Computing and Informatics, Vol. 43, 2024, 482–504, doi: 10.31577/cai_2024_2_482

RADICAL CONSTRAINT-BASED GENERATIVE ADVERSARIAL NETWORK FOR HANDWRITTEN CHINESE CHARACTER GENERATION

Xi-Ling Ye, Hong-Bo Zhang*

Department of Computer Science and Technology
Huaqiao University
Xiamen, 361021, China
e-mail: `20013083033@stu.hqu.edu.cn, zhanghongbo@hqu.edu.cn`

Li-Jie Yang

Fujian Key Laboratory of Big Data Intelligence and Security
Huaqiao University
Xiamen, 361021, China
e-mail: `yanglijie@hqu.edu.cn`

Ji-Xiang Du

Department of Computer Science and Technology
Huaqiao University
Xiamen, 361021, China
e-mail: `jxdu@hqu.edu.cn`

Abstract. Generative adversarial networks (GANs) have been used as a solution to handwritten Chinese character automatic generation (HCCAG) in recent years. However, most existing GAN-based methods adopt a pixel-based strategy, which ignores the radical structure of Chinese characters. To achieve better HCCAG, a radical constraint-based GAN (RC-GAN) is proposed in this work. In the proposed method, a gated recurrent unit (GRU)-based radical learning network is designed

* Corresponding author

to calculate the radical components among Chinese characters, and radical consistent loss is applied to train this module. Finally, the radical learning module is fused with a cyclic structure GAN to improve the performance of Chinese character generation. The experimental results show that compared with the existing GAN, the proposed method has better performance. Specifically, the proposed method can reduce the stroke error in the generated Chinese character images.

Keywords: Handwritten Chinese character generation, generative adversarial network, radical constraint, cyclic structure

Mathematics Subject Classification 2010: 68U10

1 INTRODUCTION

Chinese calligraphy is not only an expression of Chinese culture but also one of the artistic and cultural carriers of China's long history. For a long time, Chinese calligraphy has been the object of imitation and study by many researchers. Due to the large number of Chinese characters, the complexity of stroke structures and the diversity of their styles, it often takes a great amount of time and effort to imitate them well before one may achieve the desired visual aesthetic effect. The topic of this work is the automatic generation of handwritten Chinese characters, which are widely used in many applications, such as Chinese character art creation, calligraphy education and personalized character library creation. How to automatically generate Chinese characters with strong artistic effects through a computer has received extensive attention from numerous researchers, and many methods have been proposed in recent years.

Many early works on handwritten Chinese character automatic generation (HCCAG) relied on a hierarchical representation of simple strokes decomposing Chinese character images (CCIs) into strokes and then combining the strokes to mimic the writing style of Chinese characters. These approaches focused only on the local representation of Chinese characters and not on their overall stylistic features. Therefore, researchers have refined the method with various steps in an effort to optimize the overall results. However, such improvement continues to be non-significant.

With the rapid development of deep learning algorithms in computer vision and computer graphics, researchers have begun to turn to more structurally complex deep neural networks (DNNs) to solve HCCAG, such as DNN-based methods and semi-amortized variational autoencoders. Specifically, due to the development of generative adversarial networks (GANs) [1], researchers have started to use GANs to achieve image-to-image style transfer. Referring to image-to-image translation methods [2] between different domains, the HCCAG task is also regarded as the image-to-image style translation problem. In these related works, different font styles, such as DFKai-SB script[1], running script, DFKai-SB and Pen-Kai scripts[2], SIM-Kai script[3] and Lanting script[4], are regarded as different data domains. Zi2zi [3] was the first work to use GAN to generate Chinese characters but required pairwise data as the input. DCFont [4] and pyramid embedded GAN (PEGAN) [5] were the improved versions of the zi2zi method. Handwritten CycleGAN [6] was designed to solve the need for pairwise input in the zi2zi algorithm, while Calli-GAN [7] was proposed to generate Chinese calligraphy using strokes as the input feature.

These GAN-based methods can be summarized as pixel-based methods, which use pixel-level loss and adversarial loss to train the HCCAG model. However, Chinese characters are composed of a few fundamental structural components called radicals [8]. Therefore, extracting radical information in the HCCAG is crucial. Traditional image-to-image style transfer focuses on the visual effect of the whole image with a complex scene so that even if there is a small missing or subtle flaw in the generated image, we are unlikely to notice this. However, in the case of HCCAG, the generated CCI is a binary image. Small stroke errors have a greater influence on the visual effect of HCCAG. As shown in Figure 1, there are some stroke errors, including missing strokes, broken strokes, extra strokes, and incomplete strokes, in the image generated by zi2zi [3], Handwritten-CycleGAN [6] and no independent component for encoding GAN (NICE-GAN) [9].

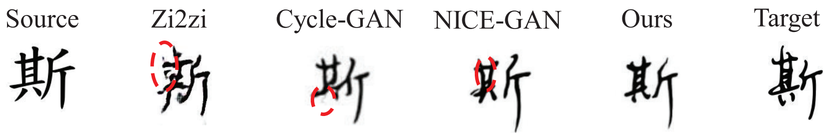

Figure 1. Examples of HCCAG with different methods. The red circle indicates the stroke error.

To address the problem of stroke errors in such methods and improve the visual effect of HCCAG, inspired by the radical analysis work of Chinese characters [8, 10], in this work, a radical constraint module is proposed to fuse with the existing GAN model, and a new radical constraint-based GAN (RC-GAN) is proposed for the HCCAG task. The key idea behind the proposed method is to constrain the radical sequence of the Chinese character to be correctly decoded from the generated CCI to avoid stroke errors. The overall framework of the proposed method is shown in Figure 2. In the proposed method, a GAN with a cycle structure is adopted, and NICE-GAN, which is an improved CycleGAN model, is used as the baseline model. We assume that the standard radicals of the same Chinese character in different domains are consistent. To implement radical constraints, a gated recurrent

[1] This represents traditional Chinese characters with Kai script.
[2] These represent handwritten Kai scripts with pen.
[3] This represents simplified Chinese characters with Kai script.
[4] This represents the characters from "The Orchid Thief" written by Wang Xizhi.

unit (GRU) model is designed to learn the radical structure sequence of the CCI. A radical consistent loss between the predicted radical sequence and the standard radical sequence is defined to train the radical learning model. Finally, the radical consistent loss is merged with the reconstruction loss, cycle loss and adversarial loss to train the proposed model.

