Multiscale fully convolutional network-based approach for multilingual character segmentation

Chao Yu | Jin Liu | Yunhui Li

Shanghai Maritime University, Shanghai, China

Correspondence
Jin Liu, Shanghai Maritime University, Shanghai, China.
Email: jinlu@shmtu.edu.cn

Funding information
National Natural Science Foundation of China, Grant/Award Number: (61872231, 61701297)

Abstract
Character segmentation is a challenging task for optical character recognition systems. Traditional methods usually utilize rule-based algorithms but most of them are not applicable in modern intelligent recognition applications that require high accuracy. It is especially the case for text containing Eastern Asian language characters with complex pictograph structures, such as Chinese. To alleviate this problem, this study proposes an encoder–decoder structure-based multiscale fully convolutional network (MSFCN) model for optical character segmentation. Comparing with other methods, MSFCN can not only effectively extract semantic details from images but also exploit boundary information of intervals between characters, thereby distinguishing characters from a background in pixel level. Extensive experiments have been conducted on two benchmark data sets of ICDAR2013 and MLCS. Obtained results prove that MSFCN achieves state-of-the-art segmentation performance and indicated its practical application value.

1 | INTRODUCTION

Computer vision has been extensively studied for many years [1, 2]. Nevertheless, there are still challenging unsolved problems, such as recognizing text with a large-scale vocabulary of multiple languages, or handwritten text. For optical character recognition applications, character segmentation is a crucial task in the recognition process [3]. The accuracy of segmentation has a significant impact on a variety of applications, such as the precision of single character recognition. Among the various writing systems, Chinese has more than 3000 frequently used characters. Thus there is no standard or universal methodology for extracting features for Chinese character segmentation. Besides, a single Chinese character may consist of multiple characters, so its structure can be very complicated. Thus, Chinese character segmentation becomes much more difficult than the Latin alphabet and Arabic numbers. Furthermore, the length of intervals between two characters in different languages varies irregularly. As shown in Figure 1, this multilingual text includes three types of character intervals with distinct features. Segmenting multilingual text is a challenging task.

To date, the mainstream character segmentation methods are morphology-based segmentation, clustering-based segmentation, and recognition-based segmentation [4, 5]. The morphology-based method cannot handle noise well. The recognition-based method requires multiple recognition of different parts of the image and consumes much time. The clustering-based method has inferior robustness to the uneven greyscale image. Recently, deep neural networks can perform automatic feature learning on raw data, and have significantly improved the machines' learning ability in various fields [6, 7]. Nevertheless, previous deep-learning-based methods for character segmentation tend to overcut the characters into their subcomponents [8–10]. Therefore, the accuracy and efficiency of such neural network-based methods are still far from satisfactory in real-world applications.

In this article, we propose a novel multiscale fully convolutional network (MSFCN) to perform segmentation for multilingual text containing Chinese and other types of characters. By scaling the original image to ones with multiple sizes, we can feed our model with more detailed feature information of raw inputs. Besides, by adopting an encoder–decoder structure and combining the scaled features with the intermediate features generated by the model, the accuracy of the model is further improved. Furthermore, we use the connected components search algorithm to optimize results generated by a deep network, then obtaining the final segmentation results.
The experimental results show that our model has better performance than others. The structure is organized as follows: Section 2 introduces the related work of character segmentation. In Section 3, we describe the method of the MSFCN and present our novel improvement. Then we present experiment environments and results on the multilingual character segmentation (MLCS) dataset in Section 4. In Section 5, we evaluated our model using the ICDAR 2013 handwritten dataset. And at last, we draw some conclusions in Section 6.

2 | RELATED WORK

2.1 | Rule-based methods

Research on character segmentation can be traced back to the 60s of the last century. At that time, researchers from the United States focused mainly on printed fonts on newspapers or bank cheques. In 1996, Casey and Lecolinet published a summary of character segmentation methods [3] and introduced some outstanding researches at that time. The article mentions three kinds of character segmentation strategies: The first strategy attempts to partition the input image into sub-images to be classified later. The second strategy attempts to avoid premature image segmentation and segments the image either explicitly by classification of specified windows, or implicitly by classification of subsets of spatial features collected from the whole image. The third strategy is a hybrid of the first two, using dissection together with recombination rules to define potential segments. After that, rule-based methods such as the connected domain method [11] and the watershed segmentation method [12] were gradually proposed. The latest rule-based method was proposed by Inkeaw et al. in 2018. They proposed a conventional method based on over-segmentation for Lanna Dhamma characters segmentation with a multilevel writing style [5].

2.2 | Machine-learning-based methods

However, rule-based methods cannot achieve satisfactory performance when facing various real-world applications. In the meantime, with the development of machine learning, researchers are turning to this new approach. In 1994, Seni et al. had a relatively complete definition of metrics for clustering segmentation methods [13]: (1) bounding box method, (2) Euclidean method, (3) minimum run-length method, (4) average run-length method and (5) run-length with Euclidean heuristic method. In 2009, Louloudis et al. [14] used methods 1 and 5, together with the convex hull distance method. They divide text lines into two categories, one is the segmentation between two words, and the other is the segmentation of a single word. After that, Louloudis used the hierarchical clustering method, and their methodology achieved a detection rate (DR) of 90.4% and a recognition accuracy (RA) of 90.6%.

