FontRL: Chinese Font Synthesis via Deep Reinforcement Learning

Yitian Liu, Zhouhui Lian[*]
Wangxuan Institute of Computer Technology, Peking University, Beijing, P.R. China
{lsflyt, lianzhouhui}@pku.edu.cn

Abstract

Automatic generation of Chinese fonts is a valuable but challenging task in areas of AI and Computer Graphics, mainly due to the huge amount of Chinese characters and their complex glyph structures. In this paper, we propose FontRL, a novel method for Chinese font synthesis by using deep reinforcement learning. Specifically, we first train a deep reinforcement learning model to obtain the Thin-Plate Spline (TPS) transformation that is able to modify the reference stroke skeleton in a mean font style into the skeleton of a required style for each stroke of every unseen Chinese character. Afterwards, we utilize a CNN model to predict the location and scale information of these strokes, and then assemble them to get the skeleton of the corresponding character. Finally, we convert each synthesized character skeleton into the glyph image via an image-to-image translation model. Both quantitative and qualitative experimental results demonstrate the superiority of the proposed FontRL compared to the state of the art. Our code is available at https://github.com/lsflyt-pku/FontRL.

Introduction

Computer fonts are widely used in our daily lives. Human beings often express various meanings by writing different words which can also contain different emotional feelings by adopting different font styles. As now more and more people are seeking their personalized fonts, existing commercial font libraries cannot satisfy the rapidly-increasing demands. The demand gap problem is hard to resolve for writing systems that consist of large numbers of characters (e.g., Chinese). For example, there are 6,763 Chinese characters in a GB2312 (an official standard) font library, while the GB18030-2005 character set contains more than 70,000 Chinese characters whose glyph structures are typically complicated. Since the number of characters contained in a Chinese font library is so large that it is quite time-consuming and costly to design new Chinese fonts. Moreover, the quality of font design relies heavily on the designer's capability and experience. To alleviate the workload and cost of designing a new font library, this paper aims to automatically synthesize glyph images for all Chinese characters in a font library by training on a small number of input character images.

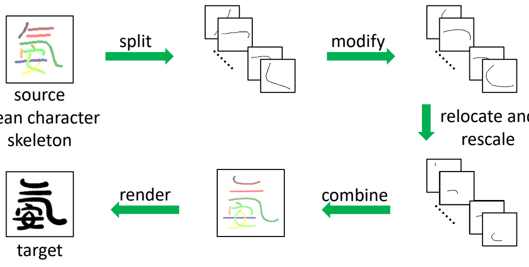

Figure 1: An overview of the proposed FontRL.

Automatic generation of Chinese fonts is a challenging task that has attracted many researchers in the last few years. Those existing methods can be roughly divided into two groups which process glyph images and sequential data, respectively.

The methods in the first group treat a given Chinese character as an image. To be specific, they typically use convolutional neural networks (CNNs) to extract the content and style features of characters, and then reconstruct features to convert glyph images in the source font to the target font. For example, based on pix2pix (Isola et al. 2017), Tian [1] proposed zi2zi by designing an encode-decode network and adding a non-trainable gaussian noise as category embedding for one-to-many modeling. DCFont (Jiang et al. 2017) replaces the above-mentioned random gaussian noise embedding with style features extracted from the font feature reconstruction network, and then establishes the mapping from the reference font to the target font by using residual blocks. Some researchers added extra prior knowledge of Chinese characters to generative models to guide the generation, such as SA-VAE (Sun et al. 2017) and SCFont (Jiang et al. 2019). In this type of methods, most of them directly use CNNs to extract and transfer font styles to generate glyph images in the target style. However, they ignore the high-level information of Chinese characters, and thus often obtain synthesis results with incorrect topological structures and blurry contours.

[*]Corresponding author.

[1]https://kaonashi-tyc.github.io/2017/04/06/zi2zi.html

The writing process naturally contains timing and position information, based on which the second type of methods treat the Chinese character as sequences of points, e.g., FontRNN (Tang et al. 2019). Through a specifically-designed RNN model, FontRNN learns the transformation from the input sequence to the target one, and thus effectively simulates the writing behaviors of human beings. However, this type of methods fail to synthesize high-quality glyph contours.

To address the above-mentioned problems, we propose FontRL, a novel Chinese font synthesis system based on deep reinforcement learning (DRL). As shown in Figure 1, we treat the writing trajectory of a given Chinese character as the combination of stroke skeletons, which are composed of point sequences. However, unlike FontRNN (Tang et al. 2019), our FontRL does not directly generate the sequential data of writing trajectories. Instead, we search for an optimal RL-action to obtain the Thin-Plate Spline (TPS) (Bookstein 1989) transformation that is able to modify the reference stroke skeleton in a mean font style into the skeleton of a required style for each stroke of every unseen Chinese character. After assembling the generated stroke skeletons to a complete character skeleton, we use a CNN-based image-to-image translation model to convert the character skeleton into a glyph image. Compared to other existing CNN-based font synthesis methods, the proposed FontRL is simple and easy to train, and it does not require to conduct pre-training on large amounts of data. In other words, our FontRL can be directly trained on a small number of input Chinese characters written/designed in a specific style to generate the corresponding high-quality font library. Compared to existing RNN-based methods, our FontRL learns how to modify the character skeleton in a reference style instead of generating the skeleton from scratch, ensuring the generation of a smooth and natural writing trajectory. In this manner, high-quality glyph images for all characters in a Chinese font library can be synthesized. In summary, major contributions of this paper are as follows.

