Shape-aware Stroke Segmentation for Calligraphic Characters

Zibo Zhang
Shenzhen University
Shenzhen, China

Xueting Liu
Caritas Institute of Higher Education
Hong Kong SAR, China
tliu@cihe.edu.hk

Chengze Li
Caritas Institute of Higher Education
Hong Kong SAR, China

Huisi Wu
Shenzhen University
Shenzhen, China
hswu@szu.edu.cn

Zhenkun Wen
Shenzhen University
Shenzhen, China

Abstract

A stroke is the minimal unit of a character. The stroke extraction and classification of calligraphy characters are helpful to the digital use of characters. However, the existing instance segmentation methods have the problems of excessive segmentation and dependence on additional character encoding information. In this paper, we propose a novel supervised stroke segmentation method based on the information of stroke categories, which can apply stably to a variety of fonts. Extensive visual and quantitative experiments have been conducted to validate the effectiveness of our method. The results and statistical data show that the proposed method is superior to existing methods in stroke segmentation.

1. Introduction

Calligraphy is an artistic expression of the beauty of words. Calligraphy is widely used in the world, including Chinese, Japanese, and English. The variety of fonts is a good measure of the vitality and robustness of a language. As one of the languages with the largest number of users globally, Chinese have a rich font ecological environment. It is a common font innovation method to change the style of existing fonts very slightly. For example, making the edges and corners of a certain category of strokes round or sharp will make the new fonts present different artistic effects, as shown in Fig. 1. If the strokes of Chinese characters can be extracted and classified, it will simplify the creation method based on existing fonts. Currently, the analysis of the structure of Chinese characters (stroke extraction and classification) is usually confined to the most traditional and standard fonts, and it is inefficient and complicated to try to annotate all fonts manually. For many fonts of various styles, the automatic analysis of Chinese characters' structure is of great significance, which is beneficial to artistic creation, design and evaluation.

Figure 1. Fonts of the same calligraphic style ("Kaiti") may still have minor stylistic differences as composed by different artists.

(a) Input (b) Kim (c) Inoue (d) Ours (e) Ground Truth

Figure 2. Comparisons on different stroke segmentation methods. (b)&(c) The existing learning-based instance segmentation methods [7, 6] frequently obtain over-segmented results. (d) Our method significantly outperforms existing methods in stroke segmentation.

The traditional stroke segmentation methods segment the strokes by analyzing the continuity of the centerline or the boundary [9, 13, 15]. Recently, deep learning methods have been proposed to extract individual strokes based on instance segmentation networks [6, 14]. However, these methods usually implicitly identify pixels near and with similar features to be a stroke, which is quite error-prone and frequently results in over-segmentation (Fig. 2(b)&(c)). Besides, the existing methods usually only work on low-resolution inputs (64×64 or 128×128) or need additional information of character encode, which is difficult to apply to the wild.

There are two main challenges to this task. Firstly, The stroke segmentation algorithm should work on any size im-

age. Secondly, stroke segmentation should work with multiple fonts. With these two goals, a general instance segmentation method without prior knowledge generally cannot work well. Actually, we can utilize prior facts to help the segmentation to obtain better segmentation. In particular, calligraphic artists generally classify strokes into 32 stroke groups based on shapes. This stroke group information can definitely help for computers to better understand the character structure and thus assist the segmentation of individual strokes.

We first decompose the input image into stroke layers using a deep neural network. Each stroke layer contains a stroke belonging to a specific stroke category instead of directly decomposing input into individual strokes. Because different stroke categories have different appearance features, extracting pixels from stroke classes is easier and more reliable than extracting pixels from individual strokes. We use the U-net structure as the backbone of a stroke layer decomposition network to process different inputs to achieve stroke segmentation of calligraphy characters of any image size. Despite the effectiveness of stroke layer decomposition, each stroke layer must still be further decomposed into individual strokes. Therefore, we further propose a stroke extraction network to segment each stroke layer's individual strokes. Finally, to show the results as vector images, we use Potrace [12] to vectorize the segmentation results. Additionally, since the existing character dataset usually does not contain stroke group information, we proposed a novel iterative method for labeling characters with stroke groups in order to train our networks. Due to the limit of the labelled dataset, our stroke layer decomposition and stroke extraction networks are now trained on a specific Chinese calligraphic style. But our method could be easily adapted to other calligraphic styles with new training data.

To evaluate our method, we apply our method on test dataset and other different calligraphic styles. Results of qualitative and quantitative evaluation show that our method significantly outperforms the existing methods in stroke segmentation. Our main contributions can be summarized as follows:

1. We propose a new stroke segmentation method to distinguish different stroke groups based on stroke shape features.

2. We propose a novel method to segment the same category of stroke group according to the position relationship.

3. Our method enables the labelling of stroke groups for Calligraphic characters automatically, which brings the potential for other calligraphic applications.

