MetaScript: Few-Shot Handwritten Chinese Content Generation via Generative Adversarial Networks

Project of AI3604 Computer Vision, 2023 Fall, SJTU

Jiazi Bu* 521030910395

Qirui Li* 521030910397

Kailing Wang* 521030910356

Xiangyuan Xue* 521030910387

Zhiyuan Zhang* 521030910377

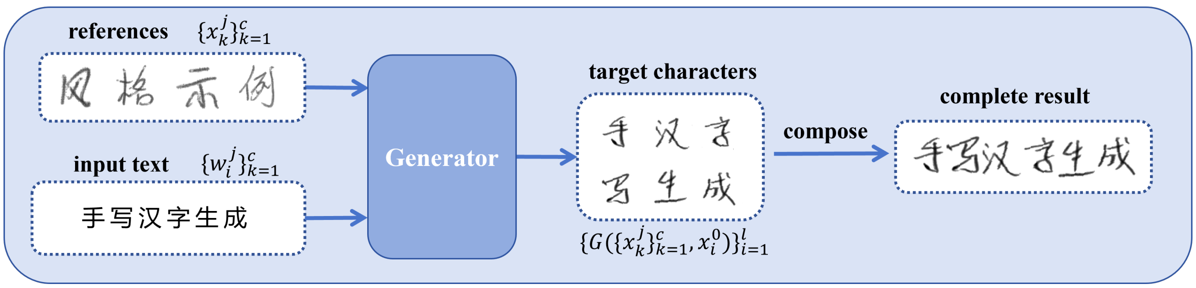

Figure 1. The overall pipeline of this project. We design a system to generate handwritten Chinese contents within a few-shot setting. The system is composed of a generator and a composer. The generator is trained to generate handwritten Chinese characters given a structure template and some style references. The composer stitches the generated characters into a handwritten style content.

Abstract

In this work, we propose MetaScript, a novel Chinese content generation system designed to address the diminishing presence of personal handwriting styles in the digital representation of Chinese characters. Our approach harnesses the power of few-shot learning to generate Chinese characters that not only retain the individual's unique handwriting style but also maintain the efficiency of digital typing. Trained on a diverse dataset of handwritten styles, MetaScript is adept at producing high-quality stylistic imitations from minimal style references and standard fonts. Our work demonstrates a practical solution to the challenges of digital typography in preserving the personal touch in written communication, particularly in the context of Chinese script. Notably, our system has demonstrated superior performance in various evaluations, including recognition accuracy, inception score, and Frechet inception distance. At the same time, the training conditions of our model are easy to meet and facilitate generalization to real applications. Our code is available at `https://github.com/xxyQwQ/metascript`.

*The authors contributed equally to this project. The authors are arranged in alphabetical order by last name.

1. Introduction

Chinese characters, utilized continuously for over six millennia by more than a quarter of the world's population, have been integral to education, employment, communication, and daily life in East Asia. The art of handwriting Chinese characters, transcends mere linguistic articulation, representing both the pinnacle of visual art and a medium for personal expression and cultivation [18]. There is an ancient Chinese saying, "seeing the character is like seeing the face." Throughout the process of personal growth, individuals develop distinct and characteristic handwriting styles. These styles can serve as symbols of one's identity. Since humanity's entry into the digital era, efficient yet characterless fixed fonts have supplanted handwritten text, eliminating the possibility of perceiving each other through the written word. This shift has engendered a sense of detachment from handwriting, significantly diminishing the personalization and warmth in textual communication.

To address this issue, we aim to devise a method that retains the individual's handwriting style while also harnessing the efficiency afforded by typing. However, Chinese characters, numbering over 100,000 distinct ideograms with diverse glyph structures, lack standardized stroke units. Generating handwritten Chinese characters in a specific style is challenging through the naive structure-based approaches. Although there have been attempts to utilize the stroke decomposition of Chinese characters [15], combined with vision transformer techniques for morphological imitation, such methods require a substantial reference character set or a highly complex character decomposition tree to generate new styles. They require extensive linguistic processing, significant storage consumption, and complex search processes, rendering them impractical for everyday use.

Consequently, we introduce MetaScript, a novel approach designed for generating a large number of Chinese characters in a consistent style using few-shot learning. Our method is trained on a handwritten dataset encompassing a variety of styles and a substantial quantity of Chinese characters. It is capable of producing high-quality stylistic imitations of a vast array of text, utilizing only a few style references and standard fonts as structural information. This approach effectively bridges the gap between the personalized nuances of handwriting and the efficiency of digital text generation.

Our work has a multitude of applications. For instance, it can serve as a straightforward alternative to traditional font generation methods, facilitating the effortless creation of personalized fonts. Our approach is capable of producing unique glyph designs, which can be instrumental in artistic and visual media design. With its low computational demands and real-time inference capabilities, our work can be integrated with large language models to generate responses in personalized fonts, thereby enriching the user experience.

We summarize our contributions in three key aspects:

- **Innovative Few-Shot Learning Model:** MetaScript employs a few-shot learning framework that enables the model to learn and replicate a specific handwriting style from a minimal set of examples. This significantly reduces the need for extensive training data, making the system more efficient and adaptable to individual styles.
- **Integration of Structural and Stylistic Elements:** Our approach uniquely combines the structural integrity of standard Chinese fonts with the stylistic elements of individual handwriting using structure and style encoders. This integration ensures that the generated characters are not only stylistically consistent but also maintain the legibility and structural accuracy essential for Chinese script.
- **Scalability and Efficiency:** The MetaScript system is designed to be scalable, capable of handling the generation of a vast number of characters without a proportional increase in computational resources or storage. This scalability is crucial given the extensive number of ideograms in the Chinese language and is a significant advancement over previous methods that required substantial storage and processing power.