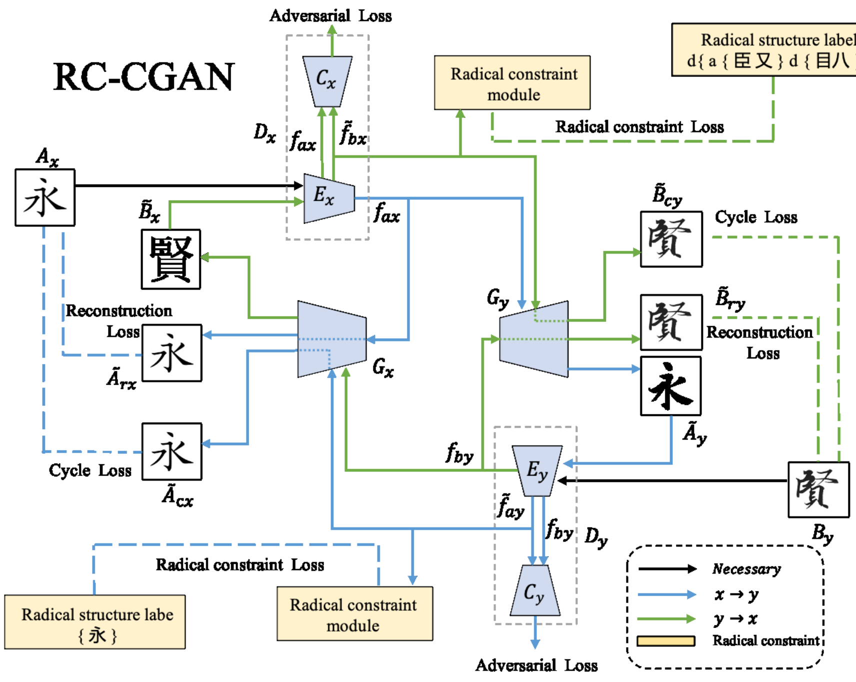

Figure 2. Illustration of the proposed RC-GAN. Blue lines indicate the conversion from the X to Y domain. Green lines indicate the transformation from Y to X domain. The network framework reuses the encoder part of the discriminator to extract image features. In addition, the method adds a radical constraint module to the GAN to generate characters with enhanced visual effects.

- We propose an RC-GAN to solve the stroke error problem of HCCAG and improve the visual effect of the generated handwritten Chinese character images.
- We apply two GRU decoders and radical consistent loss to learn the radical sequence from two domains.
- The experimental results show that the proposed method achieves state-of-the-art performance. Compared with the baseline model, the proposed model can significantly reduce the number of stroke errors in the generated handwritten Chinese character image.

The remainder of this paper is organized as follows. Section 2 introduces the related works, Section 3 describes the proposed method, Section 4 presents and discusses the experimental results, and Section 5 concludes the paper.

2 RELATED WORKS

2.1 Generative Adversarial Networks

In 2014, GAN was proposed for image-to-image transfer by Goodfellow et al. [1] and contained a generator and a discriminator. The generator was used to generate the target image, and the discriminator was used to distinguish the generated image from the real image. This proposed network framework has produced a major technological breakthrough in the field of deep learning and has started to be widely studied and applied by researchers. However, the original GAN was not mature. Problems such as gradient explosion and pattern collapse have limited its development. Therefore, researchers have proposed many improved versions of GANs to address these problems. The concept of conditional GAN [11] added new constraints to the original GAN to generate the network for the target sample, but its model training was unstable. Wasserstein GAN [12] used the Wasserstein distance instead of the Jensen-Shannon distance in the original GAN to measure the distance between true and generated samples to solve the problem of disappearing and exploding gradients.

Traditional GANs transform the images between two different domains and require pairwise images with the same content in both domains as training data, but such datasets are difficult to obtain. Therefore, CycleGAN [13] were proposed to construct a learning strategy for mismatched images. The framework of these methods usually contains two discriminators and two generators and forms a cyclic network, enabling the transitions between the two domains to be generated mutually and thus effectively solving the problem that the datasets need to be pairwise. In addition, an unsupervised GAN with adaptive layer-instance normalization (AdaLIN) for image-to-image translation added an attention mechanism module based on CycleGAN. NICE-GAN [9] proposed an improved method by reusing the discriminator, using its first part as the encoder to extract image features and enhance the performance of the network structure.

2.2 Chinese Character Generation

Early works on Chinese character generation involved researchers layering Chinese characters with strokes and then combining the strokes to mimic the font writing style. With the application of machine learning, researchers turned to deep learning for solutions to Chinese character generation. Zi2zi [3] was the first method to generate Chinese characters with a GAN. Subsequently, researchers have successively started to use GANs, such as DCFont [4] and PEGAN [5], for Chinese character generation. PEGAN improved the performance of zi2zi by introducing multiscale image pyramid refinement to transfer information. DCFont is an end-to-end learning system that automatically generates the entire Chinese character library from a few written characters. Handwritten-CycleGAN [6] was proposed based on the foundation of CycleGAN, making it unnecessary to generate a Chinese character dataset in a pairwise manner. FontRNN [14] treated Chinese characters as sequences of points (writing trajectories) and proposed handling the task of Chinese character generation by means of recurrent neural networks.

However, most of the Chinese characters generated by these methods suffer from various problems, such as missing strokes, incoherence and poor learning of character styles. Later, ChiroGAN [15] proposed extracting the skeleton information of Chinese characters using erosion and expansion methods when generating Chinese characters and then using the extracted information to generate a certain style of Chinese characters. FontRL [16] also used stroke skeleton information to generate Chinese characters. CalliGAN [7] used stroke information as a prior knowledge and formed a set of vectors to pass into the generator to generate Chinese characters to constrain the problem of missing strokes. Unfortunately, none of the above mentioned methods can solve the problems of stroke errors and missing strokes in the generation of handwritten Chinese characters. Therefore, the work of this paper is based on the idea of radical learning and improves the existing problems to solve the problems of missing strokes, incorrect strokes, and poor style of generating Chinese characters.

Although the ideal of radical learning was adopted in [17] and [18], the works in these two references and ours are completely different. In [17], a radical sequence was used as input to generate Chinese character images, while our method takes Chinese character images as input and aims to generate another handwritten stylized image of Chinese characters. In [18], a bidirectional LSTM-based network was designed to learn radical images from input images, and a structure level loss was applied to compute the loss between the generated radical images and generated whole Chinese character images. Different from the above works, we add a GRU model to learn radical sequence representation from the generated Chinese character image and define cross-entropy loss between the standard radical sequence and the predicted radical sequence as a method to reduce the stroke errors in the generated Chinese character image.

3 PROPOSED METHOD

Given two different domains $< X, Y >$, the objective of this work is to build two translation mappings, $f_{x->y} = p(Y \mid X)$ and $f_{y->x} = p(X \mid Y)$. The overall framework of the proposed RC-GAN in this paper is shown in Figure 2. In the following section, we first introduce the overview of the structure of the proposed method and then describe the proposed GRU-based radical learning network (RLN) in detail. Finally, the overall training of the proposed model is presented at the end of this section. To make this paper easy to follow, all of the abbreviations are listed in Table 1.

Abbreviations	Full Names
GAN	Generative adversarial network
HCCAG	Handwritten Chinese character automatic generation
RC-GAN	Radical constraint-based GAN
GRU	Gated recurrent unit
CCI	Chinese character image
DNN	Deep neural network
PEGAN	Pyramid embedded GAN
NICE-GAN	No independent component for encoding GAN
AdaLIN	Adaptive layer-instance normalization
RLN	Radical learning network
ReLU	Rectified linear unit
MLP	Multilayer perceptron
MSE	Mean square error
SSIM	Structural similarity index measure

Table 1. Abbreviations list

3.1 General Formulation

The proposed RC-GAN adopts the structure of CycleGAN [13], including two discriminators and two generators, as shown in Figure 2. Inspired by NICE-GAN [9], in this structure, the first part of the discriminator is reused as the image feature extractor in the generator.