2. Related Works

The main goal of this work is to classify and segment the strokes in Chinese characters. This task requires the algorithm to be able to perform structural analysis of Chinese characters. Since it is similar to this task, the related work on line drawing structure analysis can be provided as a reference. Human characters and drawings are usually composed of strokes, so there are many existing works that analyze the structure of characters and drawings through stroke segmentation.

2.1. Calligraphic Understanding and Stroke Representation

Template-based Stroke Decomposition The traditional stroke decomposition scheme matches characters or strokes to a predefined template database. The template may store information about the strokes, skeletons, and stroke point interconnection. When a specific character is queried, one can quickly obtain the decomposition result from the closest template characters [1]. Template matching can also enable downstream tasks such as font interpolation [10]. The limitation of these methods is also obvious: they are limited to processing only characters defined in the database. They are more likely to fail whenever the characters do not belong to the database, or whenever the query raster characters do not belong to a predefined calligraphic style.

Morphology-based Stroke Decomposition Multiple studies have promoted the morphological assumptions of strokes such as continuity, curvature, and interconnections to identify individual strokes and perform stroke decomposition. These methods can handle arbitrary characters without any pre-computation. In these works, [9] proposed a procedural solution. It first obtains the skeleton of a character using a thinning algorithm and refines the wrongly estimated junctions. Finally, the refined skeleton is over-segmented and reconnected according to a continuity-based criterion to reconstruct complete strokes. In parallel, [13] proposed a contour-based method by checking the convexity of the stroke groups. Assuming that the stroke groups are concave, the method subdivides stroke groups at the concave corner points and reconstructs each single stroke. In addition, [15] designed a hybrid stroke segmentation method by first obtaining the character structures through thinning and then registering the stroke segments with a pre-computed database. This database helps to improve the overall quality of the stroke cross-area (i.e., overlapping) estimation and leads to a better stroke segmentation. Although these methods manage to decompose arbitrary character stokes, they are limited by their procedural process nature and cannot well handle the characters with complex structures or with a large

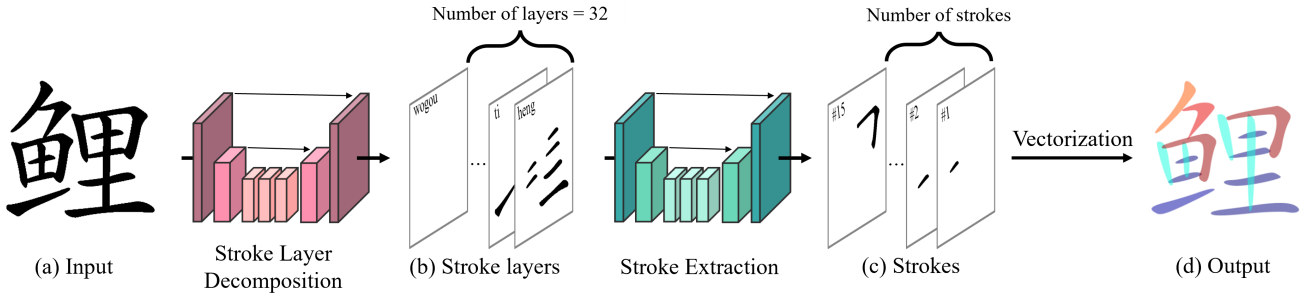


Figure 3. Overview. Our method feeds an input calligraphic character (a) into a stroke layer decomposition network to decompose the character into 32 stroke layers (b) where each stroke layer only contains the strokes belonging to the same stroke category. For each stroke layer, we further propose a stroke extraction model consisting of an iterative stroke extraction network to decompose each stroke layer into individual strokes (c). Finally, we use PoTrace [12] to vectorize individual strokes (d).

number of strokes. Moreover, many recent works on calligraphy stroke analysis has been proposed with the help of deep learning models. These methods benefit from the powerful capacity of learning models and are more robust and flexible due to the data-driven manner. [6] proposed a fast instance segmentation of strokes by adapting a convolutional neural network on calligraphy. [14] proposed another calligraphy parser by combining a semantic segmentation network together with a character encoding module. These methods show the potential of applying deep learning methods to the calligraphy parsing task. However, their methods are built upon standard object detection frameworks, which are not entirely applicable for non-realistic calligraphy images.