- We propose FontRL, a novel font synthesis system, based on deep reinforcement learning (DRL). To the best of our knowledge, we are the first to use reinforcement learning to handle the challenging task of font synthesis.
- We design a two-stage architecture which implements the modification of reference stroke skeletons and prediction of stroke bounding boxes sequentially to ensure that the generated character skeleton is satisfactory. Compared with other methods based on character skeletons, such as EasyFont (Lian et al. 2019) and SCFont (Jiang et al. 2019), higher-quality writing trajectories can be synthesized by FontRL without pre-training on large-scale data.
- Extensive experiments have been conducted to demonstrate the superiority of the proposed FontRL compared to the state of the art both quantitatively and qualitatively.

Related Work

Neural Style Transfer

Until now, a large amount of methods have been proposed on neural style transfer (Gatys, Ecker, and Bethge 2016) that aims to migrate the style from a reference image to a content image via neural networks. For instance, Isola et al. (Isola et al. 2017) proposed a first image-to-image translation model, pix2pix, by utilizing the combination of U-Net (Ronneberger, Fischer, and Brox 2015), condition GAN and PatchGAN (Isola et al. 2017). Zhu et al. (Zhu et al. 2017) designed CycleGAN to handle the image-to-image translation problem for unpaired data. One possible solution to handle the Font synthesis task is to transfer the style of a given reference font into the target font style based on neural style transfer. However, existing common-purpose methods for neural style transfer (e.g., pix2pix and CycleGAN) can not be directly used to generate satisfactory glyph images.

Chinese Font Generation

In the last few years, many researchers attempted to generate high-quality Chinese fonts by using deep generative models. Rewrite [2] employs a traditional top-down CNN structure to synthesize Chinese glyph images. Since the network is quite simple, synthesis results of Rewrite are far from satisfactory. Zi2zi, which mainly follows the network architecture of pix2pix, establishes a conditional GAN-based model for font style transfer by adding category embedding to the generator and discriminator. EMD (Zhang, Zhang, and Cai 2018) can generate novel styles and contents which are unseen in the training set by training on just a few input glyph images. However, the quality of its synthesis results is relatively poor for handwriting fonts. To handle this problem, Wen et al. (Wen et al. 2019) proposed CSR, an end-to-end deep neural network, to synthesize glyph images for unseen Chinese characters with only 750 samples as input, and Chang et al. (Chang et al. 2018) developed HCCG-CycleGAN, a deep generative model based on CycleGAN and DenseNet (Huang et al. 2017), for handwriting Chinese font generation.

However, above-mentioned methods could inevitably generated low-quality glyph images for some Chinese characters due to their complex structures. Thereby, some researchers tried to integrate some high-level prior knowledge of Chinese characters with generative models. For example, SA-VAE (Sun et al. 2017) encodes a character as a 133-bit vector, containing information of glyph structure, radical, and index codes, which has proved to work better than one-hot embedding. By introducing decomposed character skeletons as reference data and utilizing a series of stacked deep networks to synthesize writing trajectories and their corresponding glyph images, SCFont (Jiang et al. 2019) possesses a better capability of handling the problems of structural correctness and style consistency than other existing methods. As we know, a writing trajectory can also be represented as key point sequence. Following this idea, Tang et al. (Tang et al. 2019) proposed FontRNN, a RNN-based generative model, to directly synthesize writing trajectories in the target font style, showing impressive performance for cursive handwriting characters.

[2]https://github.com/kaonashi-tyc/Rewrite

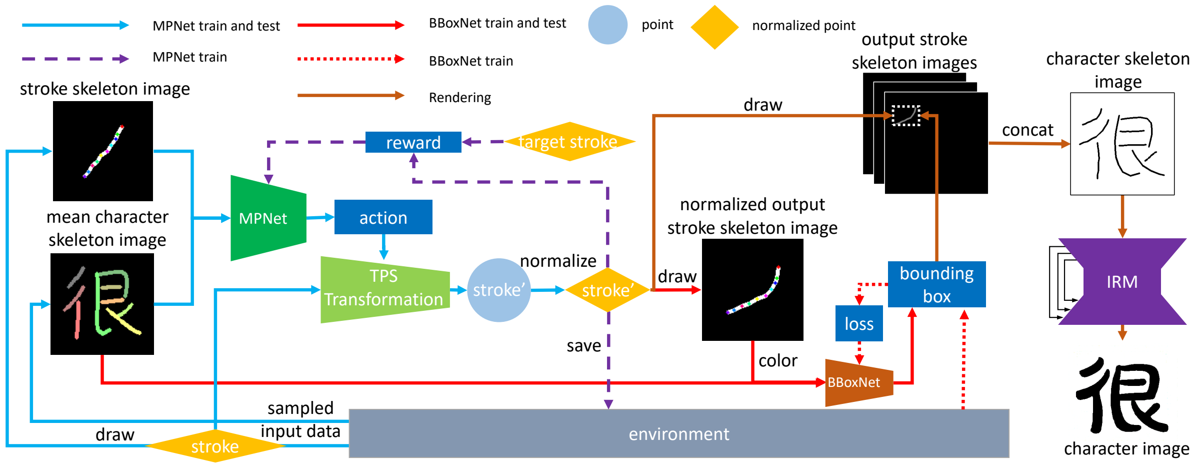

Figure 2: The architecture of our FontRL, which consists of four key components: MPNet, TPS transformation module, BBoxNet and Image Rendering Module (IRM). The dotted parts where the ground truth is available only appear during training.

