H-024

Chinese and Japanese Scene Character Recognition Using Stroke-Level Synthetic Data

Minglu Zhang†, Hideaki Goto‡, Takuo Suganuma‡

1 Introduction

Character recognition has been studied for decades and has many applications in our daily life, like OCR (Optical Character Recognition). However, even with the development of deep learning methods, the accuracy of Chinese and Japanese scene character recognition is still not satisfactory. Despite the noise in the scene images, the training data for Chinese and Japanese is also quite limited, considering that Chinese and Japanese have much larger number of characters and are more complex in structure compared with other languages.

To handle this problem, efforts on both the data side and the model side have been made. On the data side, Ren et al. propose a synthetic data engine of Chinese in [1], which is a pipeline of image process that can generate images with diversity in shape and randomness in colors. For Japanese character recognition, Horie and Goto provide a data synthesis pipeline including different kinds of transformation operators, especially the random filter, the weight of which is randomly chosen in order to introduce random distortion in the images [2].

On the model side, the most representative work is the research on zero-shot Chinese character recognition. Cao et al. decompose Chinese characters into radicals and represent each Chinese character with a tree structure, which reveals the underlying similarity of Chinese characters [3]. Chen et al. further decompose Chinese characters into strokes, which are the smallest unit of Chinese character components [4]. To make full use of the spatial information beneath the complex structures of Chinese characters, Zu et al. leverage information from all levels of character structures and make joint matching to obtain the final predictions [5].

In this paper, we develop a stroke extraction method that can obtain stroke labels for semantic segmentation from fonts. Moreover, we suggest an attention module that can integrate the information of stroke semantic segmentation to improve the recognition accuracy.

2 Scene Character Recognition with Stroke-Level Synthetic Data

2.1 Overview

The overview of our method is illustrated in Figure 1. First, we extract the stroke images from the fonts of characters, then the stroke images are regarded as stroke labels and are processed by a data synthesis engine together with font images, yielding stroke-level synthetic data. After that, a stroke-level semantic segmentation model is trained and then integrated into our attention module, which is also a part of the classification model.

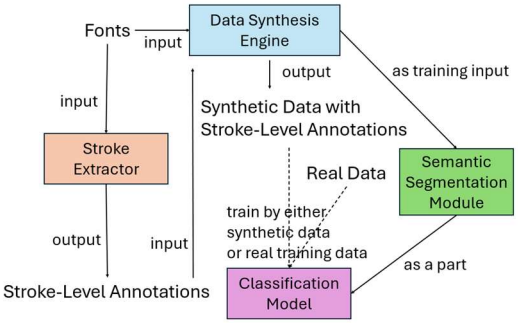

Figure 1 The overview of our method.

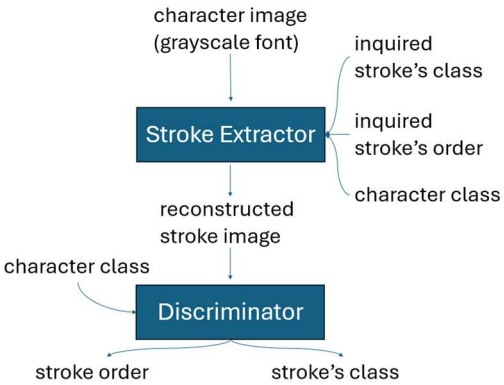

Figure 2: The framework of our stroke extraction method.

2.2 Stroke Extraction

2.2.1 Overview

Our method is inspired by [6]. In this work, models with an encoder-decoder structure are trained to infer the stroke images of the input Chinese character image in the correct writing order, so that the rich spatial information of Chinese characters can be leveraged.

