Handwritten Chinese Font Generation with Collaborative Stroke Refinement

Chuan Wen[1] Yujie Pan[1] Jie Chang[1] Ya Zhang[1] Siheng Chen[1]
Yanfeng Wang[1] Mei Han[2] Qi Tian[1]
[1]Cooperative Madianet Innovation Center, Shanghai Jiao Tong University
[2]PingAn Technology US Research Lab
{alvinwen, yujiepan, j_chang, ya_zhang, sihengc, wangyanfeng}@sjtu.edu.cn
hanmei613@pingan.com.cn wywqtian@gmail.com

Abstract

Automatic character generation is an appealing solution for typeface design, especially for Chinese fonts with over 3700 most commonly-used characters. This task is particularly challenging for handwritten characters with thin strokes which are error-prone during deformation. To handle the generation of thin strokes, we introduce an auxiliary branch for stroke refinement. The auxiliary branch is trained to generate the bold version of target characters which are then fed to the dominating branch to guide the stroke refinement. The two branches are jointly trained in a collaborative fashion. In addition, for practical use, it is desirable to train the character synthesis model with a small set of manually designed characters. Taking advantage of content-reuse phenomenon in Chinese characters, we further propose an online zoom-augmentation strategy to reduce the dependency on large size training sets. The proposed model is trained end-to-end and can be added on top of any method for font synthesis. Experimental results on handwritten font synthesis have shown that the proposed method significantly outperforms the state-of-the-art methods under practical setting, i.e. with only 750 *paired training samples.*

1. Introduction

The Chinese language consists of more than 8000 characters, among which about 3700 are frequently used. Designing a new Chinese font involves considerable tedious manual efforts. Given an initial set of manually designed samples, it is desirable to automatically complete the rest, i.e. *Chinese font synthesis*. Inspired by the recent progress of neural style transfer [6, 9, 10, 21, 25, 31], several studies have recently attempted to model font synthesis as an image-to-image translation problem [7, 23, 29].

Unlike neural style transfer, font transfer is low fault-

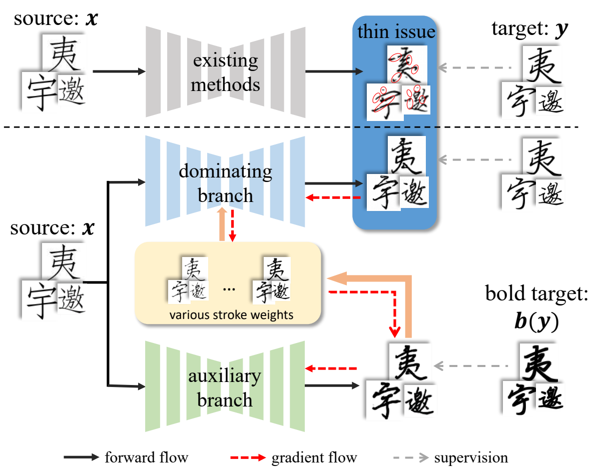


Figure 1. Collaborative stroke refinement handles the thin issue. Handwritten characters usually have various stroke weights. Thin strokes have lower fault tolerance in synthesis, leading to distortions. Collaborate stroke refinement adopts collaborative training. An auxiliary branch is introduced to capture various stroke weights, guiding the dominating branch to solve the thin issue.

tolerant because any misplacement of strokes may change the semantics of the characters [7]. So far, two main streams of approaches have been explored. One addresses the problem with bottleneck CNN structures [15, 16, 7, 18, 13] and the other proposes to disentangle representation of characters into *content* and *style* [22, 23, 29]. While promising results have been shown in font synthesis, most existing methods require an impracticable initial set of large size to train a model. For example, [15, 16, 7, 18] require 3000 paired characters for training supervision. Several recent methods [12, 23, 13] investigated learning with fewer paired characters by introducing more domain knowledge. However, they heavily depend on extra labels or pre-trained auxiliary networks to integrate the prior knowledge of radical-reuse or stroke-continuity. For example, hard codes of character structures are required in [23], and handmade stroke

Table 1. Comparison of our method with existing CNN-based *Chinese* font synthesis methods.

Methods	Pre-training set	Training size for target	Extra information
Rewrite [15]	None	3000	None
AEGN [18]			
HAN [7]			
zi2zi [16]	Hundreds of fonts (each involving 3000 samples)	3000	
EMD [29]			
SA-VAE [23]		1000	133-bit structure code for each character
DCFont [12]	20 fonts (each involving 2000 samples)	755	A feature reconstruction network pretrained on 100 fonts
SCFont [13]	25 fonts (each involving 6000 samples)	775	Human corrected stoke trajectories for all 25 $\times$ 6000 samples
Ours	15 fonts (each involving 3000 samples)	750	None

priors are needed in [13]. These extra labels or auxiliary pretrained modules require either expensive human labor or extremely large font library so it will be better if they can be implemented implicitly and automatically. Table 1 provides a comparative summary of the above methods.

Most of the above methods focused on printed font synthesis. Compared with printed typeface synthesis, *handwritten* font synthesis can be much more challenging. First, handwritten characters are usually associated with thin irregular strokes of little information. These thin strokes are prone to be missing or broken during deformation due to enormous blank pixels in font images (see Fig. 1). To solve the thin stroke issue, we thus propose *collaborative stroke refinement*. We design an auxiliary branch to synthesize thicker strokes, which are further used to guide the dominating branch to maintain stroke completeness. This is similar to the collaborative training strategy in that the dominating branch can get refinement with various stroke weights from the auxiliary branch, while the backward gradient from the dominating branch passes more accurate skeleton information to the auxiliary branch.