Reinforcement Learning-based Generation

With the help of deep reinforcement learning (DRL), many challenging problems have been resolved, such as the game of GO (Silver et al. 2017), text generation (Guo 2015) and robot control (Levine et al. 2016). The decisions of these tasks naturally form a sequence, and the writing process is similar. To write a character, we need to choose where to start and end, and how to move the pen. However, until now no work has been reported to apply DRL to synthesize glyph images. Here, we briefly review some related work.

Some researchers proposed to repaint images based on DRL, such as LPaintB (Jia et al. 2019a), PaintBot (Jia et al. 2019b), LearningtoPaint (Huang, Heng, and Zhou 2019) and SPIRAL++ (Mellor et al. 2019). The main differences between them are the utilization of different actions, rewards, and drawing windows in the corresponding reinforcement learning algorithms. In particular, PaintBot achieves style transfer by changing the training set and test set. Since character images are lack of color and texture information compared with painting images and natural images, the result of transferring font style using this method is unsatisfactory. Wu et al. (Wu et al. 2018) proposed a system which is able to draw the strokes of Chinese characters by controlling the robotic arm through DRL. However, affected by the low accuracy of the robot, the structures and contours of generated strokes are fuzzy. In addition, this system can only write characters in the training set, but is not able to handle unseen characters.

Method Description

As shown in Figure 1, the proposed FontRL first tries to learn the TPS transformation that is able to modify the writing trajectory in a mean font style into a target style for each stroke of a given Chinese character. Then, the strokes' bounding boxes are predicted and used to assemble these synthesized stroke skeletons to obtain the skeleton of the corresponding character. Finally, the synthesized character skeleton is converted into a glyph image via an image-to-image translation model. To achieve the goal mentioned above, we design a deep reinforcement learning (DRL) based architecture as depicted in Figure 2, which consists of the following four key modules: Modification Parameter estimating Network (MPNet), TPS Transformation Module (TPSTM), Bounding Box predicting Network (BBoxNet) and Image Rendering Module (IRM).

More specifically, we first use MPNet, which is based on DRL, to estimate parameters of the TPS transformation for each stroke. Then, we employ the TPS interpolation function to modify the mean stroke skeleton into the target one. Since the anchors of TPS transformation are fixed on a given canvas but strokes may be located anywhere on the canvas, in order to better modify strokes, we normalize the strokes before implementing TPS transformation. After normalization, the stroke skeleton is resized and located in the center of the canvas without position and length information. In the third step, we utilize BBoxNet to predict the centers and side lengths of the bounding squares for all synthesized stroke skeletons, and then assemble them into a complete character skeleton. At last, for the IRM module, we directly apply the skeleton-to-image translation model (StyleNet) proposed in SCFont (Jiang et al. 2019) to render the character skeleton to the corresponding glyph image. It should be pointed out that the stroke skeletons of a given glyph image are extracted in the same manner as SCFont (Jiang et al. 2019). More details of our proposed MPNet, TPSTM, and BBoxNet are presented in the following subsections.

Modification Parameter Estimating Network (MPNet)

The transformation of a stroke skeleton from the reference style to a target style can be described as multiple thin-plate spline (TPS) interpolation functions. These functions possess the same form but different parameters. We should not randomly select some functions (the bad distribution of training data leads to bad results) or use all functions (im-

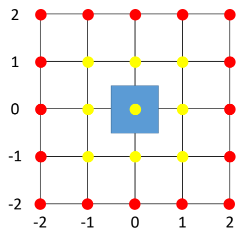

Figure 3: 25 anchor points are utilized to control the size and shape details of a stroke skeleton.

possible to get all interpolation functions) as training data. Therefore, in our system, we use deep reinforcement learning to train MPNet to get the parameters required for TPS transformation to solve the problem of lacking suitable training data. Note that the training set used in this paper just consists of 775 Chinese characters including 7004 strokes. When implementing reinforcement learning, we update the memory bank repeatedly, which expands the training data to get better performance. The details of MPNet are described as follows.

Anchor If the coordinates of key points of a stroke skeleton have the same x or y values, the stroke's minimum bounding rectangle will degenerate into a line segment. Thereby, we calculate the minimum bounding square for each stroke. Before applying TPS transformation, strokes are normalized and placed into the blue square region as shown in Figure 3, whose side length and center coordinate are 1 and $(0, 0)$, respectively. To effectively control the modification of the stroke skeleton, we use 25 anchors (see Figure 3), among which the red and yellow anchors control the overall scale and shape details of the stroke, respectively.

Action The action a_t is a 25×2 vector, which is defined as the offset of each anchor point. Since it is necessary to ensure that the relative position of each anchor point pair remains unchanged before and after modification, we experimentally restrict the range of values in the vector to $[-0.49, 0.49]$.