In the proposed method, two different domain CCIs, A_x and B_y are treated as the input, where A_x represents the image of Chinese character A with the X style. Similarly, B_y represents the image of Chinese character B with the Y style. For Chinese characters in printed font style, CCIs are generated by TrueTypeFont, while for Chinese characters in handwriting style, CCIs are generated by specific people. For example, Lanting font was written by Wang Xizhi. The output of the proposed method is images $\tilde{B}_x$ and $\tilde{A}_y$, where $\tilde{A}_y$ represents the generated CCI of Chinese character A with Y style from A_x, and $\tilde{B}_x$ represents the generated CCI of Chinese character B with X style from B_y. The translation from the X domain to the Y domain is very similar to that from the Y domain to the X domain. Therefore, in the following section, we take the translation from the X domain to the Y domain as an example to explain the calculation process in detail.

The overall structure of CCI generation from the X domain to the Y domain contains three main processes: translation from A_x to $\tilde{A}_y$, reconstruction of Chinese character A in the X domain, and cycle translation from $\tilde{A}_y$ to generate the Chinese character image A in the X domain. Given the input image A_x, image feature f_{ax} is extracted from the encoder network, followed by the classifier and the generator.

The process of CCI generation from the X domain to the Y domain and Chinese character reconstruction in the X domain can be expressed as follows:

$$\begin{aligned} f_{ax} &= E_x(A_x), \\ \tilde{A}_y &= G_y(f_{ax}), \\ \tilde{A}_{rx} &= G_x(f_{ax}), \end{aligned} \tag{1}$$

where $E_x(\cdot)$ represents the encoder network, $G_y(\cdot)$ represents the generator of the Y domain, and $G_x(\cdot)$ represents the generator of the X domain. $\tilde{A}_{rx}$ represents the reconstructed CCI of Chinese character A in the X domain.

Then, the generated CCI $\tilde{A}_y$ is fed into the encoder of the Y domain to capture the features, and then the generator in the X domain is applied to generate the original CCI of Chinese character A in the X domain from the feature of $\tilde{A}_y$, which is defined as the cycle translation. The calculation process is defined in Equation (2).

$$\begin{aligned} \tilde{f}_{ay} &= E_y(\tilde{A}_y), \\ \tilde{A}_{cx} &= G_x(\tilde{f}_{ay}), \end{aligned} \tag{2}$$

where $E_y(\cdot)$ represents the encoder network for the Y domain image, $\tilde{f}_{ay}$ represents the feature of generated CCI $\tilde{A}_y$ and $\tilde{A}_{cx}$ represents the cyclically generated CCI of character A in domain X.

In addition, the encoder network and classifier are combined to form the discriminator. D_X and D_Y represent the discriminators of the X domain and Y domain, respectively. Finally, the $\tilde{f}_{ay}$ and the f_{by} in the Y domain extracted from encoder $E_y(\cdot)$ are fed into the classifier in D_Y to calculate adversarial loss. The reconstruction loss between the generated CCI $\tilde{A}_{rx}$ and original CCI A_x is calculated by the L1 distance function. The distance between the cyclically generated CCI $\tilde{A}_{cx}$ and the original CCI A_x is calculated as cycle loss. These loss functions are fused to train the model.

Discriminator: In the proposed method, the structures of these two discriminators are exactly the same. Therefore, only discriminator $D_x(\cdot)$ is explained in detail in this paper. $D_x(\cdot)$ contains an encoder $E_x(\cdot)$, and a classifier $C_x(\cdot)$.

Input image A_x is fed into E_x for downsampling through convolution, spectral normalization and leaky rectified linear unit (ReLU) activation functions to obtain the encoded feature maps. For the classifier to determine whether a generated Chinese character image is true or false, these feature maps are fed into global average and max pooling to obtain the weight information for each feature map. Specifically, the C_x of the discriminator uses a multiscale classifier with three components: C_x^0 for the local scale (10×10 receptive field), C_x^1 for the middle scale (70×70 receptive field) and C_x^2 for the global scale (286×286 receptive field). We use the residual attention mechanism to obtain attention maps, which are connected to classifier C_x^0.

The output of C_x^0 after downsampling is connected to two branches: one branch is linked to C_x^1, and the other branch is input to C_x^2 after further downsampling through the convolution layer. C_x^0, C_x^1 and C_x^2 are all trained to determine whether an image is true or false. Figure 3 illustrates the discriminator in detail.

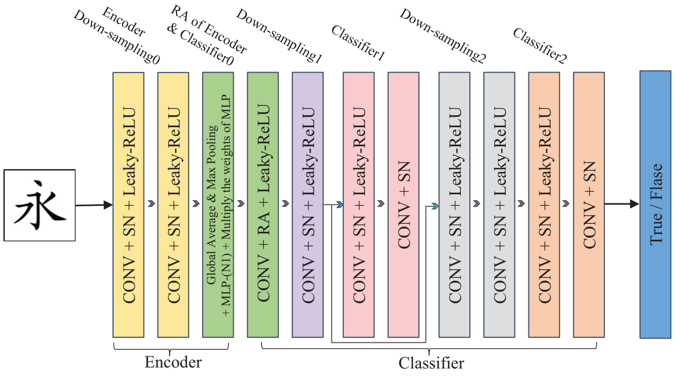

Figure 3. Illustration of the discriminator

Generators: For the generator, the feature maps output by the encoder are fed into the generator, which are sampled to obtain a new feature vector. We divide the feature vector into two branches, with one side going through global average pooling, multilayer perceptron (MLP), and the ReLU activation function to obtain α, β in AdaLIN. The other side is generated with six adaptive residual blocks, two subpixel convolutions, and one convolution to obtain the target image. The details are shown in Figure 4.

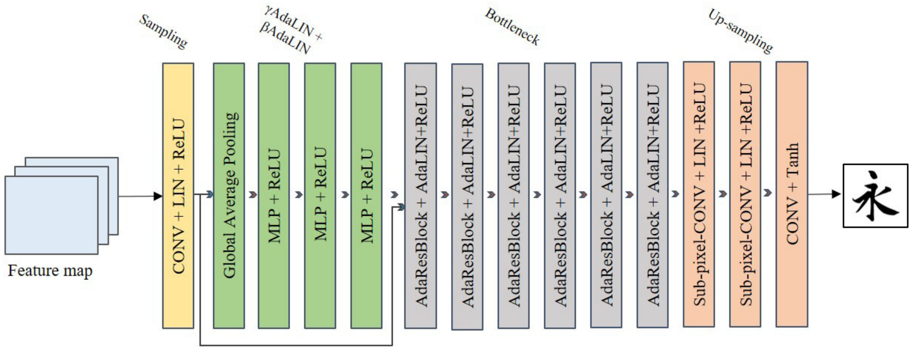
Sampling
γAdaLIN + βAdaLIN
Bottleneck
Up-sampling
Feature map
CONV + LIN + ReLU
Global Average Pooling
MLP + ReLU
MLP + ReLU
MLP + ReLU
AdaResBlock + AdaLIN+ReLU
AdaResBlock + AdaLIN+ReLU
AdaResBlock + AdaLIN+ReLU
AdaResBlock + AdaLIN+ReLU
AdaResBlock + AdaLIN+ReLU
AdaResBlock + AdaLIN+ReLU
Sub-pixel-CONV + LIN +ReLU
Sub-pixel-CONV + LIN +ReLU
CONV + Tanh
永

Figure 4. Illustration of the generator

3.2 Radical Constraint Module

To ensure the correctness of the radical in the generated CCI, a GRU-based RLN is added into the GAN model. Similar to the discriminator and generator, there are also two RLN models, which are used in the X and Y domains. These two RLN models adopt the same structure. In this section, we describe only the RLN in the X domain. In addition, in our work, the input image is correct by default, and the main purpose of this work is to reduce stroke errors in the generated Chinese character image. Radical learning on the generated image can help find stroke errors in that image and guide the generation network to output a more accurate image. Additionally, during the training process, the radical information of each Chinese character is known and can be directly used as a label for radical learning. Therefore, we only apply the radical constraint module to the generated image.