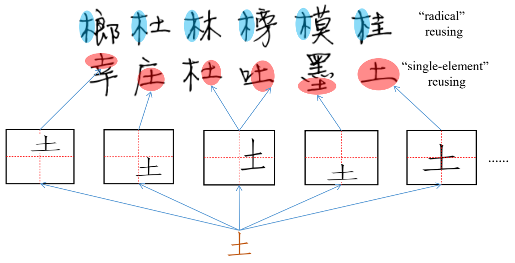


Figure 2. Online zoom-augmentation exploits the content-reuse phenomenon. A same radical can exist in various characters at various locations. Our method models the variety of locations and sizes from elementary characters without manually decomposition, significantly reducing the required quantity of training samples.

Furthermore, for practical use, the font synthesis method needs to generate thousands of characters based on only a few manually designed characters (see Fig. 2). To solve this issue, we exploit the content-reuse phenomenon in Chinese characters, i.e., the same radicals (elementary components) may present in various characters at various locations. *Online zoom-augmentation* is proposed to help the model to learn this phenomenon and to decouple the unseen characters into already learned parts. This significantly reduces the required size of training sets. And unlike SA-VAE [23], which requires additional radical embedding, it implicitly mimics the mapping between radicals at different positions. In addition, to make the networks easily capture the deformation from source to target, we further apply *adaptive pre-deformation*, which learns the size and scale to standardize characters. This allows the proposed networks to focus on learning high-level style deformation.

Combining the above techniques, we proposed an end-to-end trainable model to generate high-quality characters. We validate the effectiveness of the proposed method on several Chinese fonts including both *handwritten* and *printed* fonts. Experimental results have shown that 1) the proposed method is able to generate high quality characters with only a minimal set of 750 training samples on target fonts without explicitly using any extra prior knowledge; 2) Under the similar experimental settings and without human priors, the proposed method significantly outperforms the other state-of-the-art Chinese font generation methods.

The main contributions are summarized as follows.

- We propose an end-to-end trainable model to synthesize handwritten Chinese fonts under a practical setting of only 750 training samples;
- We propose collaborative stroke refinement, handling the thin stroke issue; online zoom-augmentation, enabling learning with fewer training samples; and adap-

tive pre-deformation, standardizing and aligning the characters;

- Experiments have shown that our method outperforms several state-of-the-arts in terms of visual effects, Mean Square Error (MSE) metric, and user metric.

2. Related Work

Most image-to-image translation tasks such as art-style transfer [8, 14], coloring [3, 28], super-resolution [17], dehazing [1] have achieved progressive improvement since the raise of CNNs. Recently, several works model font glyph synthesis as a supervised image-to-image translation task mapping the character from one typeface to another by CNN analogous to an encoder/decoder backbone [15, 16, 2, 22, 7, 12, 18, 5, 27, 13]. Specially, generative adversarial networks (GANs) [11, 19, 20] are widely adopted to obtain promising results. Differently, EMD [29] and SA-VAE [23] are proposed to disentangle the style/content representation of Chinese fonts for a more flexible transfer.

Preserving the content consistency with as small training set as possible is important in font synthesis task. Recently, "From A to Z" [24], MC-GAN [2] and [22] perform pretty well in synthesizing ornamented glyphs from a few examples. However, the generation of Chinese characters is very different from the English alphabet no matter in complexity or considerations. For Chinese font generation, the requirements of content accuracy and trainset size are more stringent, so the related works usually have more complex frameworks. To synthesize multiple fonts, zi2zi [16] utilized a one-hot category embedding, but it requires a very large font library during both pretraining and finetuning process. EMD [29] is the first model to achieve good performance on new Chinese font transfer with a few samples, but it does work poorly on handwritten fonts.

To further reduce the amount of training data, some recent works explored integrating more domain knowledge into CNN. SA-VAE [23] embedded 133-bits coding denoting the structure/content-reuse knowledge of inputs, which must rely on the extra labeling related to structure and content beforehand. In addition, it leverages a pre-trained Chinese character recognition network [26, 30] to provide the content label. Similarly, DCFont [12] employs a pre-trained 100-class font-classifier to provide a better style representation. SCFont [13] proposes a two-stage method to learn the skeleton and stroke style separately, which requires expensive annotations of stroke priors on all characters for each font and limits its practical use. In addition, DM-Font [4] explores the compositionality of Korean characters with 68 components, but is hard to be generalized on Chinese fonts which contain far more sub-glyphs.

3. Methodology

Given a source character, handwritten Chinese font generation aims to generate the same character in the target style. Let $\mathcal{C} = \{c_i\}_{i=0}^m$ be a set of m images for standardized Chinese characters, where each element c_i represents a single Chinese character. Let the source set $\mathcal{X} = \{x_i = d_{\text{source}}(c_i)|c_i \in \mathcal{C}\}_i$ and the target set $\mathcal{Y} = \{y_i = d_{\text{target}}(c_i)|c_i \in \mathcal{C}\}_i$ be two training image sets representing the characters $\mathcal{C}$ in two styles, where $d_{\text{source}}(\cdot)$ and $d_{\text{target}}(\cdot)$ denote the deformation function in the source and target styles, respectively. The model is trained with the source-target pairs, $\mathcal{X}$ and $\mathcal{Y}$. At testing phase, given an input image, $x = d_{\text{source}}(c)$, the model outputs a synthesis image, $y = d_{\text{target}}(c)$. Since that we are blind to both deformation functions, the key of this task is to learn a mapping from $d_{\text{source}}(\cdot)$ to $d_{\text{target}}(\cdot)$ based on training sets $\mathcal{X}$ and $\mathcal{Y}$. Here we use deep neural networks to learn the mapping from $d_{\text{source}}(\cdot)$ to $d_{\text{target}}(\cdot)$. To make it practical, we want the size of the training set m as small as possible.