State The state is given by the environment and defined as

$$state = \left(s^{skel}; c^{skel}; stepnum; coord\right), \tag{1}$$

where s^{skel} denotes the normalized stroke skeleton image, c^{skel} denotes the reference character skeleton image, $stepnum$ is normalized to $[0, 1]$, and $coord$ is a coordinate system with x and y values ranging from -0.5 to 0.5. All these images are obtained by painting and connecting the key points of stroke skeletons on the 128×128 canvas.

Reward In MPNet, the reward of each action is defined by the average distance of points between the generated skeleton and the target skeleton

$$r\left(s_t, a_t\right) = dist\left(s_t^{skel}\right) - dist\left(TPS\left(s_t^{skel}, a_t\right)\right), \tag{2}$$

$$dist\left(s_t^{skel}\right) = \|s_t^{skel} - s_{gt}^{skel}\|_2, \tag{3}$$

where $r(s_t, a_t)$ is the reward at the step t on the state s_t with the action a_t, $dist(s_t^{skel})$ denotes the distance between the generated skeleton and the target skeleton at the step t, and $TPS(s_t, a_t)$ will be described in the next section. In FontRL, we normalize the synthesized stroke skeleton and the ground-truth target before calculating the distance and reward (see the purple dotted line in Figure 2).

We have tried to use Q-learning by combining the value Q and the reward R to update the policy net, but the performance was poor. We find that the value Q cannot be precisely estimated in our task. So, we update the policy net by just using the reward.

Policy net The policy net decides an action based on the state and updates the policy based on the received reward. For reinforcement learning, the policy net does not need to understand the physical meaning of an action, but only needs to establish the relationship between rewards and actions. In FontRL, we adopt $Resnet18$ with a single FC layer as the policy net.

Environment As shown in Figure 2, the environment first sends the state to the policy net and modifies the stroke skeleton through TPS transformation based on the action. Then, the environment calculates the reward and returns it to the policy net to update parameters. What's more, the environment provides the target ground-truth bounding box to calculate the reward for MPNet and train BBoxNet. By adding the environment, we decouple MPNet from other modules in FontRL. This allows us to use different kinds of transformation modules without re-training other parts.

TPS Transformation Module (TPSTM)

Thin-plate spline (TPS) (Bookstein 1989) transformation is defined as follows: we smoothly bend a thin plate, which can be considered as a two-dimensional plane, so that the N source points $\mathbf{Ps} = (Ps_1, Ps_2, \ldots, Ps_N)^T$ are shifted to the corresponding target points $\mathbf{Pt} = (Pt_1, Pt_2, \ldots, Pt_N)^T$ after bending. To minimize the bending energy, we compute the interpolation functions by

$$\Phi_x(Ps_i) = c_x + A_x^T Ps_i + W_x^T S(Ps_i), \tag{4}$$

$$\Phi_y(Ps_i) = c_y + A_y^T Ps_i + W_y^T S(Ps_i), \tag{5}$$

where c_x and c_y are scalars, A_x and A_y are 2×1 vectors, and W_x and W_y are $N \times 1$ vectors. The x and y values of the coordinate of the target point Pt_i are determined by $\Phi_x(Ps_i)$ and $\Phi_y(Ps_i)$, respectively. The function $S(Ps_i)$ is defined as

$$\begin{gathered} S(Ps_i) = \left(\sigma\left(Ps_i - Ps_1\right), \cdots, \sigma\left(Ps_i - Ps_N\right)\right)^T, \\ \sigma(\Delta Ps_{ij}) = \|\Delta Ps_{ij}\|_2^2 \log \|\Delta Ps_{ij}\|_2, \end{gathered} \tag{6}$$

where $\Delta Ps_{ij} = Ps_i - Ps_j$. Then, we have

$$\begin{bmatrix} W_x & W_y \\ c_x & c_y \\ A_x & A_y \end{bmatrix} = \mathbf{\Gamma}^{-1} \begin{bmatrix} \mathbf{Pt}_x & \mathbf{Pt}_y \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, \tag{7}$$

Method	FZJHSXJW		FZSSBJW		FZTLJW		FZYNJW		FZZJ-LPYBJW	
	L1 loss	IoU	L1 loss	IoU	L1 loss	IoU	L1 loss	IoU	L1 loss	IoU
FontSL	0.1943	0.4087	0.2395	0.1928	0.1560	0.1839	0.2272	0.3609	0.1492	0.2379
FontRNN	0.1818	0.4556	0.2393	0.2768	0.1603	0.3460	0.1884	0.4845	0.1732	0.2699
pix2pix	0.1851	0.4007	0.2290	0.1846	0.1687	0.2048	0.1782	0.4135	0.1491	0.2331
DCFont	0.1630	0.4125	0.1906	0.1672	0.1459	0.1223	0.1640	0.3908	0.1327	0.1808
zi2zi	0.1483	0.4682	0.1936	0.2598	0.1558	0.2431	0.1527	0.4328	0.1400	0.2688
SCFont	0.1173	0.5459	0.1627	0.3163	0.1188	0.3574	0.1245	0.5442	0.1191	0.3197
FontRL	**0.0855**	**0.6963**	**0.1261**	**0.5399**	**0.1074**	**0.4836**	**0.0966**	**0.6890**	**0.1035**	**0.4843**

Table 1: Quantitative results of our FontRL and other existing methods.