2.2. Line Drawing Segmentation and Vectorization

Besides those calligraphy-oriented methods, there are also methods dedicated to the segmentation of line drawings. Given that the minimal unit of representation for line drawings is also (ink) strokes like characters, we may find these methods helpful for our task. Kim et al. [7] proposed a stroke-by-stroke semantic segmentation method for line drawings and calligraphy by estimating the similarities in the overlapped stroke areas. However, it still cannot create smooth and complete segmentation around complex junctions. We will illustrate this problem in the comparison of result. Meanwhile, Guo et al. [3] aimed to achieve line drawing vectorization by detecting the junction points of complex drawings and reconstructing the stroke ordering based on a neural estimation. More recently, Egiazarian et al. [2] proposed an end-to-end solution that directly converts technical drawings into vector-based stroke representations. These two vectorization methods achieve reasonable stroke segmentation on line drawings. However, they may have issues in processing calligraphy images as the line widths of the strokes are much larger.

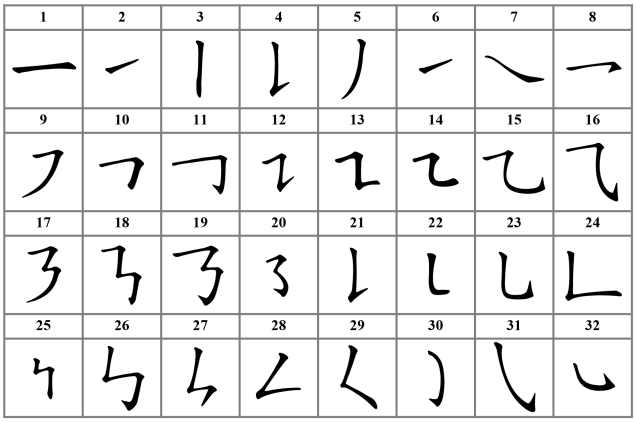


Figure 4. There are 32 commonly used calligraphic stroke categories. The name of each stroke category can be found in the supplementary materials.

3. Overview

An overview of our method is shown in Fig. 3. Given an input calligraphic character image (Fig. 3(a)), we first generate a number of stroke layers for it via the *stroke layer decomposition network*, each containing only strokes belonging to a specific stroke category (Fig. 3(b)). In this paper, we divide the calligraphic stroke into 32 categories. The mapping between the stroke layer index and the stroke category is shown in Fig. 4. We use a U-net-like [11] network architecture as our stroke layer decomposition network, which allows input with arbitrary resolution and outputs stroke layers with the same resolution. This network requires paired inputs and ground-truth stroke layers. However, the existing calligraphy dataset *Make Me a Hanzi* [8] only contains uncategorized strokes as the ground-truth. Therefore, in order to make the stroke layer decomposition network with a large amount of trainable data, we propose an iterative stroke classification network to label each stroke with one of th32 stroke categories.

Then we decompose each stroke layer into individual

Figure 5. Occasionally, strokes in the same stroke layer, i.e., belonging to the same stroke category, will overlap.

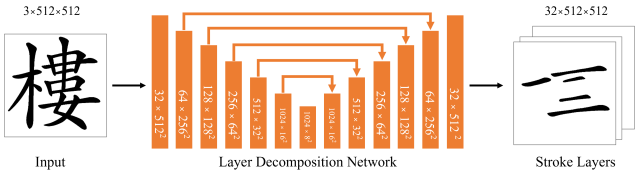


Figure 6. Network structure of the stroke layer decomposition network where the output contains 32 channels.

strokes (Fig. 3(c)). In the vast majority of cases, strokes belonging to the same stroke category (i.e., stroke layer) will not overlap in one calligraphic character. So, we use a simple flood-filling method to extract such individual strokes. Occasionally, strokes in the same stroke category will overlap, as shown by the black strokes in Fig. 5. To handle these overlapped strokes, we further propose a stroke separation network to separate each stroke layer image into individual strokes accordingly.

Finally, we use PoTrace [12] to vectorize the segmented strokes.

4. Approach

Our system consists of three parts, *stroke layer decomposition* to decompose the input calligraphic character image into 32 stroke layers, *stroke extraction* to separate individual strokes from each stroke layer, and *vectorization* to vectorize each individual stroke.

4.1. Stroke Layer Decomposition

Extracting strokes with large differences in appearance is a challenging task, especially when predicted with convolutional neural networks [7, 6]. As shown in Fig. 4, we classify the calligraphic strokes into 32 stroke categories, which is one of the general ways to classify commonly used characters in calligraphy. We observe that the strokes from different stroke categories have distinctly different appearances globally or locally, while the strokes from the same stroke category are mostly similar with only scale differences and minor deformation. Inspired by this, we propose to first decompose the input calligraphic characters into different stroke layers, each of which only contains the strokes from a specific stroke category. Since each stroke layer has its own unique stroke appearance features, identifying a stroke layer is much more robust and easier than identifying individual strokes. We propose a deep learning approach to solve this stroke layer decomposition problem.