Fig. 3 shows the proposed networks, including a *coarse generator* to generate low resolution feature maps and a *collaborative stroke refinement* to generate the final results.

3.1. Collaborative Stroke Refinement

The proposed collaborative stroke refinement, based on multi-task learning and collaborative training strategy, includes *dominating branch* and *auxiliary branch*, with a *refinement module* connecting the two branches. The auxiliary branch learns stroke deformations between the source x and a bold target $b(y)$, where $b(\cdot)$ is a bolding function; the refinement module merges information from the auxiliary branch to the dominating branch; and finally, the dominating branch learns the stroke deformation between the source x and the original target y with auxiliary information from the refinement module. The two branches and the refinement module are trained simultaneously.

Auxiliary branch. The auxiliary branch synthesizes $\hat{b(y)}$ to approximate the bold target $b(y)$ based on the source x. The bold target is obtained from original target by applying the morphological dilation operation:

$$b(y) = y \oplus e = \{z|(\hat{e})_z \cap y \neq \varnothing\},$$

where e is the structuring element, $\hat{*}$ denotes reflecting all elements of $*$ about the origin of this set, $(*)_z$ denotes translating the origin of $*$ to point z. The synthesis $\hat{b(y)}$ has bold strokes and are robust to potential distortion. Since handwritten fonts' strokes are usually highly connected, simply using dilation may cause overlapping of strokes; see Fig. 4. Instead of using $\hat{b(y)}$ as the final synthesis, we use it as auxiliary information and output to the refinement module.

Refinement module. Refinement module merges information from the auxiliary branch to the dominating branch.

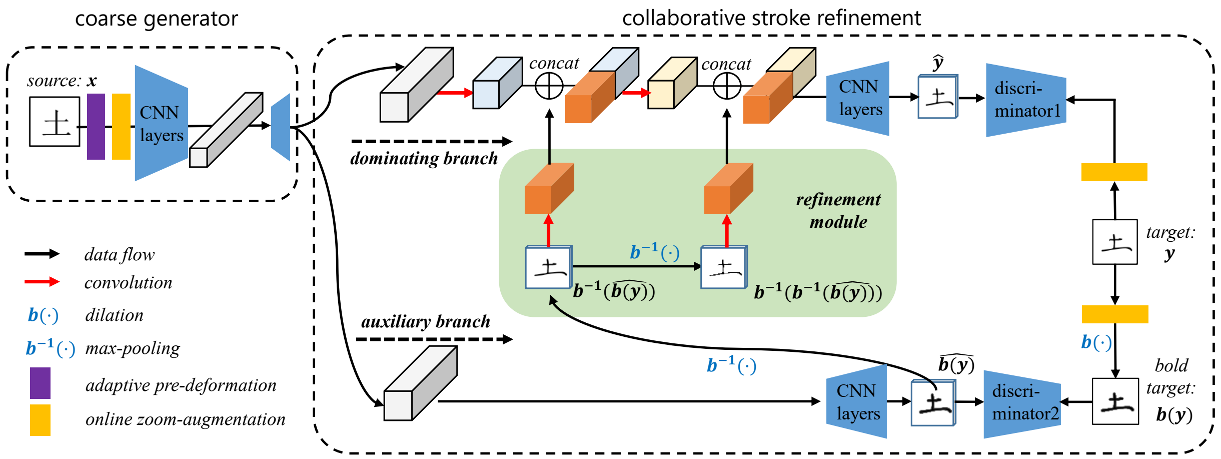


Figure 3. Schematic of proposed model including *coarse generator* and *collaborative stroke refinement*. In *coarse generator*, the source image x is transformed to a feature map after adaptive pre-deformation and online zoom-augmentation. In collaborative stroke refinement, *auxiliary branch* generates $\hat{b(y)}$, which uses $b(y)$ as ground-truth. Simultaneously, *dominating branch* generates $\hat{y}$ guided by the desired target y. In *refinement module*, $\hat{b(y)}$ is eroded by max-pooling into various stroke versions $b^{-1}(\hat{b(y)})$ and $b^{-1}(b^{-1}(\hat{b(y)}))$ and their feature maps flow to *dominating branch* as compensation. Besides, skip-connections are used between encoder and decoders in *coarse generator*, *dominating branch* and *auxiliary branch*.

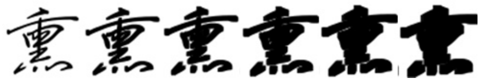


Figure 4. Dilation may cause the strokes overlapping issue.

To allow the dominating branch to capture various stroke weights, we aim to make $\hat{b(y)}$ thinner; however, directly using the erosion operation, i.e. the inverse of dilation, blocks the end-to-end training due to it is not differentiable. To overcome this, we use max-pooling to mimic erosion, which is differentiable. We use max-pooling twice to produce two stroke-weight syntheses, $b^{-1}(\hat{b(y)})$ and $b^{-1}(b^{-1}(\hat{b(y)}))$, where $b^{-1}(\cdot)$ denotes the max-pooling with kernel size 2, stride 2 and zero padding. We input these two syntheses, $b^{-1}(\hat{b(y)})$ and $b^{-1}(b^{-1}(\hat{b(y)}))$, to the convolution layers separately and obtain feature maps that reflect the same style with various stroke weights.