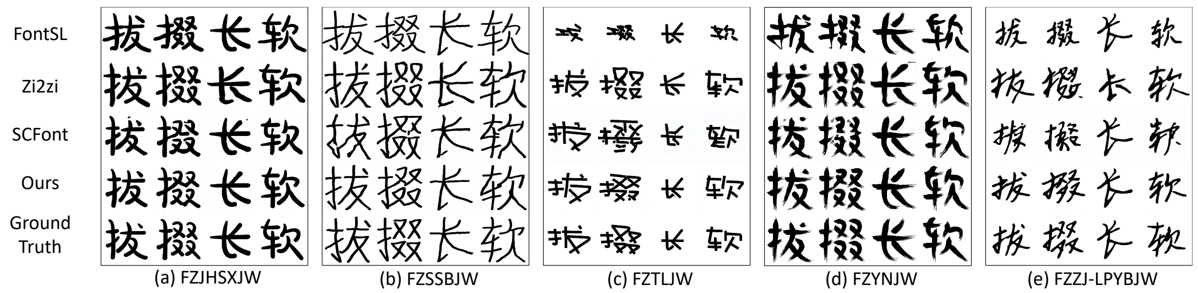

Figure 4: Comparison of text rendering results for our FontRL and other three models.

$$\mathbf{\Gamma} = \begin{bmatrix} \mathbf{S} & \mathbf{1} & \mathbf{Ps}_x & \mathbf{Ps}_y \\ \mathbf{1} & \mathbf{0} & 0 & 0 \\ \mathbf{Ps}_x^T & 0 & 0 & 0 \\ \mathbf{Ps}_y^T & 0 & 0 & 0 \end{bmatrix}, \tag{8}$$

$$\mathbf{S} = \begin{bmatrix} \sigma(\Delta Ps_{11}) & \sigma(\Delta Ps_{12}) & \cdots & \sigma(\Delta Ps_{1N}) \\ \sigma(\Delta Ps_{21}) & \sigma(\Delta Ps_{22}) & \cdots & \sigma(\Delta Ps_{2N}) \\ \vdots & \vdots & \ddots & \vdots \\ \sigma(\Delta Ps_{N1}) & \sigma(\Delta Ps_{N2}) & \cdots & \sigma(\Delta Ps_{NN}) \end{bmatrix}, \tag{9}$$

where $\mathbf{Pt}_x$ and $\mathbf{Pt}_y$ consist of the x and y values of the coordinates of N target points $\mathbf{Pt}$, respectively, and so are $\mathbf{Ps}_x$ and $\mathbf{Ps}_y$ for source points.

In FontRL, the source points $\mathbf{Ps}$ are the anchors, and the target points $\mathbf{Pt}$ can be computed by $\mathbf{Pt} = action + \mathbf{Ps}$. By solving Eq. 7, we get the TPS transformation parameters: $C = [c_x, c_y]^T$, $W = [W_x, W_y]^T$, and $A = [A_x, A_y]^T$. Then, each key point Pks on the source stroke skeleton can be transformed to the corresponding key point Pkt on the target stroke skeleton by computing

$$Pkt = \Phi(Pks) = C + A^T Pks + W^T S(Pks). \tag{10}$$

Bounding Box Predicting Network (BBoxNet)

After stroke modification via TPS transformation, we need to assemble those synthesized stroke skeletons into a complete character skeleton. Here, we propose BBoxNet to predict the center and size of the bounding square for a stroke.

The bounding box prediction module and the stroke modification module are independent of each other. So BBoxNet can be replaced by any other object detection networks. To verify its effectiveness, we adopt a simple network ($Resnet34$ with a single FC layer) to predict the stroke center and the side length. Using a small network reduces the possibility of over-fitting. As shown in Figure 2, the input of BBoxNet is the generated stroke skeleton and the reference character skeleton in a mean font style. We use different colors to label different strokes so that the model can learn the relationship of each stroke and its corresponding stroke position in the mean character skeleton to improve the accuracy of prediction.

Loss function We have conducted experiments to respectively apply the GIoU loss and the MAE Loss in the loss function of BBoxNet, and obtained similar performance. For simplicity, we choose the MAE Loss in our loss function to reduce the computation complexity.

Experiments

In this section, we first introduce the dataset used here and the implementation details of FontRL. Then, we compare FontRL with other models in both quantitative and qualitative performance. Afterwards, we further verify the effectiveness of FontRL by showing the results of text rendering and a user study. Finally, we conduct ablation studies to show the effects of TPS transformation and BBoxNet.