In previous works on radical learning [19], thirteen different basic structures of Chinese characters were defined, as shown in Figure 5. For example, "a" represents the left-right spatial relationship of two radicals, "d" represents the upper-lower spatial relationship and "single" represents a single radical to form a Chinese character. In addition, we use a pair of braces to constrain a single structure in character caption. Taking Chinese character "賢" as an example, it is captioned as "d { a {臣 又} d {目 八} }", where "臣", "又", "目" and "八" are radicals in the dictionary. Similarly, each Chinese character can be disassembled into a sequence of radical relations. Radical learning can be regarded as a problem involving sequence prediction from images.

In our experiment, the length of the radical dictionary is 473, including 460 radical codes and 13 spatial structure relationship codes. Each radical or spatial structure relationship is encoded as a one-hot vector $e_i \in \mathbb{R}^{473}$ by the dictionary, where e_i denotes the vector with a 1 in the i^{th} coordinate and 0 elsewhere. For example, for the radical "臣", it is coded as 412 in the dictionary and transferred into a one-hot vector e_{412}. Similarly, the spatial structure relationship "d" is coded as 266 in the dictionary and represented as e_{266}. Finally, the Chinese character "賢" is represented by the sequence $\{e_{266}, e_{134}, e_{412}, e_8, e_{266}, e_{369}, e_{254}\}$, where e_{134}, e_8, e_{369} and e_{254} represent "a", "又", "目" and "八" respectively.

For the generated CCI $\tilde{A}_y$, the RLN is connected to the encoder network, as shown in Figure 2. The image feature $\tilde{f}_{ay} \in \mathbb{R}_{w\times h\times l}$ is applied as the input of the RLN, where w, h and l represent the width, height and length of the feature map, respectively. In this paper, we use GRU to implement the RLN. Assume that $R = r_1, \ldots, r_T$ represents the radical sequence of each CCI and that T is the length of this sequence. $r_i \in \mathbb{R}^K$ represents the one-hot vector of each radical, where K is the length of the radical dictionary. We use a unidirectional GRU to generate the radical sequence word by word. For example, for the Chinese character "賢", $r_1 = e_{266}$. First, the input feature is transferred into the feature sequence Q, as defined in Equation (3).

$$Q = q_1, \ldots, q_{w\times h}, \quad q_i \in \mathbb{R}^l. \tag{3}$$

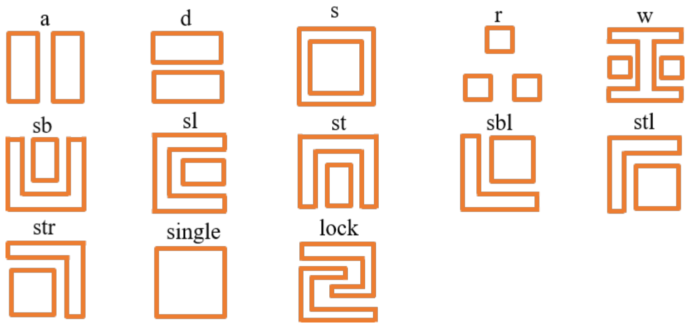

Figure 5. Illustration of the thirteen common spatial structure relationships among Chinese characters defined in [19]. Each component represents one radical.

Then, the probability of each predicted word is computed by the context vector c_t, and the current GRU hidden state represents the previous target word r_{t-1} using the following equation:

$$P(r_t \mid r_{t-1}) = f_s(W_o \times f_{max}(\mathbf{E} \times r_{t-1} + W_s \times s_t + W_c \times c_t)), \tag{4}$$

where $\mathbf{W}_o \in \mathbb{R}^{K \times \frac{m}{2}}$, $\mathbf{W}_s \in \mathbb{R}^{m \times n}$, $\mathbf{W}_c \in \mathbb{R}^{m \times L}$, $f_s(\cdot)$ denotes a softmax activation function, $f_{max}((\cdot))$ denotes a maxout activation function, $\mathbf{E}$ denotes the embedding matrix, m and n are the dimensions of embedding and GRU decoder, respectively.

Finally, the RLN uses two unidirectional GRU layers to compute implicit state $\mathbf{s}_t$:

$$\begin{aligned}\hat{\mathbf{s}}_t &= \mathrm{GRU}\left(\mathbf{r}_{t-1}, \mathbf{s}_{t-1}\right), \\ \mathbf{c}_t &= f_{\text{catt}}\ \left(\hat{\mathbf{s}}_t, \mathbf{Q}\right), \\ \mathbf{s}_t &= \mathrm{GRU}\left(\mathbf{c}_t, \hat{\mathbf{s}}_t\right),\end{aligned} \tag{5}$$

where $\mathbf{s}_{t-1}$ denotes the previous hidden state and $\hat{\mathbf{s}}_t$ is a prediction of the current GRU hidden state. $f_{\text{catt}}\ (\cdot)$ denotes the coverage-based attention model [20] parameterized as an MLP:

$$\begin{aligned}\mathbf{F} &= \mathbf{J} * \sum_{i=1}^{t-1} \boldsymbol{\alpha}_i, \\ e_{ti} &= \boldsymbol{\nu}_{\text{att}}^{\mathrm{T}} \tanh\left(\mathbf{W}_{\text{att}} \hat{\mathbf{s}}_t + \mathbf{U}_{\text{att}} \mathbf{a}_i + \mathbf{U}_f \mathbf{f}_i\right), \\ \alpha_{ti} &= \frac{\exp\left(e_{ti}\right)}{\sum_{k=1}^{L} \exp\left(e_{tk}\right)}.\end{aligned} \tag{6}$$

Coverage vector $\mathbf{F}$ is calculated from the sum of all past attention probabilities, where the spatial attention coefficient of $\mathbf{a}_i$ at time t is denoted as α_{ti}. Let n' denote the attention dimension and M denote the number of feature maps of filter $\mathbf{J}$; then, $\boldsymbol{\nu}_{\text{att}} \in \mathbb{R}^{n'}, \mathbf{W}_{\text{att}} \in \mathbb{R}^{n' \times n}, \mathbf{U}_{\text{att}} \in \mathbb{R}^{n' \times D}$ and $\mathbf{U}_f \in \mathbb{R}^{n' \times M}$. With weight α_{ti}, we compute the context vector $\mathbf{c}_t$ as follows:

$$\mathbf{c}_t = \sum_{i=1}^{L} \alpha_{ti} \mathbf{a}_i. \tag{7}$$

In the training stage, the objective function is minimized as follows:

$$L_{radical}^{x \to y} = -\sum_{i=1}^{T} \log(P(\tilde{r}_i)), \tag{8}$$

where $\tilde{r}_i$ is the predicted radical at the i^{th} time step.