2. Related Work

2.1. Generative Adversarial Networks

Generative Adversarial Networks (GANs) have been a hot research topic in the past 10 years [14]. For example, there are more than 22,900 papers related to GAN in 2023, that is: more than 2.5 papers per hour. GANs involve two parts: a generator that creates data and a discriminator that distinguishes between generated and real data. They are trained through a minimax optimization to reach a Nash equilibrium [35], where the generator effectively replicates the real data distribution. Thanks to the numerous works to enhance the objective function [20, 32], structure [7, 11, 22, 30], and transfer learning ability [22, 32, 45], of GANs, GANs now have a multitude of applications across various domains, such as Super-resolution [6, 25, 41, 44], Image synthesis and manipulation [8, 39], detection and video processing and NLP tasks [34, 43].

2.2. English Handwriting Generation

The generation of handwritten text represents a historically longstanding and classic task [33]. Early in 2007, Gangadhar et al. attempted to use Oscillatory neural networks to generate handwriting [10]. Some works used deep recurrent neural networks [2, 12, 24] to ensure enhanced consistency in the generated outcomes is imperative. Kanda et al. [21] proposes the use of reinforcement learning to evolve a rigorous future evaluation in traning. Other works imployed GANs to perform this task [1, 9, 31].

2.3. CJK Character Generation

Unlike some alphabetic languages that can generate extensive text using a limited number of templates, CJK (Chinese, Japanese, and Korean) languages are characterized by their abundant and structurally complex characters, precluding the possibility of generation through a minimal set of templates. The task of CJK (especially Chinese) character generation can be traced back to a period when computers had not yet become widespread in China [5]. Early methods decomposed characters into components or strokes [42, 46]. Later studies applied deep learning. Some of them used GANs or similar structure to transfer certain style onto stereotypes [3, 4, 19, 27]. Later advancement used Diffusion models to generate font sets [13, 15, 28].

3. Method

3.1. Overall Pipeline

Suppose the dataset $\mathcal{D}$ contains n types of Chinese characters and m writers totally. Let x_i^j denote the character with the i-th type written by the j-th writer, i.e. the script, where $i \in \{1, 2, \ldots, n\}$ and $j \in \{1, 2, \ldots, m\}$. Specifically, let x_i^0 denote the character with the i-th type rendered from a standard font, i.e. the prototype. The proposed character generator G is trained to follow the style of the references while keeping the structure of the templates. To be exact, given some references $\{x_k^j\}_{k=1}^c$ and a template x_i^0, the generated result $G(\{x_k^j\}_{k=1}^c, x_i^0)$ should be similar to x_i^j, which inherits the structure of the i-th type and the style of the j-th writer. We expect the character generator G can generalize well to the unseen references.

We utilize the adversarial learning paradigm to train the character generator G. Specifically, we introduce a multi-scale discriminator D to distinguish the generated character $G(\{x_k^j\}_{k=1}^c, x_i^0)$ from the real character x_i^j. The discriminator D is also trained to predict the type and writer of the given character x. We expect that the discriminator D can encourage the generator G to learn how to generate plausible characters.

Based on the character generator G, we can build a Chinese content generating system S. Given some style references $\{x_k\}_{k=1}^c$ and a content text $\{w_i\}_{i=1}^l$, the system S retrieves the corresponding templates $\{x_{w_i}^0\}_{i=1}^l$, generates the target characters $\{G(\{x_k\}_{k=1}^c, x_{w_i}^0)\}_{i=1}^l$, and composes them into the complete result. Such pipeline is defined as few-shot handwritten Chinese character generation.

3.2. Character Generator

Inspired by previous generative works [22, 23, 26], our proposed character generator G mainly contains three modules: a structure encoder E_α, a style encoder E_β, and a denormalization decoder D_γ. The structure encoder E_α extracts the structure information $\alpha_1, \alpha_2, \ldots, \alpha_7$ from the template x_i^0. The style encoder E_β extracts the style information β from the references $\{x_k^j\}_{k=1}^c$. Then the denormalization decoder D_γ combines the structure and style information to generate the target character x_i^j. The overview of the proposed character generator is shown in Figure 2. The loss function will be described in Equation 11 and 12.

Structure Encoder. The structure encoder E_α applies the U-Net [36] architecture, which includes 6 down-sampling blocks and 6 up-sampling blocks, extracting 7 feature maps $\alpha_1, \alpha_2, \ldots, \alpha_7$ with different scales from the template x_i^0, as shown in the blue part of Figure 2.

$$E_\alpha(x_i^0) = \{\alpha_1, \alpha_2, \ldots, \alpha_7\}. \tag{1}$$

Each block is composes of a 4×4 convolution layer with stride 2, a normalization layer, and an activation layer. There are skip connections between feature maps with the same scale. The structure encoder E_α is trained in a self-supervised manner, which expects the structure of the generated character $G(\{x_k^j\}_{k=1}^c, x_i^0)$ to be the same with that of the template x_i^0. The loss function will be described in Equation 17.

Style Encoder. The style information should be as concise as a dense feature vector. Therefore, we apply the ResNet-18 [16] architecture in the style encoder E_β with input channels modified to c and a linear layer added at the end. The style encoder E_β extracts a 512-dimensional feature vector β from the references $\{x_k^j\}_{k=1}^c$ as the style information, as shown in the orange part of Figure 2.

$$E_\beta(\{x_k^j\}_{k=1}^c) = \beta. \tag{2}$$

Similar to the structure encoder E_α, the style encoder E_β is trained in a self-supervised manner, which expects the style of the generated character $G(\{x_k^j\}_{k=1}^c, x_i^0)$ to be the same with that of the references $\{x_k^j\}_{k=1}^c$. The loss function will be described in Equation 18.