Dominating branch. The dominating branch collects auxiliary information from the refinement module and produces the final synthesis. The network structure is based on HAN with a generator and a hierarchical discriminator [7] (a discriminator to distinguish not only between the generated image and ground truth, but also between their feature maps, obtained from the first few layers before the output layer, and the feature maps of ground truth by conducting the same convolution), making the model converge faster and the results more smooth. The input of the generator is the output feature maps of the coarse generator. We process it with a convolution layer and then concatenate it with $b^{-1}(\hat{b(y)})$, which is the auxiliary information from the refinement module. We next process the concatenation with another convolution layer and concatenate it with $b^{-1}(b^{-1}(\hat{b(y)}))$. Twice concatenations allow the dominating branch to be aware of various stroke weights. We put the concatenation to the CNN to reconstruct the final synthesis $\hat{y}$. The discriminator tries to distinguish $\hat{y}$ and its feature maps from the ground truth y and y's feature maps. The dominating branch fuses auxiliary information from the refinement module, making the synthesis robust to distortion and missing strokes.

The above three parts are trained simultaneously. The auxiliary branch generates $\hat{b(y)}$ to approximate the skeleton of bold target; the refinement module extracts features from various stroke-weight syntheses and passes them to the dominating branch; the dominating branch finally merges the auxiliary information from the refinement module and produces the final syntheses. It pushes the networks to learn the deformation from one style to another with various stroke weights and handles the thin-stroke issue.

Meanwhile, in the back-propagation process, the gradient of dominating branch flows to auxiliary branch through refinement module because of the differentiability of *max-pooling*. The gradient from dominating branch passes the information of thin stroke skeleton to auxiliary branch, which alleviates the stroke overlapping issue illustrated in Fig. 4. This multi-branch design follows a similar favor of multi-task learning and collaborative training. By replacing the erosion operation by max-pooling, the information flowing through both of two branches in the forward and backward process makes them compensate each other; we thus call it *collaborative stroke refinement*.

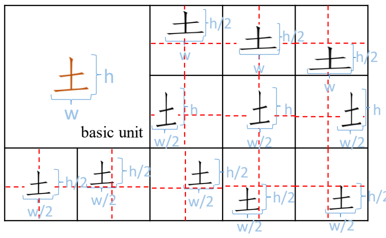


Figure 5. An example of "basic unit" and its corresponding most probable 11 typical zoom-augmented results.

3.2. Online Zoom-Augmentation

For Chinese, a radical may present in various characters at various locations, i.e. content-reuse (Fig. 2). A standardized character c can be modeled as

$$c = \sum_{\ell} d_{\ell}(b_{\ell}), \tag{1}$$

where b_ℓ is an elementary character, and $d_\ell(\cdot)$ is the deformation function associated with each elementary character. The shape of an arbitrary character c could have huge variations and it is thus hard to learn deformation directly based on c; on the other hand, the shapes of the elementary character b_ℓ are limited and it is thus much easier to learn deformation directly based on b_ℓ. The functionality of online zoom-augmentation is to explore this decomposition (1) adaptively. It has two main advantages: (i) it leverages the repetitiveness of radicals in Chinese characters and guides the networks to learn elementary structures of Chinese characters, leading to a significant reduction of the training size; and (ii) it guides the networks to learn characters at a variety of locations and sizes, making the networks robust. To our best knowledge, no CNN-based method leverages this prior knowledge except SA-VAE [23]. However, SA-VAE explicitly models this as a pre-labeling 133-bits embedding. Instead, we carefully select 750 training samples as b_ℓ; see details in Section 4.1.

We further use various positions and scales to train the element-wise deformation operator $d_\ell(\cdot)$. Specifically, when a paired image x, y is fed into the model, we zoom the centered character region to change the aspect ratio. We then translate the zoomed character region horizontally or vertically. Assuming the ratio of each character region is $h : w$, the zoomed result with $\frac{h}{2} : w$ will be vertically translated in the image (mostly translated to the upper/middle/bottom position), while the zoomed result with $h : \frac{w}{2}$ will be horizontally translated (mostly translated to the left/middle/right position). Additionally, the zoomed result with $\frac{h}{2} : \frac{w}{2}$ is translated to any possible position. The detail is illustrated in Fig. 5. Essentially, $d_\ell(\cdot)$ captures the mapping between b_ℓ and radicals of arbitrary character. Augmented training samples guide the networks to learn a variety of location patterns.

3.3. Adaptive Pre-deformation

The overall task is to learn a mapping from $x = d_{\text{source}}(c)$ to $y = d_{\text{target}}(c)$ for arbitrary character c. In Section 3.2, we decompose the character c. We can further decompose the deformations, $d_{\text{source}}(\cdot)$ and $d_{\text{target}}(\cdot)$, to ease the learning process. The main intuition is that some basic deformations, such as resize and rescale, are easy to learn. We can use a separate component to focus on learning those basic deformations and the main networks can then focus on learning complicated deformations. Mathematically, a general deformation function can be decomposed as

$$d(\cdot) = d_{\text{complex}}(d_{\text{rescale}}(d_{\text{resize}}(\cdot))).$$

The functionality of adaptive pre-deformation is to learn $d_{\text{rescale}}(\cdot)$ and $d_{\text{resize}}(\cdot)$, so that the main networks can focus on learning $d_{\text{complex}}(\cdot)$.

All paired x^i, y^i from the selected small dataset $\mathcal{S}_D$ construct the source image set $\mathcal{X}$ and the target image set $\mathcal{Y}$. We calculate the average character-region proportion r_1 and average character's height-width ration r_2 for $\mathcal{Y}$. First we find the minimum character bounding box b_i of each y^i, the height and width of b_i is respectively h_i and w_i. So,

$$r_1 = \frac{1}{N}\sum_{i}^{N}\frac{h_i \cdot w_i}{S \times S}, r_2 = \frac{1}{N}\sum_{i}^{N}\frac{h_i}{w_i};$$

where $N = 750$ and S is the image size. According to the above two statistics, we then pre-deformed each x^i to align its character region with y^i. The deformed result $\hat{x^i}$ is:

$$\hat{x^i} = d_{\text{rescale}}(d_{\text{resize}}(x^i)),$$

where $d_{\text{resize}}(\cdot)$ and $d_{\text{rescale}}(\cdot)$ denote the size-deformation and scale-deformation, respectively. The character skeleton of x^i is then roughly aligned with y^i. Here by pre-deformation, the skeleton of source character is roughly aligned with the corresponding target character, which reduces the transfer-difficulty. Specifically, the model does not fit the size and can focus on learning stroke distortions.