3.3 Loss Function

In summary, there are four loss functions used in the proposed method, including adversarial loss, cycle-consistency loss, reconstruction loss and radical constraint loss. The encoder in this paper is not only part of the discriminator but also the input to the generator. Therefore, referring to NICE-GAN [9], instead of regular adversarial loss training, decoupled training is used in the experiments to freeze the encoder weights when minimizing adversarial loss, reconstruction loss, cycle-consistency loss and radical constraint loss and to train the encoder when maximizing training adversarial loss. The details of this loss function are specified as follows:

Adversarial loss: The purpose of adversarial loss is to prompt domain transfers to produce higher-quality images.

$$\min_{G_y} \max_{D_y = (C_y \circ E_y)} L_{gan}^{x \to y} := \mathbb{E}_{y \sim Y}\left[(D_y(B_y))^2\right] + \mathbb{E}_{x \sim X}\left[(1 - D_y(G_y(E_x(A_x))))^2\right]. \tag{9}$$

Cycle-consistency loss: To reduce the variability between the generated and real images, a cycle-consistency loss is added.

$$\min_{\substack{G_y \\ G_x}} L_{cycle}^{x \to y} := \mathbb{E}_{x \sim X}\left[\|A_x - G_x(E_y(G_y(E_x(A_x))))\|_1\right]. \tag{10}$$

Reconstruction loss: Reconstruction loss is similar to cycle-consistency loss in that the two domain generators are induced to produce images that are consistent in the hidden vector space through reconstruction.

$$\min_{G_x} L_{\text{recon}}^{x \to y} := \mathbb{E}_{x \sim X}\left[\|A_x - G_x(E_x(A_x))\|_1\right]. \tag{11}$$

In particular, the loss functions from y to x, $L_{gan}^{y \to x}, L_{\text{cycle}}^{y \to x}, L_{\text{recon}}^{y \to x}, L_{\text{radical}}^{y \to x}$, are defined the same way.

Full objective: Finally, the final objective is defined by fusing these loss functions:

$$L_{disct} = \max_{E_x,C_x,E_y,C_y} \lambda_1 L_{gan}, \tag{12}$$

$$L_{genet} = \min_{G_y,G_x} \lambda_1 L_{gan} + \lambda_2 L_{\text{cycle}} + \lambda_3 L_{\text{recon}} + \lambda_4 L_{radical}, \tag{13}$$

$$L_{gan} = L_{gan}^{x\rightarrow y} + L_{gan}^{y\rightarrow x}, \tag{14}$$

$$L_{cycle} = L_{cycle}^{x\rightarrow y} + L_{cycle}^{y\rightarrow x}, \tag{15}$$

$$L_{recon} = L_{recon}^{x\rightarrow y} + L_{recon}^{y\rightarrow x}, \tag{16}$$

$$L_{radical} = L_{radical}^{x\rightarrow y} + L_{radical}^{y\rightarrow x}, \tag{17}$$

where L_{disct} represents the objective function of the discriminator, and L_{genet} represents the objective function of the generator. To make a fair comparison with the baseline model [9], λ_1, λ_2, and λ_3 are set to 1, 10 and 10, respectively. Moreover, λ_4 is discussed in the experiment.

4 EXPERIMENTS

4.1 Dataset and Implementation Details

To verify the effectiveness of this method, we evaluate the proposed method on three pairs of Chinese character translations with different styles, including DFKai-SB and Pen-Kai scripts, DFKai-SB and running scripts, and SIM-Kai and Lanting scripts, for a total of five fonts. All five fonts are used as target fonts. Among them, DFKai-SB and SIM-Kai are printed fonts, while the other three scripts are handwritten fonts. The DFKai-SB, Pen-Kai and running scripts each include 1 000 images, while the Lanting script has 324 images. Each image contains only one Chinese character. Some examples are shown in Figure 6.

In the experiment, all of images are cropped and resized to 256×256 for training and testing. We use ReLU as the activation function in the generator and leaky-ReLU with a slope of 0.2 in the discriminator. We train all models using the Adam optimizer with the learning rate 0.0001 and $(\alpha, \beta) = (0.5, 0.999)$ on NVIDIA RTX 3090 graphic card. The batch size is set to 1 for all experiments. We also use a weight decay at the rate of 0.0001. The proposed model is implemented by PyTorch. In the training process, the model is trained by 100 K iterations.

4.2 Evaluation Metrics

Mean square error (MSE) and structural similarity index measure (SSIM): To evaluate the similarity between the generated image and the real image, three evaluation metrics are applied: MSE, SSIM and human evaluation.

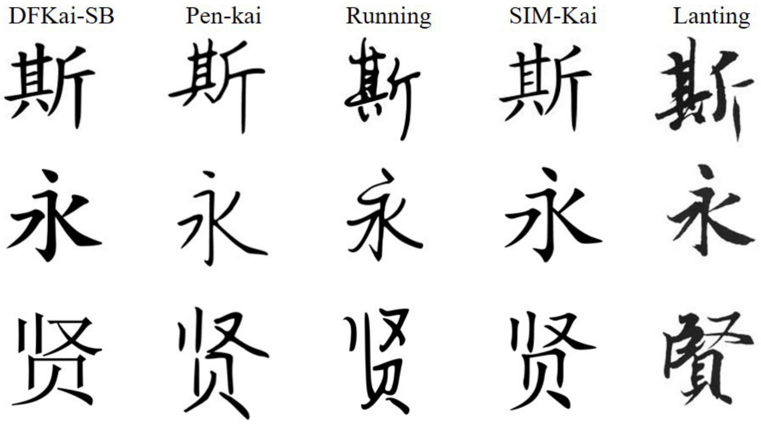

Figure 6. Illustrations of CCIs of different styles

MSE and SSIM are defined as in Equation (18).

$$
\begin{aligned}
MSE &= \frac{1}{Z}\sum_{i=1}^{Z}(I_i - \tilde{I}_i)^2, \\
SSIM &= \frac{1}{Z}\sum_{i=1}^{Z}\frac{(2\mu_{I_i}\mu_{\tilde{I}_i} + c_1)(2\sigma_{I_i\tilde{I}_i} + c_2)}{(\mu^2_{I_i} + \mu^2_{\tilde{I}_i} + c_1)(\sigma^2_{I_i} + \sigma^2_{\tilde{I}_i} + c_2)},
\end{aligned}
\tag{18}
$$

where I_i represents the real CCI, $\tilde{I}_i$ represents the generated CCI, and Z is the number of images. μ_{I_i} represents the mean of image I_i, and $\mu_{\tilde{I}_i}$ represents the mean of image $\tilde{I}_i$. $\sigma^2_{I_i}$ represents the variance in image I_i. $\sigma^2_{\tilde{I}_i}$ represents the variance in image $\tilde{I}_i$. $\sigma_{I_i\tilde{I}_i}$ represents the covariance between I_i and $\tilde{I}_i$. c_1 and c_2 are constants. The higher the value of SSIM is, the better the similarity between the generated image and the real image. Furthermore, the lower the MSE value is, the better the performance.

Human evaluation for radical error: To evaluate the quality of the generated CCIs, the radical error of the generated image is counted by humans. We select three volunteers who have been trained in Chinese calligraphy to find the radical error in the generated images of different methods. In this work, we define four radical errors: missing strokes, broken strokes, extra strokes, and incomplete strokes, as shown in Figure 4.2. The average number of radical errors is applied as the evaluation metric.