As most fonts are designed in character-level, data synthesis engines driven by fonts data cannot generate stroke-level annotations without stroke-level information of the fonts. We develop an encoder-decoder based model to reconstruct the image of the inquired stroke from the input character image. The framework of our stroke extraction method is shown in Figure 2. For each inference made by the stroke extractor, the input content does not only include the character image, but also the class of the character, the class of the inquired stroke and the order of the stroke. The extra input information other than the image is expected to aid the model to perform better on the particular task of stroke extraction. A discriminator is also included to deal with the domain gaps between the font with stroke annotations and the other fonts that do not have stroke annotations.

† Graduate School of Information Sciences, Tohoku University
‡ Cyberscience Center, Tohoku University

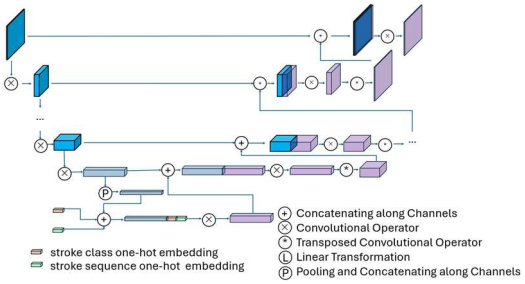

Figure 3: Structure of the Stroke Extractor.

For Japanese stroke extraction, we refer to KanjiVG [7], which provides information about stroke orders of Japanese as well as the corresponding stroke annotations for each character. By extracting the images of characters and strokes from Scalable Vector Graphics (SVG) files, a Japanese font with stroke-level annotations is obtained and is used as the supervised part of our dataset. From the annotations of KanjiVG, we summarize all the strokes into 76 classes in our study.

For Chinese stroke extraction, we use annotations from cjklib [8] and ACPM [5], and the stroke images are from makemehanzi [9]. Finally, all the strokes of Chinese characters are summarized into 31 classes. The aim of semi-supervised stroke extraction is to infer the stroke areas of a character image taken from a normal font, which only provides grayscale images of characters. After inferring the corresponding stroke images, which are also presented as grayscale images, training data for semantic segmentation of strokes can be synthesized.

The stroke extraction model is a U-Net[10] with an encoder-decoder structure, which is shown in Figure 3. One-hot embeddings are used to inform the model which stroke to extract, and the embeddings are concatenated with the deepest feature maps, which are at the bottom of the network.

2.2.2 Training Stages and Loss Functions

One pretrain stage and two finetuning stages are set for stroke extraction. In the pretrain stage, we pretrain only the stroke-level labeled font is included and is called the source font. In the finetuning stages, the fonts without stroke annotations, which are called the target fonts, will also be included.

In the pretrain stage, only the source-segmentation loss is included, and is defined as:

$$loss_{src\_seg} = \frac{1}{NHW}\sum_{n=1}^{N}\sum_{p=1}^{HW} y_{n,p}\log x_{n,p} + (1 - y_{n,p})\log(1 - y_{n,p}), \quad (1)$$

where N, H, W represent the batch size, the height of input images and the width of input images, respectively. $x_{n,p}$ denotes the value of the pth pixel in the nth batch, and $y_{n,p}$ represents the grayscale value (0 means the background area and 1 means the character area) of the pth pixel in the nth batch as the label. This loss measures how different the inferred stroke images are from their labels. Rather than Cross Entropy Loss (CE), BCE is used so that it is convenient to calculate the gradient for the purpose of the following finetuning stages.

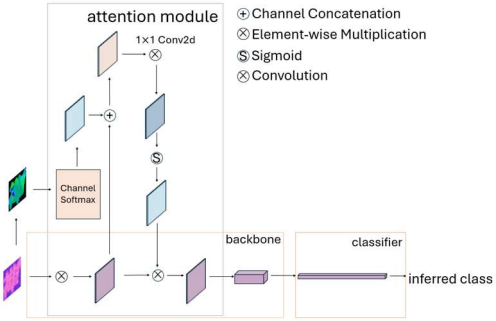

Figure 4 The overview of the attention module.