Dataset

In our experiments, we directly use the dataset introduced in (Jiang et al. 2019), which consists of glyph images of all 6763 Chinese characters and their manually-specified stroke skeletons in 5 different font styles as target and a mean font style as reference. In order to fairly compare performance, as SCFont (Jiang et al. 2019), we use an input character set proposed in (Lian, Zhao, and Xiao 2016) for training. The

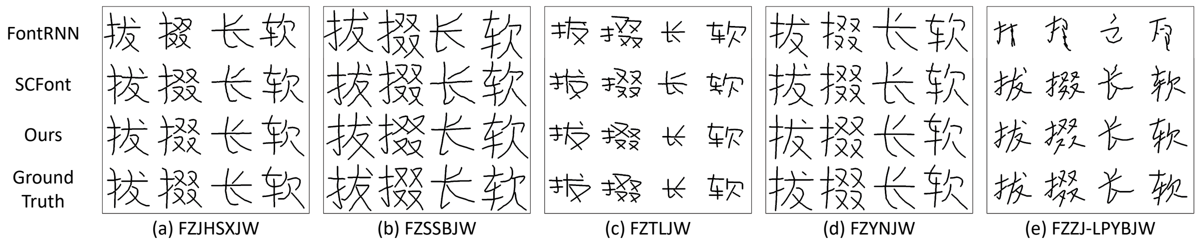

Figure 5: Comparison of synthesized character skeletons obtained by our method, FontRNN and SCFont, respectively.

input character set consists of 775 Chinese characters that are able to cover all kinds of strokes and components in the GB2312 character set.

Implementation Details

The input image resolution of MPNet is 128 × 128, and the coordinates of key points on the stroke skeleton are normalized to $[-0.5, 0.5]$. Similarly, the input image resolution of BBoxNet is selected as 128×128. In our experiments, we directly use StyleNet proposed for SCFont (Jiang et al. 2019) as the image rendering module of our FontRL and the resolution of the input image for StyleNet is 320 × 320.

The learning rate of MPNet is initialized as 0.0003, and decayed to 0.0001 after 6000 iterations; the learning rate of BBoxNet is initialized as 0.001, decayed to 0.0005 after 40 epochs, and then decayed to 0.0001 after 100 epochs; the hyper-parameters of StyleNet are set to the default values.

Quantitative and Qualitative Results

We compare FontRL with six existing methods: FontSL (Lian et al. 2019), FontRNN (Tang et al. 2019), SCFont (Jiang et al. 2019), zi2zi, DCFont (Jiang et al. 2017) and pix2pix (Isola et al. 2017). Among them, zi2zi, DCFont, and pix2pix are deep learning-based image-to-image translation approaches, which directly apply style transfer on glyph images, while FontSL, FontRNN and SCFont both generate character skeletons in the target style and render them into glyph images. Default settings of those methods as the original papers are adopted in our experiments.

As we can see from Table 1, FontRL performs much better than other models in quantitative measurements, which are estimated by computing the L1 loss and the IoU value between synthesized images and the corresponding ground truth. The lowest L1 loss and the highest IoU value have been obtained by our method, clearly demonstrating the superiority of FontRL compared to other approaches.

Figure 4 shows some qualitative results of FontRL and other methods. We can see that many local details in synthesis results obtained by zi2zi are missing. For FontSL, because of adopting many prior knowledge of Chinese characters, it guarantees the content correctness of its synthesis results, while the overall visual effects and stroke details of many characters are unsatisfactory. Generally speaking, SCFont performs well in most situations. However, it performs poorly when the target style (e.g., the FZTLJW font) is quite different against the reference one. On the contrary, as shown in Figure 4, glyph images synthesized by FontRL are very similar to the ground truth. Compared with zi2zi and SCFont, FontRL generates high-quality glyph images with correct structures and more smooth and natural strokes, especially for glyphs with complex shapes and cursive handwriting styles.

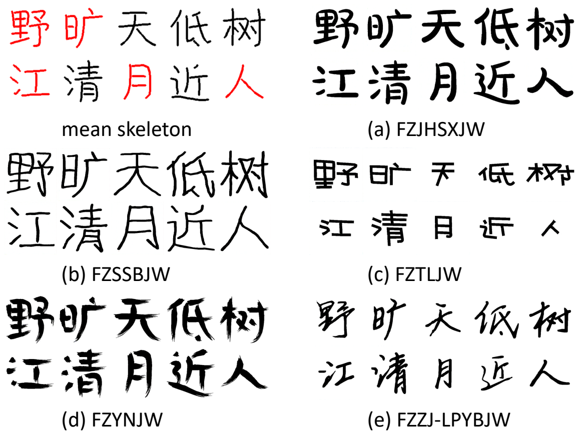

Figure 6: A Chinese poem rendered using 5 different fonts synthesized by our FontRL. The reference mean skeletons of these characters are shown on the top, where machine-generated characters are marked in red and others are written/designed by human beings.

Both FontRNN, SCFont and FontRL are deep learning based models that generate character skeletons which are then rendered into glyph images. Thereby, we compare the writing trajectories generated by these three methods. As shown in Figure 5, since the proposed FontRL uses TPS transformation to modify the key points of reference character skeletons instead of directly synthesizing the skeleton images, smoother writing trajectories can be generated compared to FontRNN and SCFont. Moreover, the style of our synthesized character skeletons is also more similar as the ground truth.