Denormalization Decoder. Both the structure information $\alpha_1, \alpha_2, \ldots, \alpha_7$ and the style information β will be fed into the denormalization decoder D_γ, which is composed of 7 cascaded denormalization blocks $D_\gamma^1, D_\gamma^2, \ldots, D_\gamma^7$, as shown in the green part of Figure 2.

$$D_\gamma(\{\alpha_i\}_{i=1}^7, \beta) = G(\{x_k^j\}_{k=1}^c, x_i^0). \tag{3}$$

The output of the denormalization decoder D_γ is the generated character $G(\{x_k^j\}_{k=1}^c, x_i^0)$, which will be directly supervised by the ground truth x_i^j.

The detailed structure of the denormalization block is shown in Figure 3. The denormalization block is composed of two identical layers, each of which contains a denormalization layer, an activation layer and a 3×3 convolution layer. There is also a skip connection between the input and output of the denormalization block, which follows the classical residual learning strategy [16].

$$D_\gamma^i(\alpha_i, \beta, \gamma_{i-1}) = \gamma_i, \tag{4}$$

where $i \in \{1, 2, \ldots, 7\}$. Specifically, we state that $\gamma_0 = \beta$ and $\gamma_7 = G(\{x_k^j\}_{k=1}^c, x_i^0)$ for simplicity.

The detailed structure of the denormalization layer is also shown in Figure 3. The denormalization layer follows the design of Adaptive Instance Normalization (AdaIN) [22], which is composed of a normalization step and a denormalization step. Inspired by [26], we also introduce an attention mechanism to fuse different feature maps softly. To be exact, the input feature map γ is first normalized in the channel dimension.

$$\bar{\gamma} = \frac{\gamma - \mu_\gamma}{\sigma_\gamma}, \tag{5}$$

where μ_γ and σ_γ are the mean and standard deviation of the input feature map γ respectively.

Then the structure feature map α will be fed into a 1×1 convolution layer to predict the mean μ_α and standard deviation σ_α of the normalized feature map $\bar{\gamma}$ for warping.

$$\hat{\alpha} = \sigma_\alpha \times \bar{\gamma} + \mu_\alpha, \tag{6}$$

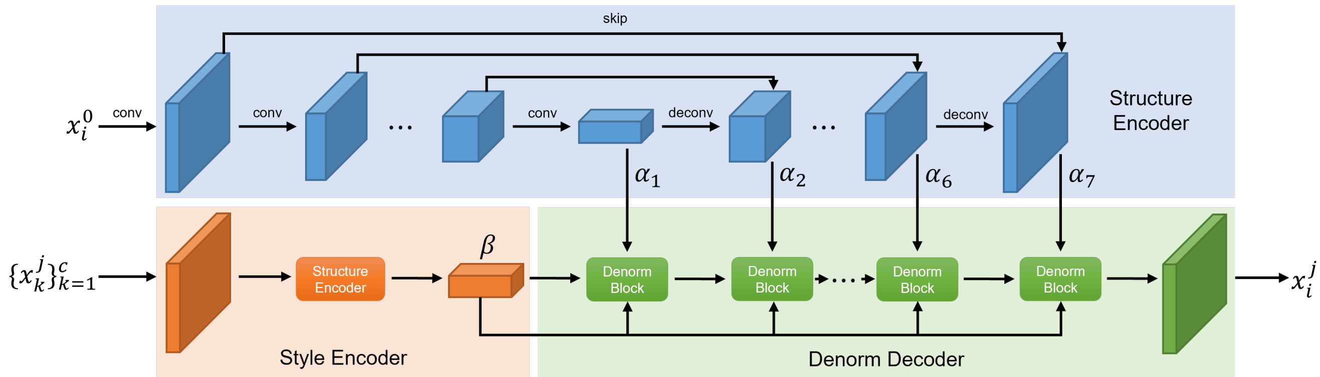

Figure 2. The overview of the proposed character generator G. The generator G mainly contains three modules: a structure encoder E_α, a style encoder E_β, and a denormalization decoder D_γ. The structure encoder E_α applies the U-Net architecture. The style encoder E_β applies the ResNet-18 architecture. The denormalization decoder D_γ is composed of 7 cascaded denormalization blocks.

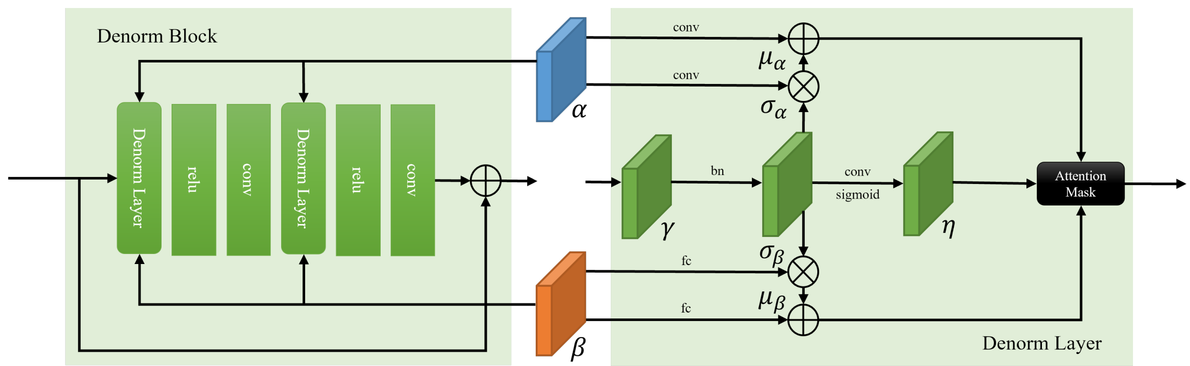

Figure 3. The detailed structure of the denormalization block and the denormalization layer. The denormalization layer is composed of a normalization step, a denormalization step and an attention mechanism. The denormalization block is composed of a skip connection and two identical layers, each of which contains a denormalization layer, an activation layer and a convolution layer.

where $\hat{\alpha}$ is the predicted structure feature map.