In finetuning stages, to make sure that the reconstructed stroke images can be roughly correct, we define a class loss and an order loss for the discriminator, for both source and target fonts. The source class loss and target class loss as (only the source class loss is shown):

$$loss_{src\_cls} = -\frac{1}{N}\sum_{n=1}^{N}\sum_{v=1}^{V} \log\frac{e^{x_{n,y_{n,v}}}}{\sum_{c=1}^{C} e^{x_{n,v}}}, \quad (2)$$

where v denotes the class index of the inquired stroke and V denotes the number of character classes.

The source stroke order loss and the target stroke order loss are also defined as below (only the source stroke order loss is shown):

$$loss_{src\_order} = -\frac{1}{N}\sum_{n=1}^{N}\sum_{d=1}^{D} \log\frac{e^{x_{n,y_{n,d}}}}{\sum_{c=1}^{C} e^{x_{n,d}}}, \quad (3)$$

where d denotes the order of the inquired stroke, and D denotes the number of stroke classes. The stroke order loss measures if the discriminator can correctly infer the inferred stroke image's stroke order.

In the second finetuning stage, to avoid generating too much overlapping when inferring strokes from the same input image, we use merging loss defined as:

$$loss_{merging} = -\frac{1}{NHW}\sum_{n=1}^{N}\sum_{p=1}^{HW} \left(\sum_{s=1}^{S} x_{n,p,s} - y_{n,p}\right)^2, \quad (4)$$

where s represents the stroke order of the stroke and S denotes the stroke number of the character, while the others are of the same definitions as (1). The merging loss measures how well the merged image of all the inferred stroke images of one character image is as similar as the original character image.

For each stage, the loss functions that are used in the previous stage are also included, and all loss functions are combined into a polynomial.

2.3 Attention Module based on semantic segmentation of strokes

We develop an attention module that can integrate semantic segmentation maps of strokes to help improve the performance of CNN models. The overview of our attention module is shown in Figure 4.

Firstly, the semantic segmentation map is obtained by the pretrained semantic segmentation model, then an argmax function is applied to the map along the channel dimension. Then, for each pixel, only the elements whose channel corresponds to the result of the argmax function are set to the value of a hyperparameter *alpha*, other elements are set to 0. After that, a softmax function is applied also along the channel dimension, then the obtained feature map is

concatenated with the feature map by the backbone along the channel dimension. Then a 1×1 convolutional layer reduces the number of channels (including normalization), followed by a sigmoid activation function.

3 Experiments and Discussion

3.1 Stroke Extractor

The stroke extraction experiments are done on both Chinese and Japanese languages, while the following steps are done only on Japanese. We use the Adam optimizer for both Chinese and Japanese stroke extractors in every stage of training. The pretrain stage takes 7,400 epochs for the Chinese stroke extractor and 4,000 epochs for the Japanese stroke extractor, with the learning rate of 0.0001. Both the finetune stage 1 and finetune stage 2 takes 20 epochs with the learning rate of 0.00001.

3.2 Data Synthesis Engine

We use a preliminary data synthesis pipeline in our laboratory to generate training data for semantic segmentation models for both Chinese and Japanese. The pipeline is composed by the permutation of basic transformation operators defined as: Perspective Transformation (PT), Morphological Transformation (MO), Color Change (CC), Gaussian Noise (GN), Random Filter (RF) and Gaussian Filter (GF). Except PT, MO and CC operators, all the other operators are the same as the ones defined in [2]. The start points of PT operator are randomly selected from four vertices of the input image within the range $[z + L * \alpha_1, z + L * \alpha_2]$, where z denotes the x or y coordinate of one of the four vertices in the input image, and L denotes the size of input image. The end points correspond to the four vertices of the output image, the size of which is the same as the input's. The MO operator performs morphological transformation with a preset possibility, and one operation among Dilation4, Dilation8, Pushup and Pushdown will be done. Dilation4 and Dilation8 follow the usual definitions and Pushup and Pushdown move the character one pixel upward or downward, respectively. The CC operator randomly generates 8-bit RGB colors for both the background part and the character part, and the generated colors satisfy the conditions shown as below.

$$\left\| \frac{r_b+g_b+b_b}{3} - \frac{r_c+g_c+b_c}{3} \right\| \geq gap, \qquad (5)$$

where b and c represent the background part and the character part, respectively, and r, g, b denote the values in each channel of RGB, ranging from 0 to 255.