3.4. Losses

Our model has 4 loss terms for optimization, which are divided into 2 groups, $L^1_{\text{pixel}} + L^1_{\text{cGAN}}$ and $L^2_{\text{pixel}} + L^2_{\text{cGAN}}$.

$$L^1_{\text{pixel}}(G_1) = \mathbb{E}_{(x,y)}[-y \cdot (\log(\hat{y})) - (1-y) \cdot \log(1-\hat{y})],$$

$$\begin{aligned} L^1_{\text{cGAN}}(G_1, D_1) =& \mathbb{E}_{(x,y)}[\log D_1(x,y)] + \\ & \mathbb{E}_{(x,G_1(x))}[1 - \log D_1(x, G_1(x))], \end{aligned} \tag{2}$$

where y is the character with target font, (x, y) is pair-wise samples, $x \sim p_{\text{source\_domain}}(x)$ and $y \sim p_{\text{target\_domain}}(y)$.

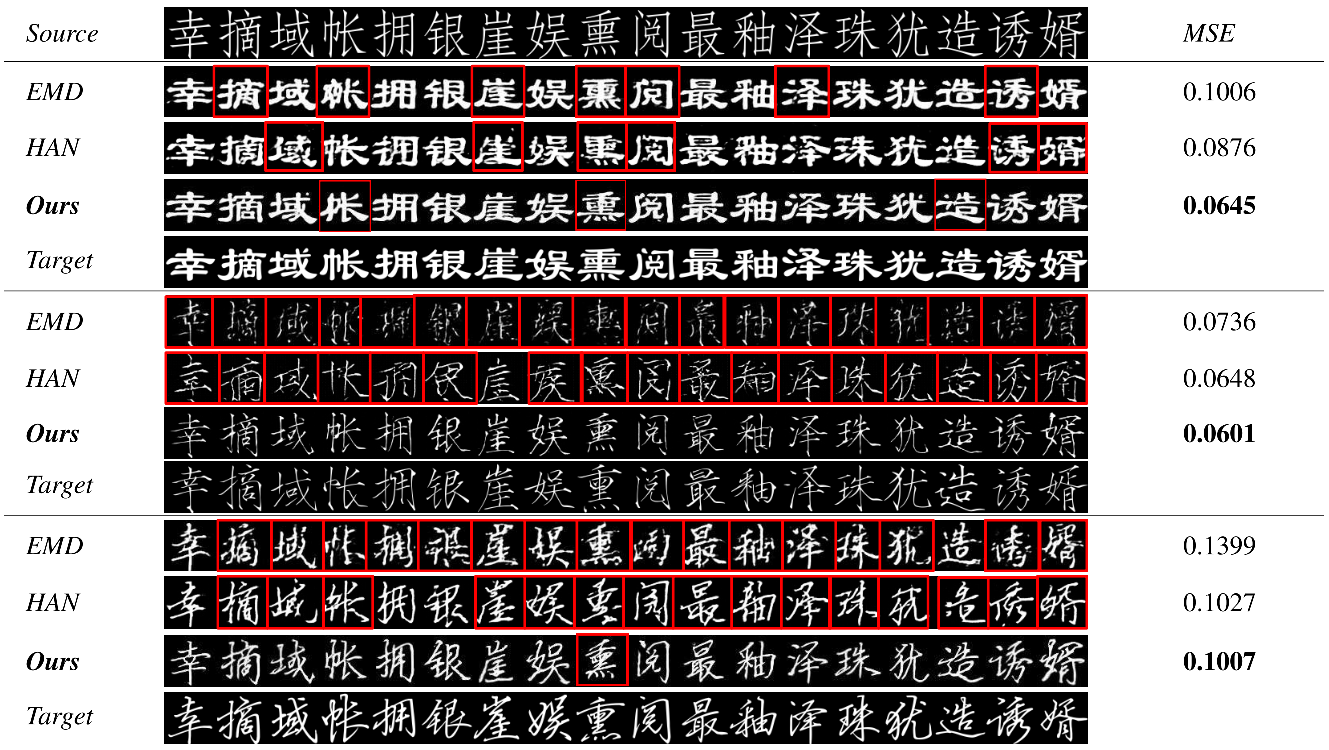


Figure 6. Performance of transferring *Fang Song* font (source) to other fonts (target) including **printed** (1st row) and **handwritten** fonts (2-3 rows). Characters in red boxes are failure samples. Both qualitative results and quantitative evaluation (MSE) show the best performance by our model.

D_1 is the *discriminator* 1. G_1 includes the entire network.

$$L^2_{\text{pixel}}(G_2) = \mathbb{E}_{(x,b(y))}[-b(y)\cdot(\log(b(\hat{y})))$$
$$-(1-b(y))\cdot\log(1-b(\hat{y}))],$$

$$L^2_{\text{cGAN}}(G_2,D_2) = \mathbb{E}_{(x,b(y))}[\log D_2(x,b(y))]+ \mathbb{E}_{(x,G_2(x))}[1-\log D_2(x,G_2(x)], \tag{3}$$

where $b(y)$ is the **bold** version character of target font. D_2 is the *discriminator* 2. G_2 only contains the *auxiliary branch*. The whole network is jointly optimized by $L^1_{\text{pixel}} + L^1_{\text{cGAN}} + L^2_{\text{pixel}} + L^2_{\text{cGAN}}$. Note that all x, y and $b(y)$ here are augmented by $d_\ell(\cdot)$ and pre-deformation function $d_{\text{resize}}(\cdot)$, $d_{\text{rescale}}(\cdot)$. We just write this for simplicity.