The style feature vector β will be fed into a linear layer to predict the mean μ_β and standard deviation σ_β of the normalized feature map $\bar{\gamma}$ for warping.

$$\hat{\beta} = \sigma_\beta \times \bar{\gamma} + \mu_\beta, \tag{7}$$

where $\hat{\beta}$ is the predicted style feature map.

In addition, the normalized feature map $\bar{\gamma}$ will be fed into a 1×1 convolution layer and a sigmoid activation layer to form the attention map η, which is used as a weighted mask to fuse the feature maps $\hat{\alpha}$ and $\hat{\beta}$.

$$\hat{\gamma} = (1 - \eta) \times \hat{\alpha} + \eta \times \hat{\beta}, \tag{8}$$

where $\hat{\gamma}$ is the output feature map. Finally, the denormalization layer completes the entire process.

The key idea of the denormalization layer is to adaptively adjust the effective regions of the structure feature map and the style feature map, so that the generated character can inherit the structure of the template and the style of the references. Compared with the AdaIN [22], the denormalization layer can fuse the feature maps of arbitrary styles instead of pairwise exchanging, which shows better flexibility and diversity in character generation.

3.3. Multi-scale Discriminator

The discriminator block D^i follows the traditional convolutional neuron network paradigm, which is composed of 5 down-sampling blocks and 3 classification heads, as shown in Figure 4. Each down-sampling block is composed of a 4×4 convolution layer with stride 2, a normalization layer, and an activation layer, which is exactly the same as that of the structure encoder E_α. Each classification head contains a single linear layer to predict the probability distribution from the extracted feature map.

$$D^i(x) = (D^i_\phi(x), D^i_\alpha(x), D^i_\beta(x)) = (\hat{y}^i_\phi, \hat{y}^i_\alpha, \hat{y}^i_\beta), \tag{9}$$

where D^i represents the i-th discriminator block, $\hat{y}^i_\phi$ represents the predicted authenticity, $\hat{y}^i_\alpha$ represents the predicted type, and $\hat{y}^i_\beta$ represents the predicted writer. We expect that the discriminator block can extract useful features to distinguish the authenticity, type and writer all together. The fundamental idea is to force the generator to learn the correct structure and style features, instead of simply cheating the discriminator. The loss function will be described in Equation 13, 14, 15 and 16.

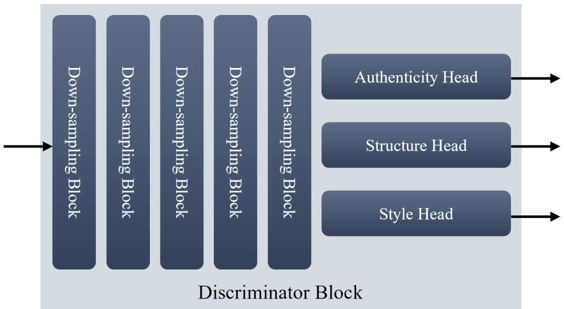

Figure 4. The structure of the discriminator block. The discriminator block is composed of 5 down-sampling blocks and 3 classification heads as a typical convolutional neuron network.

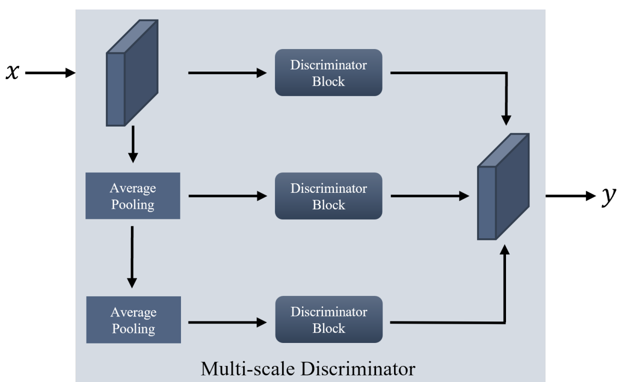

Figure 5. The overview of the multi-scale discriminator D. The discriminator D is composed of 2 average pooling layers and 3 discriminator blocks D^1, D^2 and D^3. The input character x will be down-sampled by the average pooling layers to form 3 different scales and fed into the corresponding discriminator blocks.

Inspired by [40], we apply a multi-scale discriminator D to enhance the performance of the discriminator, as shown in Figure 5. The multi-scale discriminator D is composed of 2 average pooling layers and 3 discriminator blocks D^1, D^2 and D^3. The input character x will be down-sampled by the average pooling layers to form 3 different scales, so the corresponding discriminator blocks can evaluate the input character x from different perspectives and perform better supervision. Previous works [26, 40] have shown that the multi-scale discriminator can effectively improve the quality of the generated results, especially in the high-resolution tasks.

$$D(x) = \{D^1(x), D^2(x'), D^3(x'')\}, \tag{10}$$

where x' and x'' represent the input character x down-sampled once and twice respectively. The loss function will be described in Equation 11 and 12.