By the order of PT-MO-CC-GF-(RF-GF)×3-GN with configuration shown as Table 1, both the character images for training and the annotations for semantic segmentation are generated. The annotations for semantic segmentation are grayscale images generated along with the generation of character images for training from the font images, and the only difference is that the annotation images do not go through the operation of CC and GF. The detailed configuration is shown in Table 1.

Table 1: Data Synthesis Configuration

operation	parameter
PT	α_1= -0.1, α_2= 0.1
MO	*probability*=0.7
CC	*gap* = 50
GN	μ = 0 σ = 0.1
RF	δ = 1
GF	σ ∈ (0, 10)

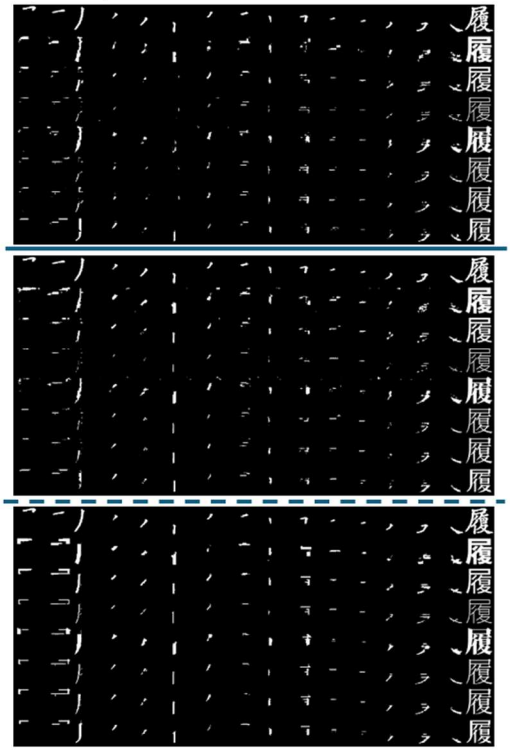

Figure 5 The results of stroke extraction. From top to bottom are results after the pretrain stage, the first finetuning stage and the second finetune stage.

3.3 Semantic Segmentation of Strokes

For both Chinese and Japanese character recognition, we train the semantic segmentation model with the learning rate of 0.00001 for 140 epochs in total. The optimizer is set to Adam, and we use xavier uniform method to initialize the weights of the models.

We test our semantic segmentation model on the dataset JPSC1400 [11], which is composed of 1,400 Japanese scene characters.

3.4 Discussion on experiment results

As there are a lot of similarities between Chinese and Japanese characters, in this paper we only analyze the results of Chinese stroke extraction and semantic segmentation of Japanese strokes.

Figure 6: An example of the training data for semantic segmentation.

Table 2 The experiment on effectiveness of stroke semantic segmentation maps.

number	model	highest accuracy (%)
1	Control Group 1	89.07
2	Control Group 2	90.43
3	Experimental Group	**91.21**

To better analyze the performance of our Chinese stroke extractor, we choose the character “履”, which consists of a large number of strokes.

The results of stroke extraction after each training stage are shown in Figure 5. The first row corresponds to the source font while the other 7 rows correspond to target fonts. The columns correspond to the strokes of the character, following the stroke order from left to right. The result shows that the pretrain stage is enough for the source font, while many of the reconstructed strokes of target fonts are broken, and some even failed with all pixels in the black color. After the first finetune stage, the problem that the stroke extractor generates images with no stroke region is solved, and the shapes of strokes becomes clearer. However, there are still many overlapping areas between strokes of the same character from the same font. After the second finetune stage, the problem of overlapping areas has been solved. Although there are still entangled strokes according to the reconstruction results, it usually happens when the number of strokes is large, and for most characters with a relatively small number of strokes, the influence is tolerable.