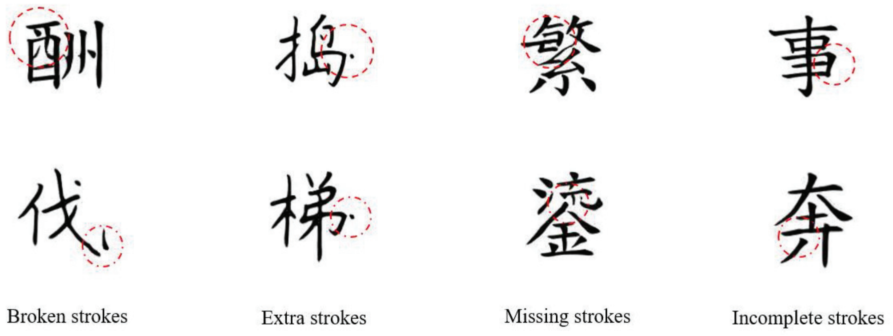

Figure 7. Examples of these four radical errors

4.3 Experimental Results and Discussion

Parameter Discussion. First, we discuss the influence of parameter λ_4 on the overall loss function. λ_4 is set from 10 to 50. The experimental results are shown in Figure 8. It can be found from these results that the value of λ_4 has little effect on the MSE and SSIM results. Therefore, in this work, λ_4 is set to 30, which exhibits the best performance.

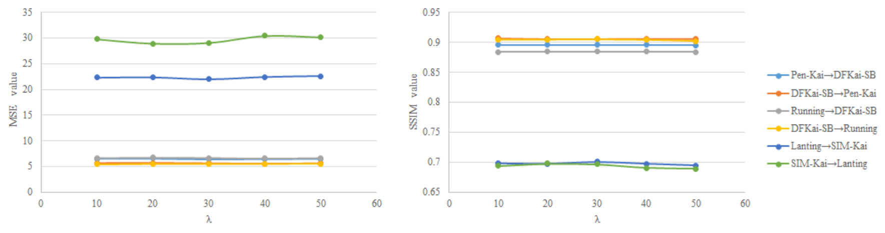

Figure 8. Experimental results of different λ_4

In addition, we show the accuracy of radical sequence prediction in Table 2. BLEU, a widely used metric in sequence prediction, is applied to evaluate the performance of radical sequence prediction in RC-GAN. From these results, it can be observed that compared with the handwritten fonts, the radical prediction is more effective for the printed fonts DFKai-SB and SIM-Kai.

Comparison with existing methods. To evaluate the effectiveness of the proposed method, we compare the proposed RC-GAN with the existing GAN for HCCAG. The comparison results of MSE are shown in Table 3, those of SSIM are shown in Table 4, and those of human evaluation are reported in Tables 5 and 6.

Style	BLEU-1
Pen-Kai → DFKai-SB	0.76
DFKai-SB → Pen-Kai	0.44
Running → DFKai-SB	0.60
DFKai-SB → Running	0.48
Lanting → SIM-Kai	0.85
SIM-Kai → Lanting	0.49

Table 2. BLEU values for radical sequence prediction

Style	Ours	NICE-GAN	Handwritten-CycleGAN	zi2zi
Pen-Kai → DFKai-SB	**6.408**	6.596	11.149	7.389
DFKai-SB → Pen-Kai	**5.591**	5.746	9.601	6.246
Running → DFKai-SB	6.667	**6.633**	11.355	8.996
DFKai-SB → Running	**5.507**	5.520	9.128	6.548
Lanting → SIM-Kai	22.039	22.263	25.306	**21.091**
SIM-Kai → Lanting	29.044	**28.745**	31.367	36.714

Table 3. MSE values for different methods

MSE evaluation. From the MSE value results, we find that for the HCCAG task, compared with the CycleGAN and zi2zi methods, the proposed method and NICE-GAN achieve a lower MSE value in most experiments. Specifically, compared with NICE-GAN, our methods have the lowest MSE value for the translations from Pen-Kai script to DFKai-SB script, DFKai-SB script to Pen-Kai script and Pen-Kai script to running script.

SSIM evaluation. From the SSIM value results, we find that the results of all methods are very similar. Nonetheless, the proposed method achieves the best performance on the translations from Pen-Kai script to DFKai-SB script, running script to DFKai-SB script, DFKai-SB script to running script and SIM-Kai script to Lanting script. Although SSIM is a commonly used metric for evaluating the performance of HCCAG, we argue that it is not entirely suitable and reasonable. SSIM uses the similarity of the distribution of image brightness to measure the similarity of images. It is difficult to measure the tiny difference

Style	Ours	NICE-GAN	Handwritten-CycleGAN	zi2zi
Pen-Kai → DFKai-SB	**0.895**	0.895	0.887	0.884
DFKai-SB → Pen-Kai	0.906	0.906	0.899	**0.909**
Running → DFKai-SB	**0.885**	0.884	0.877	0.858
DFKai-SB → Running	**0.905**	0.903	0. 899	0.897
Lanting → SIM-Kai	0.700	0.696	0.684	**0.716**
SIM-Kai → Lanting	**0.697**	0.696	0.680	0.639

Table 4. SSIM values for different methods

DFKai-SB → Pen-Kai	Ours	NICE-GAN	Handwritten-CycleGAN
Extra strokes	6	13	2
Missing strokes	1	1	79
Broken strokes	14	44	52
Incomplete strokes	21	4	55
Total	42	62	188

Table 5. Illustrations of the Pen-Kai style Chinese characters statistics for DFKai-SB → Pen-Kai

DFKai-SB → Running	Ours	NICE-GAN	Handwritten-CycleGAN
Extra strokes	35	78	5
Missing strokes	10	44	78
Broken strokes	9	38	27
Incomplete strokes	16	60	82
Total	70	220	192

Table 6. Illustrations of the running style Chinese characters statistics for DFKai-SB → running

between Chinese characters in images, for example, stroke errors in CCIs. Therefore, to evaluate the performance of these different methods, human evaluation is adopted in this work.

Human evaluation. Tables 5 and 6 show the number of stroke errors in the generated CCIs. In this experiment, we translate the DFKai-SB script to the Pen-Kai script and the running script. Both DFKai-SB and Pen-Kai are regular scripts, and it is relatively simple to transfer the DFKai-SB script to the Pen-Kai script. Due to the large difference between the DFKai-SB script and the running script, this becomes a more difficult task. In the generated Pen-Kai script and running script images, the numbers of stroke errors of the proposed method are 42 and 70, respectively. Compared with NICE-GAN, the numbers of stroke errors are reduced by 20 and 150. In Tables 5 and 6, we also show the detailed values for the four types of stroke errors. For the Pen-Kai script, the proposed method greatly reduces the number of broken errors. For the running script, the number of these four errors also declines significantly.

In summary, from the above comparison results, it can be seen that the proposed RC-GAN is slightly superior to the state-of-the-art methods in commonly used MSE and SSIM indicators. Additionally, in terms of the statistical indicators of stroke errors, our method can significantly reduce the number of stroke errors and improve HCCAG performance. These comparison results verify the effectiveness of the proposed method. In addition, NICE-GAN is applied as the baseline model of the proposed method. The comparison results between RC-GAN and NICE-GAN also verify the effectiveness of the proposed radical constraint module.

Visualization evaluation. Some visualization results of different methods are shown in Figures 9 and 10. From Figures 9 and 10, we also find that some stroke errors of the existing methods were corrected in the proposed method. These visual analysis results further verify that the proposed radical constraint in this paper is helpful for improving the accuracy of Chinese character generation.