3.4. Training Objective

We utilize adversarial learning to train the character generator G and the multi-scale discriminator D and introduce 5 kinds of loss functions: adversarial loss $\mathcal{L}_{adv}$, classification loss $\mathcal{L}_{cls}$, structure loss $\mathcal{L}_{str}$, style loss $\mathcal{L}_{sty}$ and reconstruction loss $\mathcal{L}_{rec}$. The overall loss function is a weighted sum over them. Formally, for the generator, the overall loss function is defined as

$$\begin{aligned} \mathcal{L}_{all}^G = &\lambda_{adv}^G \mathcal{L}_{adv}^G + \lambda_{cls}^G \mathcal{L}_{cls}^G \\ &+ \lambda_{str}^G \mathcal{L}_{str}^G + \lambda_{sty}^G \mathcal{L}_{sty}^G + \lambda_{rec}^G \mathcal{L}_{rec}^G, \end{aligned} \tag{11}$$

and for the discriminator it is defined as

$$\mathcal{L}_{all}^D = \lambda_{adv}^D \mathcal{L}_{adv}^D + \lambda_{cls}^D \mathcal{L}_{cls}^D, \tag{12}$$

where λ_{adv}^G, λ_{cls}^G, λ_{str}^G, λ_{sty}^G, λ_{rec}^G, λ_{adv}^D and λ_{cls}^D are the hyperparameters to balance the loss functions.

Adversarial Loss. The adversarial loss is to train the discriminator D to distinguish the generated character $G(\{x_k^j\}_{k=1}^c, x_i^0)$ from the real character x_i^j, which indirectly encourages the generator G to generate more plausible characters. Binary cross entropy is applied as the adversarial loss. Formally, for the generator, the adversarial loss is defined as

$$\mathcal{L}_{adv}^G = -\sum_{s=1}^{3} \log D_\phi^s(G(\{x_k^j\}_{k=1}^c, x_i^0)), \tag{13}$$

and for the discriminator it is defined as

$$\begin{aligned} \mathcal{L}_{adv}^D = &-\sum_{s=1}^{3} \log D_\phi^s(x_i^j) \\ &-\sum_{s=1}^{3} \log[1 - D_\phi^s(G(\{x_k^j\}_{k=1}^c, x_i^0))]. \end{aligned} \tag{14}$$

Classification Loss. The classification loss is to train the discriminator D to precisely predict the type and writer of the given character x. Different from the adversarial loss, the generator G is also trained to minimize the classification loss, which indirectly encourages the generator G to generate characters with accurate structure and style. Cross entropy is applied as the classification loss. Formally, for the generator, the classification loss is defined as

$$\begin{aligned} \mathcal{L}_{cls}^G = &-\sum_{s=1}^{3} \log D_\alpha^s(G(\{x_k^j\}_{k=1}^c, x_i^0)) \\ &-\sum_{s=1}^{3} \log D_\beta^s(G(\{x_k^j\}_{k=1}^c, x_i^0)), \end{aligned} \tag{15}$$

and for the discriminator it is defined as

$$\begin{aligned}\mathcal{L}_{cls}^{D} = & -\sum_{s=1}^{3} \log D_{\alpha}^{s}(x_i^j) - \sum_{s=1}^{3} \log D_{\beta}^{s}(x_i^j) \\ & -\sum_{s=1}^{3} \log[D_{\alpha}^{s}(G(\{x_k^j\}_{k=1}^{c}, x_i^0))] \\ & -\sum_{s=1}^{3} \log[D_{\beta}^{s}(G(\{x_k^j\}_{k=1}^{c}, x_i^0))].\end{aligned} \tag{16}$$

We should note that the classification loss is indispensable, which introduces effective supervision to prevent the generator from simply generating meaningless characters to cheat the discriminator. We will show how the classification loss solves the problem of mode collapse in Section 4.

Structure Loss. Intuitively, we expect that the generated character $G(\{x_k^j\}_{k=1}^{c}, x_i^0)$ can inherit the structure of the template x_i^0. Therefore, The feature maps $\alpha_1, \alpha_2, \ldots, \alpha_7$ extracted by the structure encoder E_α should be invariant. The structure loss not only encourages the generator G to generate the correct structure, but also encourages the structure encoder E_α to extract valid structure features. The structure loss for the generator is formally defined as

$$\mathcal{L}_{str}^{G} = \frac{1}{2}\|E_\alpha(G(\{x_k^j\}_{k=1}^{c}, x_i^0)) - E_\alpha(x_i^0)\|_2^2. \tag{17}$$

Style Loss. We also expect that the generated character $G(\{x_k^j\}_{k=1}^{c}, x_i^0)$ can follow the style of the references $\{x_k^j\}_{k=1}^{c}$. Therefore, the feature vector β extracted by the style encoder E_β should be invariant. The style loss not only encourages the generator G to generate the correct style, but also encourages the style encoder E_β to extract valid style features. The style loss for the generator is formally defined as

$$\mathcal{L}_{sty}^{G} = \frac{1}{2}\|E_\beta(G(\{x_k^j\}_{k=1}^{c}, x_i^0)) - E_\beta(\{x_k^j\}_{k=1}^{c})\|_2^2. \tag{18}$$

Reconstruction Loss. The reconstruction loss represents the pixel-wise difference between the generated character $G(\{x_k^j\}_{k=1}^{c}, x_i^0)$ and the ground truth x_i^j, which is a direct supervision to the generator G. L_1 norm is applied as the reconstruction loss. The reconstruction loss for the generator is formally defined as

$$\mathcal{L}_{rec}^{G} = \|G(\{x_k^j\}_{k=1}^{c}, x_i^0) - x_i^j\|_1. \tag{19}$$

We should note that the reconstruction loss is not to force the generated character to be exactly the same with the ground truth because such constraint will limit the diversity of the generated characters. Instead, the reconstruction loss should be controlled to a reasonable range.

