cases. For $icon_{71}$, we fail to extend color to the right edge , when the input icon's color distribution is complex. For $icon_{81}$, PWGAN is unable to recognize the accurate

area that should become higher than the background, when the icon's background consists of areas with various colors.

In this paper, we introduce icon-to-icon translation index to measure icon-to-icon translation's quality. However, it's far from perfect. A better quantitative index for icon-to-icon translation's quality, will become an important part in the future works of icon-to-icon translation. As PWGAN use a pixel-level weight matrix, by applying another task-specific weight matrix, PWGAN may be helpful for other image translations (e.g., face translation) where key elements often appear in a certain region.

Acknowledgment. This research was funded by the Science Foundation of Fujian Province (No. 2021J011188), the Xiamen Science and Technology Planning Project (No.3502Z20206073), the Open Fund of Engineering Research Center of Big Data Application in Private Health Medicine, Fujian Provincial University (KF2020003), and the XMUT Scientific Research Project (YKJCX2019113).

REFERENCES

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, pp. 2672–2680, 2014.

[2] L. Kang, R.-S. Chen, N. Xiong, Y.-C. Chen, Y.-X. Hu, and C.-M. Chen, "Selecting hyper-parameters of gaussian process regression based on non-inertial particle swarm optimization in internet of things," *IEEE Access*, vol. 7, pp. 59504–59513, 2019.

[3] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5967–5976, 2017.

[4] E. K. Wang, X. Zhang, F. Wang, T.-Y. Wu, and C.-M. Chen, "Multilayer dense attention model for image caption," *IEEE Access*, vol. 7, pp. 66358–66368, 2019.

[5] A. Sage, E. Agustsson, R. Timofte, and L. V. Gool, "Logo synthesis and manipulation with clustered generative adversarial networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5879–5888, 2018.

[6] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *IEEE International Conference on Computer Vision*, pp. 2242–2251, 2017.

[7] T. Park, A. A. Efros, R. Zhang, and J.-Y. Zhu, "Contrastive learning for unpaired image-to-image translation," in *European Conference on Computer Vision*, pp. 319–345, 2020.

[8] J. Kim, M. Kim, H. Kang, and K. Lee, "U-GAT-IT: unsupervised generative attentional networks with adaptive layer-instance normalization for image-to-image translation," in *International Conference on Learning Representations*, pp. 1–19, 2020.

[9] R. Chen, W. Huang, B. Huang, F. Sun, and B. Fang, "Reusing discriminators for encoding: Towards unsupervised image-to-image translation," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8165–8174, 2020.

[10] M. Liu, T. Breuel, and J. Kautz, "Unsupervised image-to-image translation networks," in *Advances in Neural Information Processing Systems*, pp. 700–708, 2017.

[11] X. Huang, M. Liu, S. J. Belongie, and J. Kautz, "Multimodal unsupervised image-to-image translation," in *European Conference on Computer Vision*, pp. 179–196, 2018.

[12] B. Zhou, A. Khosla, À. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2921–2929, 2016.

[13] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *International Conference on Learning Representations*, pp. 1–16, 2016.

[14] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International Conference on Machine Learning*, pp. 214–223, 2017.

[15] C. Chen, C. Lin, Y. Liu, J. Liao, M. Yeh, Y. Chung, and C. Hsu, "Applying image processing technology to automatically detect and adjust paper benchmark for printing machine," *Journal of Information Hiding and Multimedia Signal Processing*, vol. 12, no. 2, pp. 56–64, 2021.
[16] K. Wang, F. Li, C.-M. Chen, M. M. Hassan, J. Long, and N. Kumar, "Interpreting adversarial examples and robustness for deep learning-based auto-driving systems," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–10, 2021.
[17] Z. Yi, H. R. Zhang, P. Tan, and M. Gong, "Dualgan: Unsupervised dual learning for image-to-image translation," in *IEEE International Conference on Computer Vision*, pp. 2868–2876, 2017.
[18] T. Kim, M. Cha, H. Kim, J. K. Lee, and J. Kim, "Learning to discover cross-domain relations with generative adversarial networks," in *International Conference on Machine Learning*, pp. 1857–1865, 2017.
[19] T.-H. Sun, C.-H. Lai, S.-K. Wong, and Y.-S. Wang, "Adversarial colorization of icons based on contour and color conditions," in *ACM International Conference on Multimedia*, pp. 683–691, 2019.
[20] T. Karamatsu, G. Benitez-Garcia, K. Yanai, and S. Uchida, "Iconify: Converting photographs into icons," in *Proceedings of the 2020 Joint Workshop on Multimedia Artworks Analysis and Attractiveness Computing in Multimedia*, pp. 7–12, 2020.
[21] Y. Taigman, A. Polyak, and L. Wolf, "Unsupervised cross-domain image generation," in *International Conference on Learning Representations*, pp. 1–15, 2017.
[22] CookDev, "Cardicons2." `https://www.coolapk.com/apk/cookdev.iconpack.ii`. Last accessed June 29, 2020.
[23] Meolunr, "Mbe style." `https://www.coolapk.com/apk/me.iacn.mbestyle`. Last accessed June 29, 2020.
[24] GomoTheGom, "Circle ring - free icon pack." `https://play.google.com/store/apps/details?id=com.panotogomo.iconpack.circle.ring`. Last accessed June 29, 2020.
[25] B. Norgelas-Dzimidas, "Voxel – flat style icon pack." `https://play.google.com/store/apps/details?id=com.benx9.palmtree`. Last accessed June 29, 2020.
[26] modetime, "Yogurt." `https://www.coolapk.com/apk/com.modetime.yogurt`. Last accessed June 29, 2020.
[27] LittleGA, "Color." `https://www.coolapk.com/apk/com.ga.iconpack.color`. Last accessed June 29, 2020.
[28] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," in *Advances in Neural Information Processing Systems*, pp. 6626–6637, 2017.
[29] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826, 2016.