Text Rendering

Our FontRL can not only generate high-quality glyph images but also ensure the style consistency between machine-generated characters and input samples. As shown in Figure 6, we render a Chinese poem by using 5 Chinese fonts generated by FontRL, in which machine-synthesized characters are marked in red while human-created characters are

Font	Accuracy
FZJHSXJW	0.5029
FZSSBJW	0.5099
FZTLJW	0.5077
FZYNJW	0.5040
FZZJ-LPYBJW	0.5001
Average	0.5049

Table 2: User study results

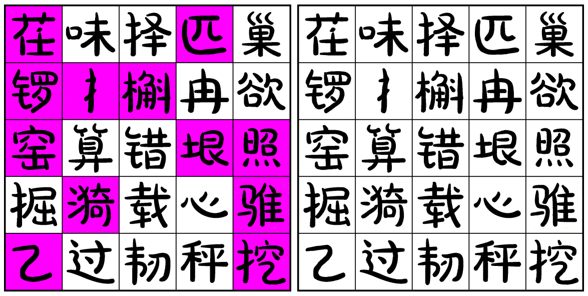

Figure 7: An example of a region of the test paper and answer paper used in our user study. Glyph images in colored blocks are synthesized by FontRL and others are written/designed by human beings.

rendered in black. As we can see, images synthesized by our model are consistent in style with the corresponding input samples, making it hard to distinguish machine-generated characters from those written/designed by human beings.

User Study

We also verify the effectiveness of our method by conducting a user study, in which 153 educated Chinese people participated. The ages of those participants range from 13 to 56 years old. In this test, participants were asked to find out all glyph images that are generated by FontRL from the random mixture of 100 human-created and 100 machine-generated character images in one of the above-mentioned five font styles (see Figure 7 for an example). In our web-based user interface, 50 ground-truth glyph images in a target font style are also displayed as guidance. As we can see from Table 2, the average selecting accuracy for each font style is very close to 50%, which means that it is quite difficult for participants to distinguish glyph images synthesized by our FontRL and those written/designed by human beings.

Ablation Studies

TPS or Affine Transformation In Figure 8, we demonstrate why the proposed FontRL uses TPS transformation instead of affine transformation. As mentioned before, MPNet outputs multiple parameters to control various linear transformations such as rotation, translation, shear-warp, etc. Compared with TPS transformation, the stretching and deformation ranges of affine transformation are much smaller (examples are marked in blue), and complex skeleton modifications cannot be accomplished via linear transformation (examples are marked in red). Also, modifications based on affine transformation may occasionally result in unrecognizable characters (see the third column in Figure 8).

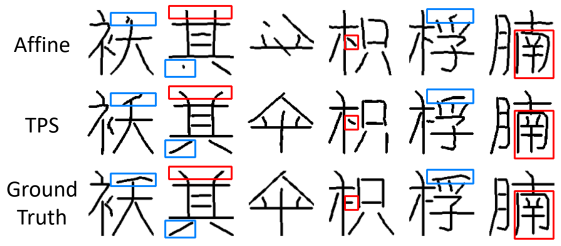

Figure 8: Comparison of methods adopting TPS transformation and affine transformation, respectively.

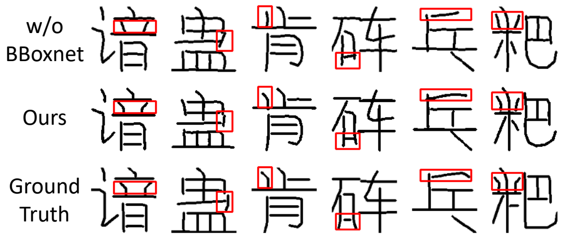

Figure 9: The effect of BBoxNet.

maxstep	1	2	3	4	5
distance	0.310	0.303	0.281	0.266	0.266

Table 3: The average distance of points between the generated skeleton and the target skeleton in the FZJHSXJW font with different maximum numbers of steps.

Effect of BBoxNet BBoxNet is proposed to predict the position and size of each stroke's bounding square. Thus, the stroke skeletons can be normalized before implementing TPS transformation. In this manner, the performance of skeleton modification is markedly improved. We compare the synthesis results with and without BBoxNet in Figure 9, where red boxes are used to mark the regions in which the two synthesis results differ significantly. We observe that with BBoxNet the proposed FontRL can synthesize glyph images with better stroke details. Moreover, due to the small size of the network, BBoxNet is computationally efficient.

Maximum Number of Steps

In order to prove the effectiveness of reinforcement learning (RL) in FontRL, we calculate the average distance of points between the generated skeleton and the target skeleton under different maximum numbers of steps ($maxstep$) as shown in Table 3. When $maxstep$ is set to 1, the RL algorithm used in FontRL degenerates to supervised learning. As $maxstep$ increases, the average distance of stroke skeletons gradually decreases and converges. Comparing the results obtained by setting $maxstep$ to 1 and 5, it can be observed that the RL algorithm in FontRL improves the quality of syn-

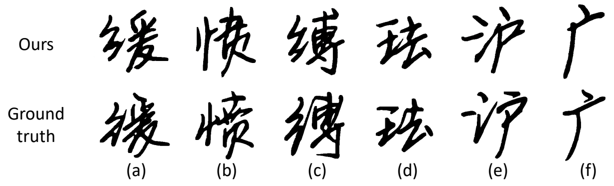

Figure 10: Some failure cases of the proposed FontRL.

thesized stroke skeletons.

Limitations

For the generation of cursive handwriting fonts, there is still room for improvement in the details of glyph images synthesized by our method. When the structure of a glyph is too complex, FontRL cannot generate high-quality ligatures between some stroke pairs as human-written glyphs (Figure 10(a)(b)(c)). Moreover, when the target skeleton and the reference mean skeleton are significantly different, it is difficult to achieve all deformation effects through TPS transformation (Figure 10(d)(e)). Finally, the predicting accuracies of stroke bounding boxes for some characters are still unsatisfactory, leading to poor synthesis results (Figure 10(f)).