Network Architecture The architecture of our stroke layer decomposition network is built on the U-net [11] as in Fig. 6. The network is fully convolutional with six downscaling levels and six upscaling levels, so the input of the network can be of any resolution. The output of this network has the same resolution as the input, but with 32 channels, each referring to a stroke category. For example, the input characters in Fig. 6 have five strokes from the first stroke category, so the first channel of their output should contain only these five strokes. The advantage of the 32 channels is that the overlap between the two strokes can be easily handled. The overlap of strokes is the most challenging part of the task because it falls into both stroke categories. We choose U-net over other networks for this stroke layer decomposition task because its receptive field is relatively large, which is important for identifying stroke features, especially when the stroke is large.

Training Dataset Preparation Once the idea of the stroke layer decomposition network is determined, the next problem to be solved is the preparation of the dataset since there exists no publicly available training dataset with stroke category labels. The existing calligraphy dataset *Make Me a Hanzi* [8] contains the stroke segmentation labelling for $9,574$ calligraphic characters. These Chinese characters belong to the same font and contain both simplified and traditional characters. By labelling the segmented strokes in this dataset with its stroke category information, we can build our dataset for the stroke layer decomposition network. The original dataset was divided into the training set and test set in a ratio of 9:1. Our synthesized dataset was generated following the same division. There are altogether $112,617$ strokes for the characters in this dataset, so manually labelling these strokes is extremely tedious and time-consuming. For fast and accurate labelling, we propose to classify the stroke category of each stroke via an iterative stroke category classification network.

We use ResNet [5] as the backbone of our classification network. To start training, we first manually select 8 strokes for each stroke category. If some stroke categories were very rare (less than 8 strokes in all characters), we select all strokes in such stroke categories. Finally, we obtain 300 strokes and their stroke category label as our initial training dataset to train our stroke category classification network. After the training, we move all strokes with high confidence predictions (maximal stroke category prediction > 0.9) and their predicted stroke category into the training dataset and then trained the stroke classification network

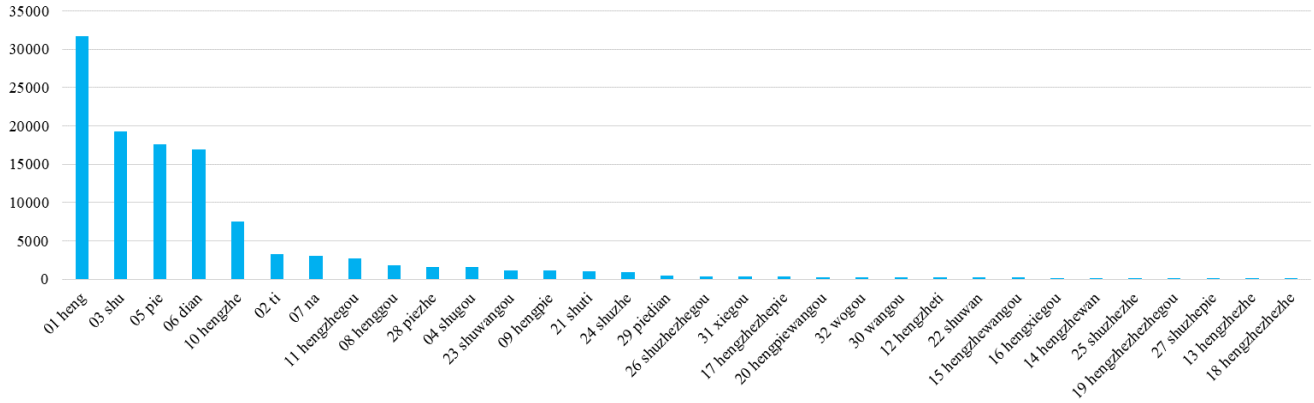


Figure 7. The number of different stroke categories.

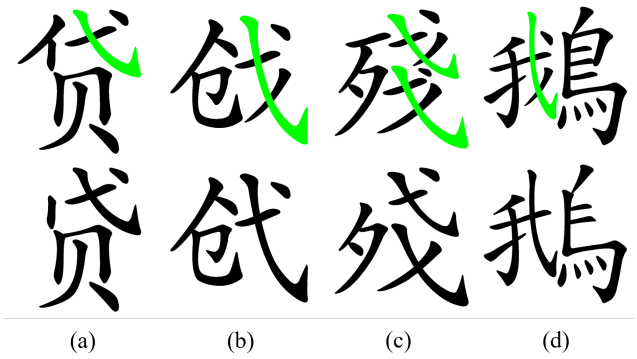


Figure 8. We generate fake calligraphic characters by randomly removing frequently appeared strokes from characters that contain less frequently appeared strokes. The first line is the original character, and the red strokes are the less frequently appear strokes. The second line is the fake character.