4. Experiments

We conduct extensive experiments to validate the proposed method and compare it with two state-of-the-art methods, HAN [7] and EMD [29]. These methods are chosen because they do not rely on extra human-annotated information and support a fair comparison. We also select HAN as our backbone, and more results on other backbones are displayed in Section 4.4.

4.1. Training Sample Selection

For Chinese characters, "compound" characters appear as obvious layout structure (e.g. left-right, top-bottom, surrounding, left-middle-right or top-middle-bottom), while "single-element" characters cannot be structural decomposed. Both radicals and "single-element" characters are known as basic units to construct all characters (see Fig. 8). Many compound Chinese characters share the same basic units in themselves, which means despite over 8000 characters in the Chinese language, there are only rather limited basic units (including 150 radicals and about 450 "single-element" characters). Based on this prior knowledge, a small set of characters are selected as training samples. We select 450 "single-element" characters and 150×2 compound characters covering all 150 radicals, to create a small dataset $\mathcal{S}_D$ totally including 750 training samples.

Our method can benefit from a pretraining process to further enhance its performance. This is implemented by concatenating a category embedding to the encoded character embedding in coarse generator, just like zi2zi[16]. We pretrain our model on 3000 commonly-used characters for 15 fonts, and then finetune the model by 750 samples on target font. This enables the model to observe more character skeletons and also speeds up the convergence.

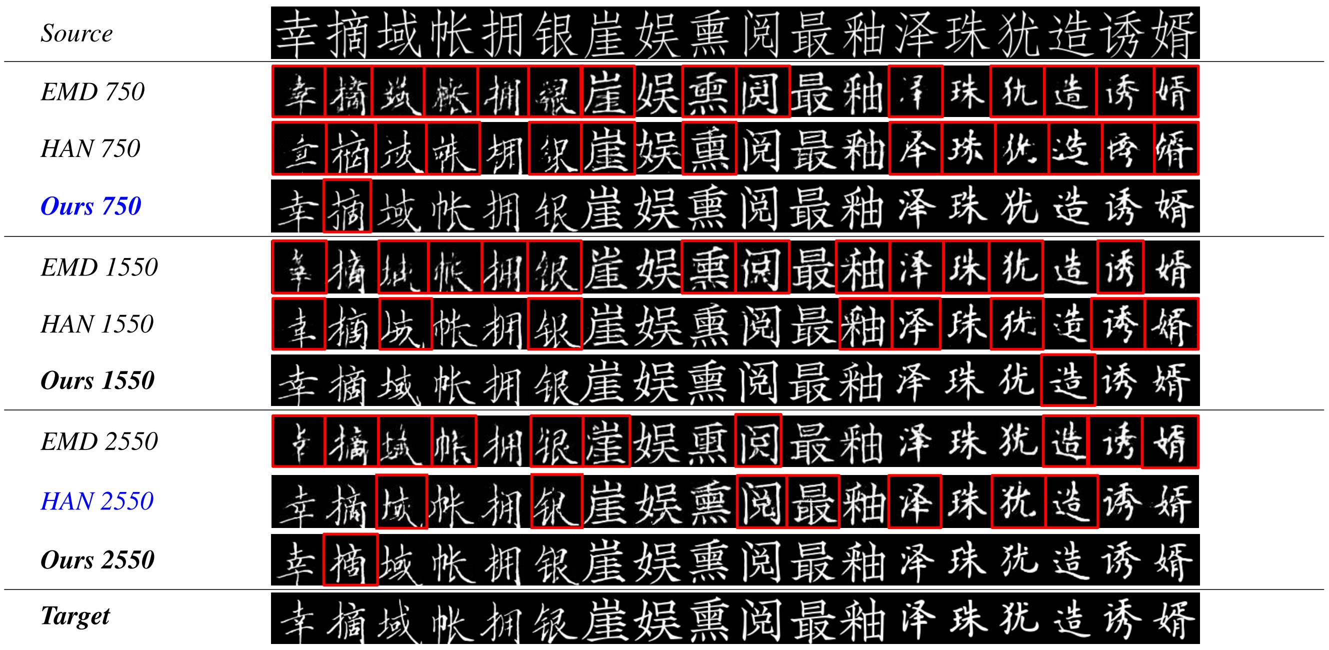
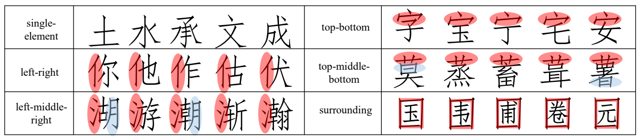


Figure 7. Performance comparison on three handwritten font with increasing size of training set. The comparison between *Ours 750* and *HAN 2550* demonstrates that our method achieves the equal even better performance with much less training set.



single-element
土 水 承 文 成
top-bottom
字 宝 宁 宅 安
left-right
你 他 作 估 伏
top-middle-bottom
莫 蒸 蓄 葺 薯
left-middle-right
湖 游 潮 渐 瀚
surrounding
国 围 圃 圈 园

Figure 8. Examples of "compound" and "single-element" characters. Some radicals are marked in red.