3.5. Content Composition

We develop a typesetting tool, Typewriter, for reorganizing output characters into a complete result. For a given text and some style references, we generate an individual character for each character in the input text using the methods described in the preceding subsections. To better and more intuitively display the effectiveness of our method in generating Chinese character content, we first apply a random transformation to the generated character images, mimicking the alignment effect of human handwriting. Subsequently, they are arranged into a complete image. The details are shown in Algorithm 1.

Algorithm 1: Typewriter Procedure

Input : style references $\{x_k\}_{k=1}^{c}$, content text $\{w_i\}_{i=1}^{l}$, expected character size $size_c$, expected line width $width_l$
Output : arranged handwritten content image I_a

Initialize: $cursor \leftarrow$ position 0, $I_a \leftarrow$ empty image;
for w_i *in* $\{w_i\}_{i=1}^{l}$ **do**
  if w_i *is line break* **then**
    move $cursor$ to next line;
    continue;
  else if w_i *is space* **then**
    $g_i \leftarrow$ empty space with $\frac{size_c}{2}$ width;
  else if w_i *is character* **then**
    generate $g_i \leftarrow G(\{x_k\}_{k=1}^{c}, x_{w_i}^0)$ with $size_c$;
  else if w_i *is punctuation* **then**
    $g_i \leftarrow$ punctuation template;
  end
  apply random transformation on g_i;
  plot g_i at $cursor$ in I_a and update $cursor$;
  if $cursor > width_l$ **then**
    move $cursor$ to next line;
  end
end
return handwritten content image I_a

4. Experiments

4.1. Experiment Setup

We finetune the hyperparameters, train the model and build the system. Some results are shown in Figure 6 and 7. The training details are as follows.

Dataset. The training set is build based on the CASIA-HWDB-1.1 dataset [29]. We select approximately 1M script images from 300 writers, including 3755 common Chinese characters. By preprocessing, the script images are resized into 128×128 resolution, named by the types and classified by the writers, which will be used as style references. We also render the prototype images of the characters with the same resolution from the Source Han Sans CN

Figure 6. Characters synthesized by the generator with 4 references trained for 100k iterations. The first 4 rows are the style references. The 5th row is the structure template. The 6th row is the ground truth. The last row is the generated character.

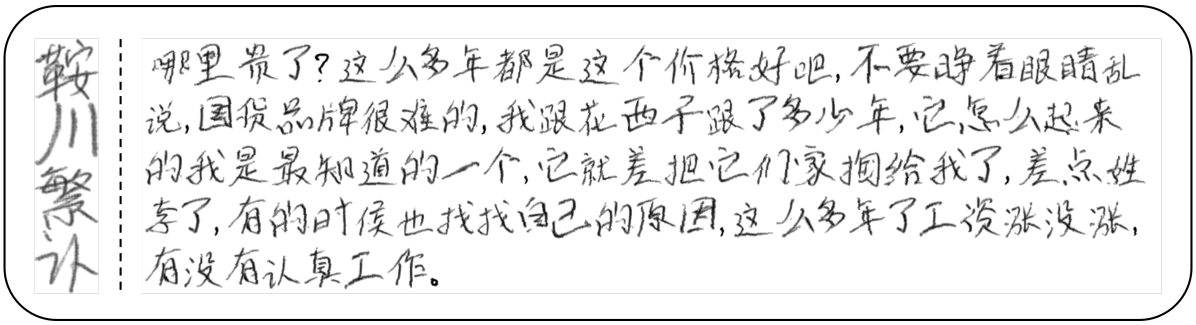

Figure 7. Content generated by the system, which is built based on the generator and cascaded with the type writter. The left hand side shows the style references. The right hand side shows the generated results.

font, which will be used as structure templates.

Implementation details. The model is trained for 100k iterations with a batch size of 32. We use the Adam optimizer for training. The learning rate is set to 0.0001 for both the generator and the discriminator. The weights of the loss function are $\lambda^G_{\text{adv}} = 1$, $\lambda^G_{\text{cls}} = 1$, $\lambda^G_{\text{str}} = 0.5$, $\lambda^G_{\text{sty}} = 0.1$, $\lambda^G_{\text{rec}} = 20$, $\lambda^D_{\text{adv}} = 1$ and $\lambda^D_{\text{cls}} = 1$.

Evaluation metrics. To evaluate the quality and diversity of the generated results, we apply recognition accuracy (RA), inception score (IS) [37] and Frechet inception distance (FID) [17] as the evaluation metrics. We use an EfficientNetV2 model [38] pretrained on the CASIA-HWDB-1.1 dataset as the classifier to extract high-level features and perform low-level predictions for computation. The test set is manually selected from the CASIA-HWDB-1.0 dataset, which contains 1000 characters written by 10 writers.

4.2. Ablation Studies

We conduct two separate ablation studies to investigate the best number of references and verify the effectiveness of each loss term.

Reference number. To figure out how the number of references affects the performance of the model, we train the model with 1, 2, 4 and 8 references for 100k iterations respectively. The results are shown in Table 1.

We can see that in general, as the number of references increases, the inception score increases and the Frechet inception distance decreases, which indicates better authenticity and diversity of the generated results. This is reasonable because more references provide more sufficient style information for the generator. However, the recognition accuracy shows strong fluctuations and achieves the best performance with 4 references. Therefore, we prefer to use 4 references in practice.

Table 1. Different reference numbers exhibit variations in model performance, including recognition accuracy (RA), inception score (IS) and Frechet inception distance (FID).

Reference	RA↑	IS↑	FID↓
1	79.8%	55.543	197.950
2	77.5%	58.524	190.193
4	**81.9%**	58.172	**187.365**
8	76.2%	**58.598**	188.411

Loss function. Another question is that whether each loss term is necessary for training the model. To answer this question, we train the model with 4 references for 100k iterations, but each loss term is removed respectively. The results are shown in Table 2.