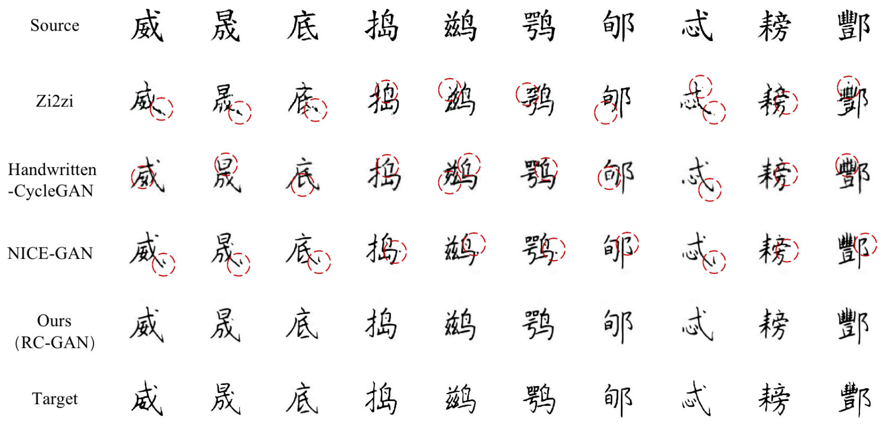

Figure 9. Examples of Pen-Kai script images generated by different methods. The red circle indicates the stroke error.

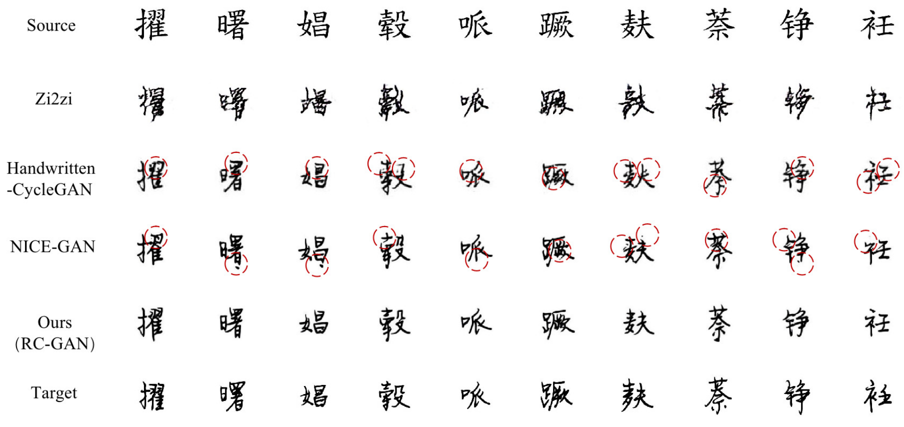

Figure 10. Examples of the running script images generated by different methods. The red circle indicates the stroke error.

Unseen Chinese character generation. To further verify the effectiveness of the proposed method, we apply the proposed method to generate unseen Chinese characters. In this experiment, 1 000 handwritten running style Chinese characters that are not used in the training process are regarded as the testing set. The experimental results are shown in Tables 7 and 8. Table 7 shows that the proposed method has a lower MSE value. Table 8 shows the number of stroke errors in these generated unseen Chinese characters. From these comparison results, we find that there are fewer stroke errors in the proposed method.

Method	MSE	SSIM
Ours	5.55	0.90
NICE-GAN	5.61	0.90
Handwritten-CycleGAN	9.48	0.89
zi2zi	6.74	0.89

Table 7. Comparison results on unseen running style Chinese characters

Running	Ours	NICE-GAN	Handwritten-CycleGAN
Extra strokes	56	82	20
Missing strokes	5	108	90
Broken strokes	15	21	22
Incomplete strokes	15	90	121
Total	91	301	253

Table 8. Illustrations of the unseen running style Chinese characters statistics

Comparison of different fonts and limitations. In addition, from the comparison results in Tables 5 and 6, we find that the GAN-based methods, including the proposed RC-GAN, have better performance in hard pen HCCAG than in soft pen HCCAG. In detail, for Pen-Kai, running, and DFKai-SB scripts, the MSE value of these GAN-based methods falls in the range $[5, 6]$. However, for the Lanting script, it exceeds 29. Similarly, for Pen-Kai, running, and DFKai-SB scripts, the SSIM values of these methods exceed 0.85, but this value is less than 0.7 for the Lanting script.

Compared to the previous tasks, generating the Lanting script is more challenging. We argue that in the Lanting script, the stroke width is quite different compared to other fonts. The widths of the previous hard pen fonts are relatively consistent. However, the width of the Lanting script changes with the stroke. Because of some continuous strokes and changes in the stroke that constitute its style, it is not very appropriate to use stroke error to measure the quality of the generated Lanting script image. Therefore, in this paper, we show some visual comparison results in the Lanting script in Figure 11. Some stroke errors of the state-of-the-art methods have also been labeled in this figure. From this comparison, we also find that the proposed radical constraint is effective for the generated Lanting script. Certainly, how to effectively generate Chinese characters with the Lanting script and quantitatively evaluate the artistic effects of this script are still challenging problems in HCCAG. Thus, we attempt to conduct an in-depth discussion of the possible directions for future work.

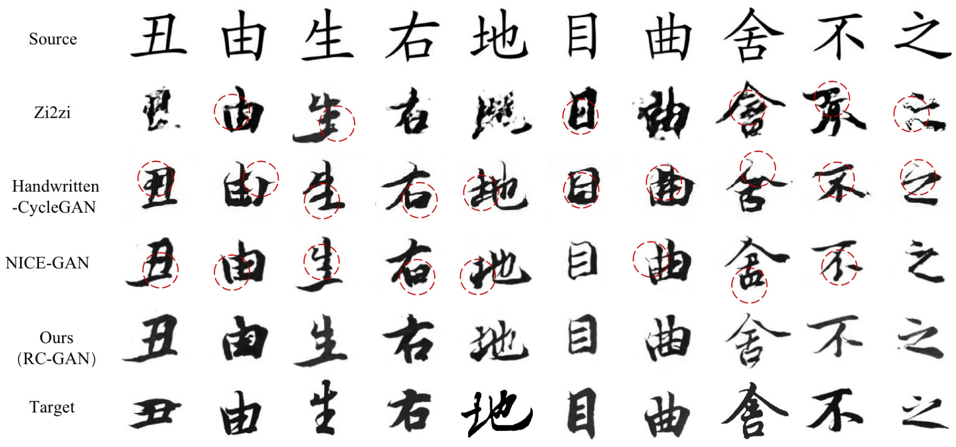

Figure 11. Examples of Lanting script images generated by different methods. The red circle indicates the stroke error.

5 CONCLUSIONS

In this paper, we propose incorporating a radical constraint module in GAN for the HCCAG task. A GRU decoder is used to achieve radical sequence learning from the input CCI. Finally, a new HCCAG method, named RC-GAN, is proposed in this work. We use three styles of Chinese characters, including Kai, running and Lanting scripts, to verify the effectiveness of the proposed method. The experimental results show that the proposed method achieves competitive results on common MSE and SSIM indicators. Additionally, compared with the state-of-the-art methods, the proposed method can obviously reduce the number of stroke errors in the generated CCIs.