4.2. Comparison with Baseline Methods

As mentioned earlier, two state-of-the-art methods of Chinese font synthesis we compared with are HAN and EMD. HAN [7] is especially proposed for handwritten fonts. It proposes a hierarchical adversarial discriminator to improve the performance. Experiments show it relies on about 2500 paired training samples to achieve a good performance; EMD [29] achieves style transfer from a few samples by disentangling the style/content representation of fonts, while it relies on a large font library for training and performs poorly on handwritten fonts.

Performance on Small Dataset $\mathcal{S}_D$. Fig. 6 shows the comparison of our method with HAN and EMD under the selected small dataset $\mathcal{S}_D$. For EMD, for fair comparison, we pretrain the model with totally the same dataset as our model and then finetune it with $\mathcal{S}_D$. For HAN, we directly train it on $\mathcal{S}_D$ as it is not designed for pretraining. We choose both **printed-style** and **handwritten-style** fonts to fairly demonstrate the performance. Results show that our model slightly outperforms baseline methods on printed-style fonts (1st row) since printed-style fonts are always featured with regular structure and wide strokes, so the model takes no advantages of *collaborative stroke refinement*. However, our method achieves impressive results on handwritten fonts featuring thin strokes (2nd row) or irregular structure (3rd rows). Compared with baselines, our model generates more details without missing or overlapped strokes. For baselines, we can barely recognize some of their synthesized characters, where defections happen.

Performance on Different Training-set Size. We change the size of training set to 1550 and 2550 by randomly adding samples to $\mathcal{S}_D$. As shown in Fig. 7, our model gets smoother results if given more training samples. Improvement is more obvious on handwritten styles because the generated printed fonts have already been perfect on 750 training characters. For baselines, their results on 2550 training samples still cannot exceed ours on 750 characters.

Besides MSE, to rationally measure the fidelity of synthesized characters, we conduct a user study. 100 volunteers are invited to subjectively rate the synthesized characters from score 1 to 5, where 1 is the worst and 5 is the best. As shown in Fig. 9, the ratings of all methods are improved with more training samples. However, when the size is larger than 1550, the rising trend of our method stops and the score begins to float up and down. Thus we conclude that 1550 training samples have completely covered radicals/single-element characters with different shapes so that more training samples will not bring

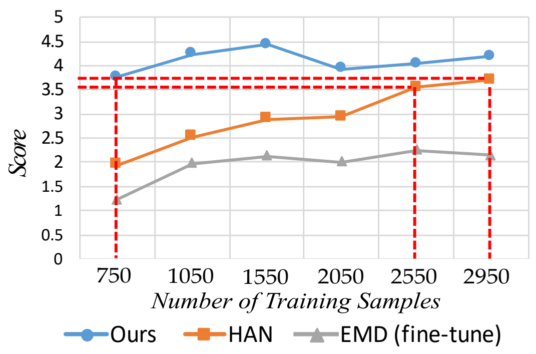


Figure 9. In the user study, our method reaches close subjective score with a much smaller training set compared with baselines.

improvement. Additionally, according to the user study, when the size of the training set is 750, our method achieves equal even higher subjective score compared with HAN trained by 2550 paired characters.

4.3. Ablation Study

Effect of Adaptive Pre-deformation We conduct experiments on removing adaptive pre-deformation or not. As shown in the second row of Fig. 10, some strokes are missing if the model is trained without pre-deformation. When absolute locations of a certain stroke are seriously discrepant between the source character and the target, the network may be confused about whether this stroke should be abandoned or be mapped to another position. The adaptive pre-deformation roughly aligns the strokes, relieving the model of learning the mapping of stroke location.

Effect of Online Zoom-Augmentation The results after removing the online zoom-augmentation technique are shown in the third row of Fig. 10. The generated strokes are so disordered that we even cannot recognize the characters. Without zoom-augmentation, the model is actually trained with only 750 paired samples, which leads to serious over-fitting. Zoom-augmentation induces the model implicitly to learn the shape and location diversity. Besides, it also models common structure information by these vertically or horizontally translated characters. So our method can reconstruct correct topological structures of characters.

Effect of Stroke Refinement We disconnect the data flow from *refinement module* to *dominating branch* to analyze the effect of stroke refinement module. Comparison results in Fig. 10 show that characters generated with stroke refinement strategy have more continuous structures, while those without stroke refinement present seriously deviant and even broken strokes, especially on cursive handwritten fonts (*Font* 1 and *Font* 2). These phenomena prove the effectiveness of stroke refinement.

Effect of Pre-training We omit the pretraining process and train the model on target font from scratch. Though only 750 paired characters are used, the results are still competitive with few missing strokes. This proves that online zoom-augmentation helps handle the problem of limited data.

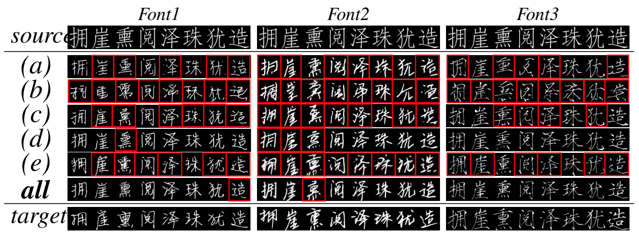


Figure 10. Ablation analysis without adaptive pre-deformation (a), online zoom-augmentation (b), stroke refinement (c), pre-training (d) and GAN-related losses (e).

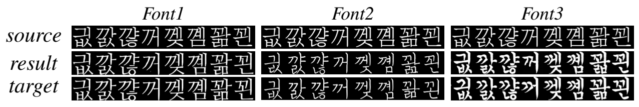


Figure 11. Results of Korean fonts prove that our method can be transplanted to other languages with content reuse phenomenon.