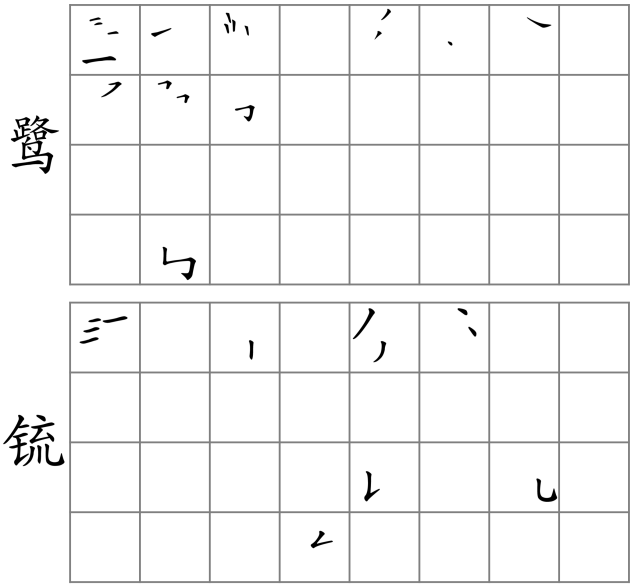


Figure 9. Examples of stroke layer decomposition which decompose an input calligraphic character into 32 stroke layers.

again based on the new training data. As this process continues, the training set will get larger and larger, and the number of cases without high confidence strokes will get smaller and smaller. The iteration is terminated when no high-confidence strokes are added at a certain iteration. After the completion of the iteration, there are very few strokes ($< 0.01\%$) that do not obtain enough confidence level. For these very few cases, we will perform manual classification. The number of strokes in each predicted stroke category can be found in Fig. 7.

In addition, we can see that the frequencies of the different stroke categories are clearly unbalanced. Based on statistics in Fig. 7, the stroke category with the largest frequency is the first stroke category, which contains $31,591$ strokes in all calligraphic characters in our training dataset. In contrast, the stroke category with the lowest frequency is the 18th stroke category, which contains only 1 stroke in all characters. This makes the network not quite robust to segmentation of these less frequently appeared stroke categories. Therefore, we first propose to augment our training data by introducing fake characters that contain less frequently appeared stroke. To compose characters that follow general character structure, we randomly select some characters that contain less frequently appeared strokes and randomly remove several frequently appeared strokes. Fig. 8 shows several fake and orignal characters containing the 31st stroke. By introducing fake characters to our training dataset, the robustness of the segmentation result can be significantly improved, especially for stroke layers that do not appear frequently.

Implementation Details We conducted all of our experiments using PyTorch on an NVidia GeForce RTX 2080Ti with 11GB of memory. Our model is optimized using RMSprop with a weight decay of 1e-8, momentum of 0.9, and batch size of 16. We train our model with the image dimen-

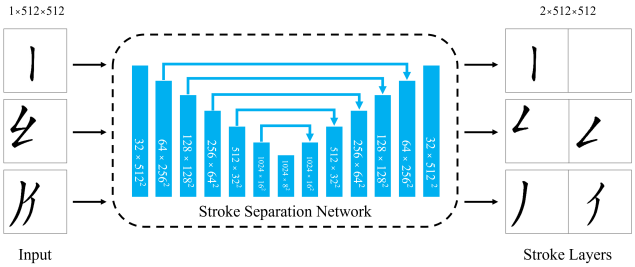


Figure 10. Network structure of the stroke layer decomposition network where the input contains one channel and the output contains two channels.

sion of 512^2 for 40k iterations, which lasts about 9 hours. The initial learning rate is set to 0.0001, and the learning rate is reduced with the plateau strategy. We adopt binary cross-entropy as the loss function. Two examples of our stroke layer decomposition result is shown in Fig. 9.

4.2. Stroke Separation and Vectorization

While stroke layers are simpler to be identified than single strokes, a stroke layer may contain multiple strokes of the same stroke category. As a result, we need further localize a single stroke from each stroke layer. We use the flood-fill method to extract all connected black components from the stroke layer because most strokes in a stroke layer will not overlap. Each connected component is considered a potential stroke. Output noise is identified as potential strokes with areas smaller than 0.0001 times the total area of the image and will be removed directly. Then, for each potential stroke image, we feed it into the stroke separation network, which outputs two stroke images, each with one separated stroke. If the potential stroke has only one stroke, one of the output stroke images should be empty. When more than two strokes are overlapping, the stroke separation network will output one image containing one stroke and another image containing all other strokes based on the position relationship. The stroke separation network will then be fed the output two stroke images until all stroke images contain only one stroke.