However, due to the large number of strokes, the complexity of the structure and the variety of expressions, Chinese character generation remains a great challenge. First, the style of Chinese calligraphy is very personalized. For the same font, the styles written by different people are quite different. Learning the styles of specific people, especially when the quantity of data is small, is an important issue for future research. Moreover, how to evaluate the artistic effect of generating CCIs is also a problem worthy of in-depth study. We argue that the commonly used MSE and SSIM measures are pixel-based evaluation indicators that judge only image similarity. The stroke error proposed in this work evaluates the completeness and correctness of the generated Chinese characters. However, there is still no good solution for evaluating artistic effects.

Acknowledgment

This work was supported by the Natural Science Foundation of China (No. 61871196 and 62001176); Natural Science Foundation of Fujian Province of China (No. 2020-J01085); the Promotion Program for Young and Middle-Aged Teacher in Science and Technology Research of Huaqiao University (ZQN-YX601).

REFERENCES

[1] Goodfellow, I.—Pouget-Abadie, J.—Mirza, M.—Xu, B.—Warde-Farley, D.—Ozair, S.—Courville, A.—Bengio, Y.: Generative Adversarial Nets. In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., Weinberger, K. Q. (Eds.): Advances in Neural Information Processing Systems 27 (NIPS 2014). Curran Associates, Inc., 2014, pp. 2672–2680, doi: 10.48550/arXiv.1406.2661.

[2] Isola, P.—Zhu, J. Y.—Zhou, T.—Efros, A. A.: Image-to-Image Translation with Conditional Adversarial Networks. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 5967–5976, doi: 10.1109/CVPR.2017.632.

[3] Tian, Y.: Zi2zi: Master Chinese Calligraphy with Conditional Adversarial Networks. 2017, `https://github.com/kaonashi-tyc/zi2zi`.

[4] Jiang, Y.—Lian, Z.—Tang, Y.—Xiao, J.: DCFont: An End-to-End Deep Chinese Font Generation System. SIGGRAPH Asia 2017 Technical Briefs (SA '17), ACM, 2017, doi: 10.1145/3145749.3149440.

[5] Sun, D.—Zhang, Q.—Yang, J.: Pyramid Embedded Generative Adversarial Network for Automated Font Generation. 2018 24th International Conference on Pattern Recognition (ICPR), 2018, pp. 976–981, doi: 10.1109/ICPR.2018.8545701.

[6] Chang, B.—Zhang, Q.—Pan, S.—Meng, L.: Generating Handwritten Chinese Characters Using CycleGAN. 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), 2018, pp. 199–207, doi: 10.1109/WACV.2018.00028.

[7] Wu, S. J.—Yang, C. Y.—Hsu, J. Y. J.: CalliGAN: Style and Structure-Aware Chinese Calligraphy Character Generator. CoRR, 2020, doi: 10.48550/arXiv.2005.12500.

[8] Zhang, J.—Zhu, Y.—Du, J.—Dai, L.: Trajectory-Based Radical Analysis Network for Online Handwritten Chinese Character Recognition. 2018 24th International Conference on Pattern Recognition (ICPR), 2018, pp. 3681–3686, doi: 10.1109/ICPR.2018.8546074.

[9] Chen, R.—Huang, W.—Huang, B.—Sun, F.—Fang, B.: Reusing Discriminators for Encoding: Towards Unsupervised Image-to-Image Translation. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 8165–8174, doi: 10.1109/CVPR42600.2020.00819.

[10] Zhang, J.—Du, J.—Dai, L.: Radical Analysis Network for Learning Hierarchies of Chinese Characters. Pattern Recognition, Vol. 103, 2020, Art. No. 107305, doi: 10.1016/j.patcog.2020.107305.

[11] Mirza, M.—Osindero, S.: Conditional Generative Adversarial Nets. CoRR, 2014, doi: 10.48550/arXiv.1411.1784.

[12] Arjovsky, M.—Chintala, S.—Bottou, L.: Wasserstein Generative Adversarial Networks. In: Precup, D., Teh, Y. W. (Eds.): Proceedings of the 34th International Conference on Machine Learning (ICML '17). Proceedings of Machine Learning Research (PMLR), Vol. 70, 2017, pp. 214–223, `http://proceedings.mlr.press/v70/arjovsky17a/arjovsky17a.pdf`.

[13] Zhu, J. Y.—Park, T.—Isola, P.—Efros, A. A.: Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. 2017 IEEE International Conference on Computer Vision (ICCV), 2017, pp. 2242–2251, doi: 10.1109/ICCV.2017.244.

[14] Tang, S.—Xia, Z.—Lian, Z.—Tang, Y.—Xiao, J.: FontRNN: Generating Large-Scale Chinese Fonts via Recurrent Neural Network. Computer Graphics Forum, Vol. 38, 2019, No. 7, pp. 567–577, doi: 10.1111/cgf.13861.

[15] Gao, Y.—Wu, J.: GAN-Based Unpaired Chinese Character Image Translation via Skeleton Transformation and Stroke Rendering. Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, 2020, No. 1, pp. 646–653, doi: 10.1609/aaai.v34i01.5405.

[16] Liu, Y.—Lian, Z.: FontRL: Chinese Font Synthesis via Deep Reinforcement Learning. Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35, 2021, No. 3, pp. 2198–2206, doi: 10.1609/aaai.v35i3.16318.

[17] Xue, M.—Du, J.—Zhang, J.—Wang, Z. R.—Wang, B.—Ren, B.: Radical Composition Network for Chinese Character Generation. In: Lladós, J., Lopresti, D., Uchida, S. (Eds.): Document Analysis and Recognition – ICDAR 2021. Springer, Cham, Lecture Notes in Computer Science, Vol. 12821, 2021, pp. 252–267, doi: 10.1007/978-3-030-86549-8_17.

[18] Huang, Y.—He, M.—Jin, L.—Wang, Y.: RD-GAN: Few/Zero-Shot Chinese Character Style Transfer via Radical Decomposition and Rendering. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J. M. (Eds.): Computer Vision – ECCV 2020. Springer, Cham, Lecture Notes in Computer Science, Vol. 12351, 2020, pp. 156–172, doi: 10.1007/978-3-030-58539-6_10.

[19] Wang, W.—Zhang, J.—Du, J.—Wang, Z. R.—Zhu, Y.: DenseRAN for Offline Handwritten Chinese Character Recognition. 2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR), IEEE, 2018, pp. 104–109, doi: 10.1109/ICFHR-2018.2018.00027.

[20] See, A.—Liu, P. J.—Manning, C. D.: Get to the Point: Summarization with Pointer-Generator Networks. Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL, 2017, pp. 1073–1083, doi: 10.18653/v1/P17-1099.

Xi-Ling Ye is currently working toward her M.Sc. degree in software engineering in the Department of Computer Science and Technology of Huaqiao University, Xiamen, China.

Hong-Bo Zhang received his Ph.D. degree in computer science from the Xiamen University in 2013. Currently, he is Associate Professor with the Department of Computer Science and Technology of Huaqiao University. His research interests include computer vision, pattern recognition, human action recognition and application.

Li-Jie Yang is Associate Professor with the Department of Computer Science and Technology of Huaqiao University. Her research interests include computer graphics and application.

Ji-Xiang Du is with the Huaqiao University. From February 2003 on, in pursuit for Ph.D. degree in pattern recognition and intelligent system in the University of Science and Technology of China (USTC), Hefei, China, and in December 2005, he received his Ph.D. degree. Now, he is Professor at the College of Computer Science and Technology at Huaqiao University. His current research mainly concerns pattern recognition and machine learning.