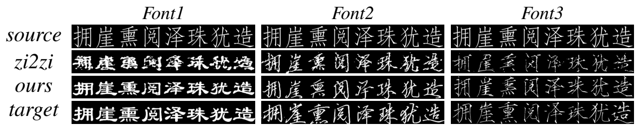


Figure 12. Results of Chinese fonts by regular zi2zi (2nd row) and our model with zi2zi backbone (3rd row).

Effect of GAN We remove the influence of two discriminators by setting GAN-related loss terms, Eq. 2 and Eq. 3, to zero. The results in Fig. 10 generally are a little blurry.

4.4. Extension Study

Portability on Language We conduct experiments on Korean fonts, which also have thin stroke issues. Fig. 11 shows that the syntheses are impressively similar to the targets, validating our model can be applied to generation tasks on fonts of other languages with handwriting style and content reuse phenomenon.

Portability on Backbone Our method is a common framework for any elementary font generation model. We simply change the backbone from HAN to zi2zi and also see remarkable improvement (see Fig. 12).

5. Conclusions

We propose an end-to-end model to synthesize handwritten Chinese fonts with only 750 training samples. The model includes three main novelties: collaborative stroke refinement, handling the *thin stroke issue*; online zoom-augmentation, exploiting the content-reuse phenomenon; and adaptive pre-deformation, standardizing and aligning the characters. We perceptually and quantitatively evaluate our model and the experimental results validate the its effectiveness.

References

[1] Cosmin Ancuti, Codruta O Ancuti, Radu Timofte, Luc Van Gool, Lei Zhang, Ming-Hsuan Yang, Vishal M Patel, He Zhang, Vishwanath A Sindagi, Ruhao Zhao, et al. NTIRE 2018 challenge on image dehazing: methods and results. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2018.

[2] Samaneh Azadi, Matthew Fisher, Vladimir G. Kim, Zhaowen Wang, Eli Shechtman, and Trevor Darrell. Multi-content GAN for few-shot font style transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

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

[4] Junbum Cha, Sanghyuk Chun, Gayoung Lee, Bado Lee, Seonghyeon Kim, and Hwalsuk Lee. Few-shot compositional font generation with dual memory. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[5] Bo Chang, Qiong Zhang, Shenyi Pan, and Lili Meng. Generating handwritten Chinese characters using Cyclegan. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2018.

[6] Huiwen Chang, Jingwan Lu, Fisher Yu, and Adam Finkelstein. PairedCycleGan: Asymmetric style transfer for applying and removing makeup. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[7] Jie Chang, Yujun Gu, Ya Zhang, and YanFeng Wang. Chinese handwriting imitation with hierarchical generative adversarial network. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2018.

[8] Dongdong Chen, Lu Yuan, Jing Liao, Nenghai Yu, and Gang Hua. Stylebank: An explicit representation for neural image style transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[9] Yang Chen, Yu-Kun Lai, and Yong-Jin Liu. CartoonGAN: Generative adversarial networks for photo cartoonization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[10] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

[12] Yue Jiang, Zhouhui Lian, Yingmin Tang, and Jianguo Xiao. Dcfont: an end-to-end deep Chinese font generation system. In *SIGGRAPH Asia Technical Briefs*, 2017.

[13] Yue Jiang, Zhouhui Lian, Yingmin Tang, and Jianguo Xiao. SCFont: Structure-guided Chinese font generation via deep stacked networks. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

[14] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[15] Kaonashi-tyc. Rewrite: Neural style transfer for Chinese fonts. `https://github.com/kaonashi-tyc/Rewrite`, 2016.

[16] Kaonashi-tyc. zi2zi: Master Chinese calligraphy with conditional adversarial networks. `https://github.com/kaonashi-tyc/zi2zi`, 2017.

[17] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[18] Pengyuan Lyu, Xiang Bai, Cong Yao, Zhen Zhu, Tengteng Huang, and Wenyu Liu. Auto-encoder guided GAN for Chinese calligraphy synthesis. In *Proceedings of the 14th IAPR/IEEE International Conference on Document Analysis and Recognition (ICDAR)*, 2017.

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

[20] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *The International Conference on Learning Representations (ICLR)*, 2016.

[21] Artsiom Sanakoyeu, Dmytro Kotovenko, Sabine Lang, and Bjorn Ommer. A style-aware content loss for real-time hd style transfer. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[22] Akshay Srivatsan, Jonathan Barron, Dan Klein, and Taylor Berg-Kirkpatrick. A deep factorization of style and structure in fonts. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019.

[23] Danyang Sun, Tongzheng Ren, Chongxuan Li, Hang Su, and Jun Zhu. Learning to write stylized Chinese characters by reading a handful of examples. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2018.

[24] Paul Upchurch, Noah Snavely, and Kavita Bala. From a to z: supervised transfer of style and content using deep neural network generators. *arXiv preprint arXiv:1603.02003*, 2016.

[25] Jue Wang and Anoop Cherian. Learning discriminative video representations using adversarial poerturbations. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[26] Xuefeng Xiao, Lianwen Jin, Yafeng Yang, Weixin Yang, Jun Sun, and Tianhai Chang. Building fast and compact convolutional neural networks for offline handwritten Chinese character recognition. *Pattern Recognition*, 72:72–81, 2017.

[27] Shuai Yang, Jiaying Liu, Wenjing Wang, and Zongming Guo. TET-GAN: Text effects transfer via stylization and destylization. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

[28] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[29] Yexun Zhang, Ya Zhang, and Wenbin Cai. Separating style and content for generalized style transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[30] Zhao Zhong, Xu-Yao Zhang, Fei Yin, and Cheng-Lin Liu. Handwritten Chinese character recognition with spatial transformer and deep residual networks. In *Proceedings of the IAPR International Conference on Pattern Recognition (ICPR)*, 2016.

[31] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.