Conclusion

In the paper, we proposed a novel Chinese font synthesis system named FontRL. The key idea is to adopt deep reinforcement learning to learn how to draw the writing trajectory of a given Chinese character and then convert the character skeleton into a glyph image by using an image-to-image translation model. Specifically, we utilized the modification parameter estimation network (MPNet) and the TPS transformation module to achieve the mapping of a stroke skeleton from the reference style to the target style. And we adopted the bounding box predicting network (BBoxNet) to estimate the center and size of each stroke and assembled synthesized strokes into character skeletons. Finally, we employed an existing model to render character skeletons into images. Extensive experiments demonstrated that FontRL obtains the best performance in both quantitative results and visual effects compared to other font synthesis methods.

Acknowledgements

This work was supported by Beijing Nova Program of Science and Technology (Grant No.: Z191100001119077), National Natural Science Foundation of China (Grant No.: 61672043 and 61672056), Center For Chinese Font Design and Research, and Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology).

References

Bookstein, F. L. 1989. Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Transactions on pattern analysis and machine intelligence* 11(6): 567–585.

Chang, B.; Zhang, Q.; Pan, S.; and Meng, L. 2018. Generating handwritten chinese characters using cyclegan. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 199–207. IEEE.

Gatys, L. A.; Ecker, A. S.; and Bethge, M. 2016. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2414–2423.

Guo, H. 2015. Generating text with deep reinforcement learning. *arXiv preprint arXiv:1510.09202* .

Huang, G.; Liu, Z.; Van Der Maaten, L.; and Weinberger, K. Q. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4700–4708.

Huang, Z.; Heng, W.; and Zhou, S. 2019. Learning to Paint with Model-based Deep Reinforcement Learning. *CoRR* abs/1903.04411. URL http://arxiv.org/abs/1903.04411.

Isola, P.; Zhu, J.-Y.; Zhou, T.; and Efros, A. A. 2017. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1125–1134.

Jia, B.; Brandt, J.; Mech, R.; Kim, B.; and Manocha, D. 2019a. LPaintB: Learning to Paint from Self-Supervision. *arXiv preprint arXiv:1906.06841* .

Jia, B.; Fang, C.; Brandt, J.; Kim, B.; and Manocha, D. 2019b. Paintbot: A reinforcement learning approach for natural media painting. *arXiv preprint arXiv:1904.02201* .

Jiang, Y.; Lian, Z.; Tang, Y.; and Xiao, J. 2017. DCFont: an end-to-end deep Chinese font generation system. In *SIGGRAPH Asia 2017 Technical Briefs*, 22. ACM.

Jiang, Y.; Lian, Z.; Tang, Y.; and Xiao, J. 2019. SCFont: Structure-Guided Chinese Font Generation via Deep Stacked Networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 4015–4022.

Levine, S.; Finn, C.; Darrell, T.; and Abbeel, P. 2016. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research* 17(1): 1334–1373.

Lian, Z.; Zhao, B.; Chen, X.; and Xiao, J. 2019. EasyFont: A Style Learning-Based System to Easily Build Your Large-Scale Handwriting Fonts. *ACM Transactions on Graphics (TOG)* 38(1): 6.

Lian, Z.; Zhao, B.; and Xiao, J. 2016. Automatic generation of large-scale handwriting fonts via style learning. In *SIGGRAPH ASIA 2016 Technical Briefs*, 12. ACM.

Mellor, J. F.; Park, E.; Ganin, Y.; Babuschkin, I.; Kulkarni, T.; Rosenbaum, D.; Ballard, A.; Weber, T.; Vinyals, O.; and Eslami, S. 2019. Unsupervised doodling and painting with improved spiral. *arXiv preprint arXiv:1910.01007* .

Ronneberger, O.; Fischer, P.; and Brox, T. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, 234–241. Springer.

Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. 2017. Mastering the game of go without human knowledge. *Nature* 550(7676): 354–359.

Sun, D.; Ren, T.; Li, C.; Su, H.; and Zhu, J. 2017. Learning to write stylized chinese characters by reading a handful of examples. *arXiv preprint arXiv:1712.06424* .

Tang, S.; Xia, Z.; Lian, Z.; Tang, Y.; and Xiao, J. 2019. FontRNN: Generating Large-scale Chinese Fonts via Recurrent Neural Network. In *Computer Graphics Forum*, volume 38, 567–577. Wiley Online Library.

Wen, C.; Chang, J.; Zhang, Y.; Chen, S.; Wang, Y.; Han, M.; and Tian, Q. 2019. Handwritten Chinese Font Generation with Collaborative Stroke Refinement. *arXiv preprint arXiv:1904.13268* .

Wu, R.; Fang, W.; Chao, F.; Gao, X.; Zhou, C.; Yang, L.; Lin, C.-M.; and Shang, C. 2018. Towards deep reinforcement learning based Chinese calligraphy robot. In *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 507–512. IEEE.

Zhang, Y.; Zhang, Y.; and Cai, W. 2018. Separating style and content for generalized style transferzhang2018separating. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 8447–8455.

Zhu, J.-Y.; Park, T.; Isola, P.; and Efros, A. A. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, 2223–2232.