As shown in Fig. 10, we adopt a network architecture for our stroke separation network that is similar to the previously presented stroke layer decomposition network. The difference is that the input has one channel, and the output has two channels. We use the same training strategy as our previous network, as presented in the previous subsection. According to our data, 1.68 percent of strokes overlap with strokes of the same categories. Only 3 of the 32 categories have multiple cases of overlapping. In the overlapping case, 99.07 percent of the overlapped strokes are composed of two strokes, while 0.93 percent of them contains three strokes. As a result, we only prepare input images that contain two or three overlapped strokes from the same stroke category in our prepared training data, which includes 825 images of two strokes and 5 images of three strokes. We also prepare 940 images of a single stroke to ensure that our network outputs the accurate result for single-stroke images. Fig. 10 shows some examples of outputs from our stroke separation network.

Finally, we vectorize the segmentation output using Potrace [12] and use the same parameters throughout the vectorization process.

Table 1. Compared with different segmentation methods (64×64).

	[Kim et al. 2018]	[Inoue and Yamasaki 2019]	Ours
Stroke IOU	0.958	0.926	0.975
Layer IOU	-	-	0.992

5. Experimental Results

5.1. Qualitative Analysis

We first visually compare our stroke segmentation results to those of two state-of-the-art methods, Kim's method [7] and Inoue's method [6]. Some examples are shown in Fig. 13, where we can see that segmentation of the overlapping regions among different strokes is critical. Even after dividing the stroke segmentation into path segmentation and stroke intersection segmentation, Kim's method fails to achieve smooth enough line segmentation for the difficult characters. In addition, without excellent path segmentation, its stroke intersection segmentation cannot achieve high accuracy, as shown in Fig. 13(b) by the obvious disconnection of the brushstrokes and the abrupt intersection segmentation. Inoue's method further divides the strokes into foreground and background and uses the Mask-RCNN [4]to segment different strokes to reduce the problem of stroke disconnection. However, without an efficient mechanism to control and refine the intersections among different brushstrokes, Inoue's method cannot precisely extract the overlapping regions (Fig. 13(c)). Unlike previous methods, our method can preserve the integrity of each stroke, even when the strokes from different categories have clearly different appearances globally or locally, resulting in better stroke segmentation results (Fig. 13(d)).

On the other hand, we also applied our method to segment more challenging characters with new fonts to demonstrate the effectiveness of our method. Several typical selected fonts are shown in Fig. 11. The fonts in the left four columns differ only slightly from our dataset, and the font styles in the right four columns differ even more. Since segmentation by stroke category is more in line with the writing rules of Chinese characters, our method obviously can achieve accurate stroke segmentation results, which also proves that segmentation by stroke categories is a robust and feasible way.