The models with one of the loss terms removed show performance degradation to varying degrees except for $\mathcal{L}_{str}$, which has been implicitly covered by the strong representation ability of the structure encoder. Removing $\mathcal{L}_{sty}$

Table 2. Changes in model performance with different loss terms removed, including adversarial loss ($\mathcal{L}_{adv}$), classification loss ($\mathcal{L}_{cls}$), structure loss ($\mathcal{L}_{str}$), style loss ($\mathcal{L}_{sty}$) and reconstruction loss ($\mathcal{L}_{rec}$), compared to the baseline model ($\mathcal{L}_{all}$).

Loss	RA↑	IS↑	FID↓
w/ $\mathcal{L}_{all}$	81.9%	58.172	187.365
w/o $\mathcal{L}_{adv}$	0.0%	1.098	287.104
w/o $\mathcal{L}_{cls}$	0.1%	1.601	318.286
w/o $\mathcal{L}_{str}$	84.4%	57.578	181.738
w/o $\mathcal{L}_{sty}$	67.4%	55.108	204.016
w/o $\mathcal{L}_{rec}$	72.3%	54.316	202.782

Figure 8. Examples of the generated results with $\mathcal{L}_{adv}$ removed.

Figure 9. Examples of the generated results with $\mathcal{L}_{cls}$ removed.

and $\mathcal{L}_{rec}$ leads to a modest decrease in performance, while removing $\mathcal{L}_{adv}$ and $\mathcal{L}_{cls}$ results in a complete failure. We show some examples of the generated results with $\mathcal{L}_{adv}$ and $\mathcal{L}_{cls}$ removed in Figure 8 and 9 respectively. We can see that the generator cannot learn the handwritten style without adversarial learning, and fails to generate the correct character pattern with classification loss removed, which can even lead to serious mode collapse. Therefore, we claim that all the loss terms are necessary.

5. Conclusion

In this paper, we introduced MetaScript, a novel system for generating handwritten Chinese content using few-shot learning and Generative Adversarial Networks. Our approach effectively bridges the gap between the personalized nuances of handwriting and the efficiency of digital text generation. The key contributions of our work include the development of an innovative few-shot learning model, the integration of structural and stylistic elements in character generation, and the scalability and efficiency of the MetaScript system.

Our experiments demonstrate that MetaScript can successfully replicate a variety of handwriting styles with high fidelity using only a few style references. The system shows promising results in terms of recognition accuracy, inception score, and Frechet inception distance, indicating its effectiveness in generating authentic and diverse handwritten Chinese characters.

However, there are still challenges and limitations to be addressed. The quality of generated characters can vary depending on the number and quality of style references provided. Additionally, while our model performs well with common Chinese characters, its effectiveness with less common or more complex characters requires further exploration.

Future work will focus on enhancing the robustness and versatility of MetaScript: 1) We aim to enhance the robustness and versatility of the system, focusing on more sophisticated few-shot learning techniques. This enhancement is expected to significantly improve MetaScript's ability to learn effectively from limited data. 2) Another pivotal area of interest is the extension of our approach to non-Latin scripts, including Arabic and Devanagari. These scripts, with their rich handwriting traditions, present unique challenges and opportunities for our handwriting generation model. 3) Finally, we plan to integrate MetaScript into real-world applications. This integration involves embedding our system into digital education tools and personalized digital communication platforms, thereby infusing the warmth and personality of traditional handwriting into the digital realm.

References

[1] Eloi Alonso, Bastien Moysset, and Ronaldo Messina. Adversarial generation of handwritten text images conditioned on sequences. In *2019 international conference on document analysis and recognition (ICDAR)*, pages 481–486. IEEE, 2019. 2

[2] Suraj Bodapati, Sneha Reddy, and Sugamya Katta. Realistic handwriting generation using recurrent neural networks and long short-term networks. In *Proceedings of the Third International Conference on Computational Intelligence and Informatics: ICCII 2018*, pages 651–661. Springer, 2020. 2

[3] Bo Chang, Qiong Zhang, Shenyi Pan, and Lili Meng. Generating handwritten chinese characters using cyclegan. In *2018 IEEE winter conference on applications of computer vision (WACV)*, pages 199–207. IEEE, 2018. 2

[4] Jie Chang, Yujun Gu, Ya Zhang, Yan-Feng Wang, and CM Innovation. Chinese handwriting imitation with hierarchical generative adversarial network. In *BMVC*, page 290, 2018. 2

[5] Shi-Kuo Chang. An interactive system for chinese character generation and retrieval. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-3(3):257–265, 1973. 2

[6] Zihan Ding, Xiao-Yang Liu, Miao Yin, and Linghe Kong. Tgan: Deep tensor generative adversarial nets for large image generation. *arXiv preprint arXiv:1901.09953*, 2019. 2

[7] Ishan Durugkar, Ian Gemp, and Sridhar Mahadevan. Generative multi-adversarial networks. *arXiv preprint arXiv:1611.01673*, 2016. 2

[8] Kiana Ehsani, Roozbeh Mottaghi, and Ali Farhadi. Segan: Segmenting and generating the invisible. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6144–6153, 2018. 2

[9] Sharon Fogel, Hadar Averbuch-Elor, Sarel Cohen, Shai Mazor, and Roee Litman. Scrabblegan: Semi-supervised varying length handwritten text generation, 2020. 2

[10] Garipelli Gangadhar, Denny Joseph, and V Srinivasa Chakravarthy. An oscillatory neuromotor model of handwriting generation. *International journal of document analysis and recognition (ijdar)*, 10:69–84, 2007. 2

[11] Arnab Ghosh, Viveka Kulharia, Vinay P Namboodiri, Philip HS Torr, and Puneet K Dokania. Multi-agent diverse generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8513–8521, 2018. 2