The synthetic data is shown in Figure 6. Each column corresponds to one stroke except the last column, which denotes the complete character. The first row corresponds to the input while the second row corresponds to the labels. We train our stroke-level semantic segmentation model by the synthetic data.

Finally, our experiment is carried out on JPSC1400. Two control groups are set to prove the effectiveness of the attention module. Control Group 1 corresponds to the plain network without the module, while the classification model in Control Group 2 has the attention module, but the semantic segmentation submodule has not been trained. The number of epochs is set to 40 and the optimizer is Adam with learning rate 0.0001. The value of *alpha* is set to 0.75.

The result shows that the structure of the attention module itself works, and the accuracy can be further improved with stroke-level semantic segmentation module trained. About 2.14% accuracy improvement is obtained in total compared with the plain network.

4 Conclusion

In this paper, we have developed a semi-supervised stroke extractor for both Chinese and Japanese characters. This method can help extract the strokes from an arbitrary character image and has been successfully applied in both Chinese and Japanese characters. The stroke-level semantic segmentation with the synthetic data also shows the potential to improve character recognition accuracy as additional information other than the input image.

As future work, we seek to find more ways to leverage the structural information of Chinese and Japanese characters as additional information to improve the model’s performance.

References

[1] Xiaohang Ren, Kai Chen, and Jun Sun. “A CNN Based Scene Chinese Text Recognition Algorithm With Synthetic Data Engine.” arXiv, April 7, 2016. http://arxiv.org/abs/1604.01891.

[2] Fuma Horie and Hideaki Goto. “Japanese Scene Character Recognition Using Random Image Feature and Ensemble Scheme:” In Proceedings of the 8th International Conference on Pattern Recognition Applications and Methods, 414–420. 2019. https://doi.org/10.5220/0007341904140420.

[3] Zhong Cao, Jiang Lu, Sen Cui, and Changshui Zhang. “Zero-Shot Handwritten Chinese Character Recognition with Hierarchical Decomposition Embedding.” Pattern Recognition 107 (November 2020): 107488. https://doi.org/10.1016/j.patcog.2020.107488.

[4] Jingye Chen, Bin Li, and Xiangyang Xue. “Zero-Shot Chinese Character Recognition with Stroke-Level Decomposition.” In Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, 615–621. 2021. https://doi.org/10.24963/ijcai.2021/85.

[5] Xinyan Zu, Haiyang Yu, Bin Li, and Xiangyang Xue. “Chinese Character Recognition with Augmented Character Profile Matching.” In Proceedings of the 30th ACM International Conference on Multimedia, 6094–6102. 2022. https://doi.org/10.1145/3503161.3547827.

[6] Zongze Chen, Wenxia Yang, and Xin Li. “Stroke-Based Autoencoders: Self-Supervised Learners for Efficient Zero-Shot Chinese Character Recognition.” Applied Sciences 13, no. 3 (January 30, 2023): 1750. https://doi.org/10.3390/app13031750.

[7] KanjiVG. https://kanjivg.tagaini.net/index.html

[8] cjklib. https://github.com/cburgmer/cjklib

[9] makemehanzi. https://github.com/skishore/makemeahanzi

[10] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation.” arXiv, May 18, 2015. http://arxiv.org/abs/1505.04597.

[11] Fuma Horie and Hideaki Goto. “Synthetic Scene Character Generator and Multi-Scale Voting Classifier for Japanese Scene Character Recognition.” In 2018 International Conference on Image and Vision Computing New Zealand (IVCNZ), 1–6. 2018. https://doi.org/10.1109/IVCNZ.2018.8634801.