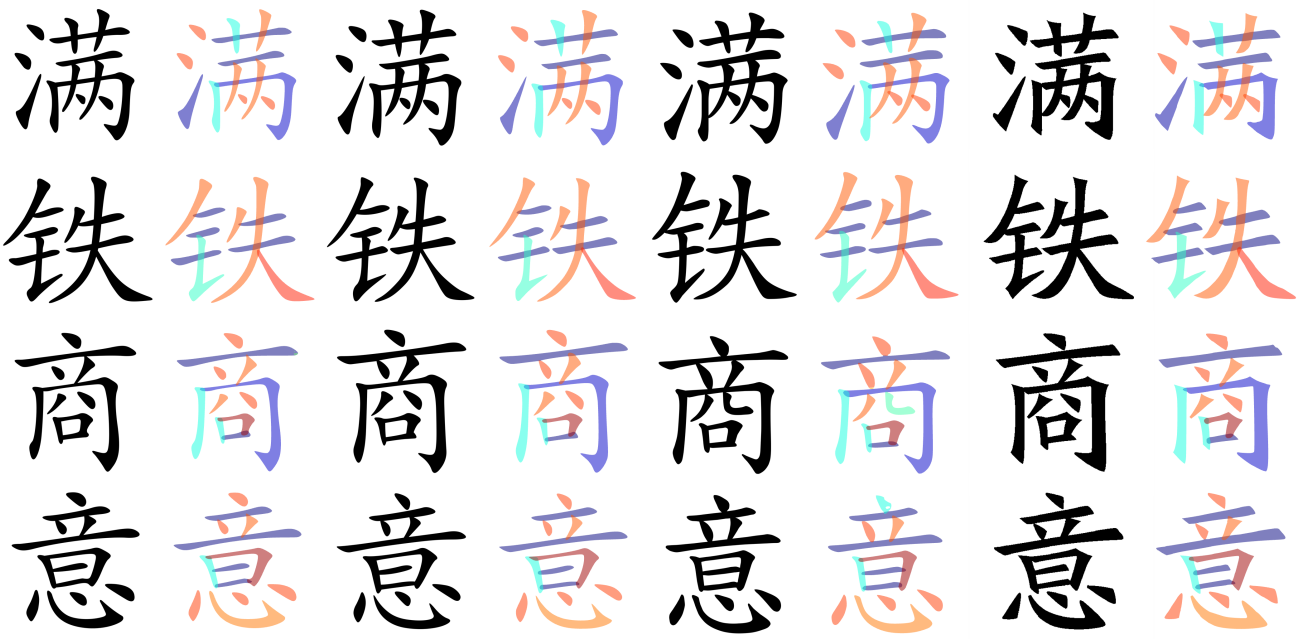


Figure 11. Compare different fonts for different segmentation methods. The resolution of the input is 512 × 512 for all examples.

Table 2. Segmenting overlap strokes in different image sizes.

	64 × 64	128 × 128	512 × 512
Layer IOU	0.9732	0.9836	0.9706
Stroke IOU	0.9818	0.9894	0.9834

5.2. Quantitative Analysis

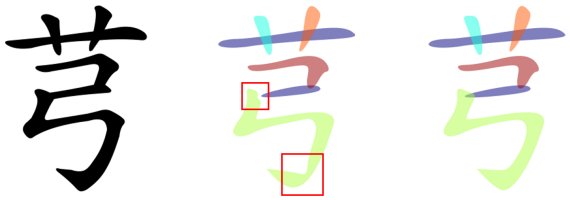
To quantitatively compare our stroke segmentation method with other state-of-the-art methods, we further performed a statistics comparison based on two metrics, including "Stroke IOU" and "Layer IOU". For each segmented pixel path, we find the closest path in the ground truth (i.e., the intersection area between them is the largest among all paths) and compute the ratio of the number of pixels in the intersection divided by the number in the union. Therefore, we can formulate the "Stroke IOU" as following,

$$IOU_{stroke} = \frac{1}{N}\sum_{i}^{N} max\left(\sum_{j}^{N}\frac{P_i \bigcap G_j}{P_i \bigcup G_j}\right) \quad (1)$$

where N represents the number of categories of strokes. P_i is the model's prediction of category i strokes, and G_j is the ground truth of the j category strokes.

Similarly, we can employ the second comparative metric, "Layer IOU" to evaluate the performance of our method on stroke classification as following,

$$IOU_{layer} = \frac{1}{N}\sum_{i}^{N}\frac{|P_i \bigcap G_i|}{|P_i \bigcup G_i|} \quad (2)$$



(a) (b) (c)

Figure 12. Before and after comparison of fake characters is introduced. (a) Input. (b) Output with no fake characters introduced. (c) Output with fake characters introduced. With the introduction of false characters, overlaps are more accurate, especially for less frequent stroke categories.

where N represents the number of categories of strokes. P_i is the model's prediction of category i strokes, and G_i is the ground truth of the i categories strokes. Note that the other methods we compared only identify and segment individual strokes, so "Layer IOU" is not applicable for these methods. The statistics results are as shown in Table 1. We can also easily observe that our method generally outperforms all competitors in "Stroke IOU" metrics. In addition, we also measure the impact of different resolutions on our stroke segmentation. As shown in Table 2, we can still obtain satisfied "Stroke IOU" and "Layer IOU" values for segmenting overlap strokes in three different image sizes.

Table 3. Different image sizes with vs without fake characters.

	64 × 64	64 × 64 + fake	128 × 128	128 × 128 + fake	512 × 512	512 × 512 + fake
Layer IOU	0.9900	0.9920	0.9912	**0.9934**	0.9812	0.9861
Stroke IOU	0.9680	0.9750	0.9756	**0.9814**	0.9645	0.9756

5.3. Importance of Fake Data

We created 4,382 fake characters and tested them at different image sizes to measure their impact on our performance. As shown in Table 3, fake characters improve the performance of different metrics and image sizes. In terms of metrics, for each character, the weight of an infrequent stroke in layer IOU is 1/32, while the weight of stroke in stroke IOU is 1/(Number of Strokes). According to our statistics, the average number of strokes in each character is 11.8, which is significantly smaller than the number of stroke layers (i.e. 32), so the increment of stroke IOU is more significant than the increment of layer IOU. In terms of image sizes, the size of 64 × 64 and 128 × 128 have similar increments, while the size of 512 × 512 has a larger increment due to the increased difficulty in segmentation. A visual comparison is presented in Fig. 12. The segmentation result was evidently improved, especially in the stroke overlapping area. In summary, it is evident from our statistical results that fake characters can significantly improve the performance of the stroke layer decomposition network, indicating the importance of fake data for training a better segmentation network.