[12] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013. 2

[13] Dongnan Gui, Kai Chen, Haisong Ding, and Qiang Huo. Zero-shot generation of training data with denoising diffusion probabilistic model for handwritten chinese character recognition. *arXiv preprint arXiv:2305.15660*, 2023. 2

[14] Jie Gui, Zhenan Sun, Yonggang Wen, Dacheng Tao, and Jieping Ye. A review on generative adversarial networks: Algorithms, theory, and applications, 2020. 2

[15] Haibin He, Xinyuan Chen, Chaoyue Wang, Juhua Liu, Bo Du, Dacheng Tao, and Yu Qiao. Diff-font: Diffusion model for robust one-shot font generation. *arXiv preprint arXiv:2212.05895*, 2022. 2

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 3

[17] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017. 7

[18] Yu Ji, Wen Wu, Yi Hu, Xiaofeng He, Changzhi Chen, and Liang He. Automatic personality prediction based on users' chinese handwriting change. In *CCF Conference on Computer Supported Cooperative Work and Social Computing*, pages 435–449. Springer, 2022. 1

[19] Haochuan Jiang, Guanyu Yang, Kaizhu Huang, and Rui Zhang. W-net: One-shot arbitrary-style chinese character generation with deep neural networks. In *Neural Information Processing*, pages 483–493, Cham, 2018. Springer International Publishing. 2

[20] Alexia Jolicoeur-Martineau. The relativistic discriminator: a key element missing from standard gan. *arXiv preprint arXiv:1807.00734*, 2018. 2

[21] Keisuke Kanda, Brian Kenji Iwana, and Seiichi Uchida. What is the reward for handwriting? — a handwriting generation model based on imitation learning. In *2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 109–114, 2020. 2

[22] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019. 2, 3, 4

[23] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8110–8119, 2020. 3

[24] K. Manoj Kumar, Harish Kandala, and N. Sudhakar Reddy. Synthesizing and imitating handwriting using deep recurrent neural networks and mixture density networks. In *2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pages 1–6, 2018. 2

[25] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017. 2

[26] Lingzhi Li, Jianmin Bao, Hao Yang, Dong Chen, and Fang Wen. Faceshifter: Towards high fidelity and occlusion aware face swapping. *arXiv preprint arXiv:1912.13457*, 2019. 3, 5

[27] Meng Li, Jian Wang, Yi Yang, Weixing Huang, and Wenjuan Du. Improving gan-based calligraphy character generation using graph matching. In *2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pages 291–295. IEEE, 2019. 2

[28] Qisheng Liao, Gus Xia, and Zhinuo Wang. Calliffusion: Chinese calligraphy generation and style transfer with diffusion modeling. *arXiv preprint arXiv:2305.19124*, 2023. 2

[29] Cheng-Lin Liu, Fei Yin, Da-Han Wang, and Qiu-Feng Wang. Casia online and offline chinese handwriting databases. In *2011 international conference on document analysis and recognition*, pages 37–41. IEEE, 2011. 6

[30] Ming-Yu Liu and Oncel Tuzel. Coupled generative adversarial networks. *Advances in neural information processing systems*, 29, 2016. 2

[31] Xiyan Liu, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. Handwritten text generation via disentangled representations. *IEEE Signal Processing Letters*, 28:1838–1842, 2021. 2

[32] Yongyi Lu, Yu-Wing Tai, and Chi-Keung Tang. Attribute-guided face generation using conditional cyclegan. In *Proceedings of the European conference on computer vision (ECCV)*, pages 282–297, 2018. 2

[33] Mehul Madaan, Aniket Kumar, Shubham Kumar, Aniket Saha, and Kirti Gupta. Handwriting generation and synthesis: A review. In *2022 Second International Conference on Power, Control and Computing Technologies (ICPC2T)*, pages 1–6, 2022. 2

[34] Olof Mogren. C-rnn-gan: Continuous recurrent neural networks with adversarial training. *arXiv preprint arXiv:1611.09904*, 2016. 2

[35] Lillian J Ratliff, Samuel A Burden, and S Shankar Sastry. Characterization and computation of local nash equilibria in continuous games. In *2013 51st Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 917–924. IEEE, 2013. 2

[36] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015. 3

[37] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016. 7

[38] Mingxing Tan and Quoc Le. Efficientnetv2: Smaller models and faster training. In *International conference on machine learning*, pages 10096–10106. PMLR, 2021. 7

[39] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. *Advances in neural information processing systems*, 29, 2016. 2

[40] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8798–8807, 2018. 5

[41] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy. Esrgan: Enhanced super-resolution generative adversarial networks. In *Proceedings of the European conference on computer vision (ECCV) workshops*, pages 0–0, 2018. 2

[42] Songhua Xu, Tao Jin, Hao Jiang, and Francis CM Lau. Automatic generation of personal chinese handwriting by capturing the characteristics of personal handwriting. In *Twenty-First IAAI Conference*, 2009. 2

[43] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of the AAAI conference on artificial intelligence*, 2017. 2

[44] Yuan Yuan, Siyuan Liu, Jiawei Zhang, Yongbing Zhang, Chao Dong, and Liang Lin. Unsupervised image super-resolution using cycle-in-cycle generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 701–710, 2018. 2

[45] Yiheng Zhang, Zhaofan Qiu, Ting Yao, Dong Liu, and Tao Mei. Fully convolutional adaptation networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6810–6818, 2018. 2

[46] Baoyao Zhou, Weihong Wang, and Zhanghui Chen. Easy generation of personal chinese handwritten fonts. In *2011 IEEE international conference on multimedia and expo*, pages 1–6. IEEE, 2011. 2