6. Conclusion

We proposed a novel supervised deep learning approach based on stroke category information which first decomposes the input character into stroke layers by shape features and then decomposes each stroke layer into individual strokes. While our current stroke segmentation network is trained on a single calligraphic style due to the limitation of the publicly available training dataset, our method should be able to be applied to more calligraphic styles provided by the dataset. We will continue our research in this direction in the future.

References

[1] X. Chen, Z. Lian, Y. Tang, and J. Xiao. An automatic stroke extraction method using manifold learning. In *38th Annual Conference of the European Association for Computer Graphics, Eurographics 2017 - Short Papers, Lyon, France, April 24-28, 2017*, pages 65–68. Eurographics Association, 2017. 2

[2] V. Egiazarian, O. Voynov, A. Artemov, D. Volkhonskiy, A. Safin, M. Taktasheva, D. Zorin, and E. Burnaev. Deep vectorization of technical drawings. In A. Vedaldi, H. Bischof, T. Brox, and J. Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XIII*, volume 12358 of *Lecture Notes in Computer Science*, pages 582–598. Springer, 2020. 3

[3] Y. Guo, Z. Zhang, C. Han, W. Hu, C. Li, and T. Wong. Deep line drawing vectorization via line subdivision and topology reconstruction. *Comput. Graph. Forum*, 38(7):81–90, 2019. 3

[4] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 6

[5] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 4

[6] N. Inoue and T. Yamasaki. Fast instance segmentation for line drawing vectorization. In *Fifth IEEE International Conference on Multimedia Big Data, BigMM 2019, Singapore, September 11-13, 2019*, pages 262–265. IEEE, 2019. 1, 3, 4, 6, 9

[7] B. Kim, O. Wang, A. C. Öztireli, and M. H. Gross. Semantic segmentation for line drawing vectorization using neural networks. *Comput. Graph. Forum*, 37(2):329–338, 2018. 1, 3, 4, 6, 9

[8] S. Kishore. Make me a hanzi. 3, 4

[9] H. Leung and H. Kang. Stroke extraction for chinese calligraphy images. In *Asia Pacific Workshop on Visual Information Processing (VIP2005)*, pages 223–229, 2005. 1, 2

[10] Z. Lian and J. Xiao. Automatic shape morphing for chinese characters. In Z. Zhang and Z. Li, editors, *SIGGRAPH Asia 2012 Technical Briefs, Singapore, November 28 - December 1, 2012*, pages 2:1–2:4. ACM, 2012. 2

[11] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 3, 4

[12] P. Selinger. Potrace: a polygon-based tracing algorithm. *Potrace (online), http://potrace. sourceforge. net/potrace. pdf (2009-07-01)*, 2, 2003. 2, 3, 4, 6, 9

[13] Y. Sun, H. Qian, and Y. Xu. A geometric approach to stroke extraction for the chinese calligraphy robot. In *2014 IEEE International Conference on Robotics and Automation, ICRA 2014, Hong Kong, China, May 31 - June 7, 2014*, pages 3207–3212. IEEE, 2014. 1, 2

[14] W. Wang, Z. Lian, Y. Tang, and J. Xiao. Deepstroke: Understanding glyph structure with semantic segmentation and tabu search. In *MultiMedia Modeling - 26th International Conference, MMM 2020, Daejeon, South Korea, January 5-8, 2020, Proceedings, Part I*, volume 11961 of *Lecture Notes in Computer Science*, pages 353–364. Springer, 2020. 1, 3

(a) Input (b) Kim (c) Inoue (d) Ours (e) Ground-truth

Figure 13. Comparison with different stroke segmentation methods. The resolution of the input is 512 × 512 for all examples. (b)&(c) [7] and [6] are prone to errors at the intersection of strokes. (d) Our method can obtain much more correct and precise segmentation results, especially at junctions. We use Potrace [12] to vectorize the segmentation results. Similar to the other methods, the vectorization process slightly smooths the contour of the strokes and leads to minor shape change between raster and vector characters, as shown in the green boxes in the last row.

[15] Z. Xu, Y. Liang, Q. Zhang, L. Dong, and E. Izquierdo. Decomposition and matching: Towards efficient automatic chinese character stroke extraction. In *2016 Visual Communications and Image Processing, VCIP 2016, Chengdu, China, November 27-30, 2016*, pages 1–4. IEEE, 2016. 1, 2