Recently in 2015, Ryu et al. found that in some traditional segmentation methods, many connections between segmentation lines are often ignored [4]. Thus, they turn the problem into a mixed integer quadratic programming problem. By using the cutting plane method to find the optimal solution, they have about 90% accuracy in the experimental data given by international conference on document analysis and recognition (ICDAR) in 2009 and 2013. In this way, a lot of parts are systematically segmented, but they do not consider the contents of each part. Xu et al. proposed an algorithm based on the hidden Markov model in the recognition algorithm [8]. After preprocessing the picture, they use an HMM recognizer to process the image into a sequence of the field images. Then dynamic programming method (such as the Viterbi algorithm) is used to find the optimal recognition results. Although the segmentation points can be selected dynamically, the time of segmentation takes too long while the accuracy is not as good as previous methods.

2.3 | Deep-learning-based methods

As the deep-learning-based methods become popular, they have been applied to the task of text segmentation too. Srijari et al. used a three-layer neural network [15] in segmentation for Arabic characters. They used a connected domain method for rough segmentation. Then two parts of a word appear in Arabic, so connected domains with a closer distance can be combined into a cluster. In 2012, Wang, T. et al. leveraged the representational power of multilayer neural networks [16], and a common framework to train highly accurate text detector and character recognizer modules. They integrated these two modules into a complete end-to-end, lexicon-driven, scene text recognition system that achieves state-of-the-art performance on standard benchmarks. Chen et al. address the task of semantic image segmentation with deep learning [17]. They highlighted convolution with upsampled filters (atrous convolution) as a powerful tool in dense prediction tasks and proposed atrous spatial pyramid pooling to segment objects at multiple scales.

At the same time, a new optimizer method Adam is proposed [18]. This has also been observed independently by Dai et al. [19]. They presented region-based fully convolutional networks (FCN) for accurate and efficient object detection.
Their region-based detector is an FCN with almost all computations shared on the entire image. They proposed position-sensitive score maps to address a dilemma between translation invariance in image classification and translation variance in object detection. FCN is also an effective segmentation method so far [20]. Yang et al. developed a deep learning algorithm for contour detection with an FCN network [21]. Different from previous low-level edge detection, their algorithm focuses on detecting higher-level object contours. Zhang et al. presented an FCN model [22] to predict the salient map of text regions holistically. Noh presented a novel semantic segmentation algorithm by training a deep deconvolution network on top of the convolutional layers adopted from the VGG 16-layer net [23]. The deconvolution network is composed of deconvolution and unpooling layers. It identifies pixel-wise class labels and predicts segmentation masks. Long et al. built an FCN that takes an input of the arbitrary size and produces correspondingly sized output with efficient inference and learning [24]. They adapt deeper classification networks (AlexNet, the VGG net and GoogLeNet) into FCN and transfer their learnt representations by fine-tuning to the segmentation task. In 2018, Yang et al. proposed a useful attention mechanism for convolutional neural network (CNN) models in image classification tasks [25]. In 2019, Li et al. [26] proposed a progressive scale expansion algorithm and combined it with a feature pyramid network (FPN) [27] to obtain a novel framework for a shape robust text detection. In 2019, Baek, Y. et al. [9] proposed Character Region Awareness for Text Detection (CRAFT), which focused on combing characters and affinity between characters into text lines. They used this method to detect curve text line in scene image and achieve a good performance. In the same year, Zhang et al. [10] proposed another method named look more than once (LOMO) for text detection. The LOMO consists of a direct regressor, an iterative refinement module and a shape expression module for text proposal localization, text refinement and a flexible shape expression.

However, there are still no examples of applying semantic segmentation to character segmentation tasks. That is because the result of semantic segmentation cannot directly represent the result of character segmentation. Further processing of the predicted results is required to obtain the character segmentation results. Besides, these existing methods use the convolution of a single scale in feature extraction and ignore the intervals information between characters, which lead to the unsatisfactory cutting effect. And the problem of MLCS remains challenging, and the neural network models need to be further optimized.

3 | METHODOLOGY

3.1 | Problem definition

Most character segmentation tasks use rule-based algorithms like greyscale histograms to extract spatial information from images. Although these methods can effectively capture morphological features, the semantic information for characters is always ignored, which leads to errors in the segmentation results. Thus, we regard character segmentation as a supervised pixel-wise classification task which can be defined as the following:

\[ H = \zeta(I) \]  

(1)

where \( I \) denotes the input RGB image with length \( b \) and width \( w \), \( \zeta \) denotes the semantic segmentation model and \( H \) denotes the two-channel heat map as the final output. Our final goal is to build a semantic segmentation model to predict the heat map \( H \). The definition of \( I, H \) is described as the following:

\[ I = \left\{ p_{b,w}(r, g, b) \mid b > 0, w > 0 \right\} \]  

(2)

\[ H = \left\{ p_{b,w}(t, f) \mid b > 0, w > 0 \right\} \]  

(3)

where \( p_{b,w}(t, f) \) denotes the value of pixel at \((b, w)\). If the pixel belongs to text region, \( t = 1, f = 0 \) and vice versa.

Next, the results of the heat map will be converted to the results of character segmentation. The definition of the function is shown as follow:

\[ R = \xi(H) \]  

(4)

\[ R = \left\{ p_{b,w} \mid b > 0, w > 0, p_{b,w} \in \mathbb{N} \right\} \]  

(5)

The value of \( R \) is a natural number denotes which character region that pixel belongs to.

3.2 | MSFCN

Typical CNN models consist of convolutional layers, pooling layers, activation layers, dropout layers and fully connected layers. Convolutional layers convolve input feature maps or input images with some linear filters. The procedure can be denoted as follows:

\[ (r_k)_{ij} = (F_k \cdot M_k)_{ij} + b_k \]  

(6)

where \( r_k \) denotes the result of convolutional process, \( F_k \) denotes the \( k \)-th filters, \( M_k \) denotes the \( k \)-th feature maps and \( b_k \) denotes the bias of this \( k \)-th filter. Activation function does the nonlinear transformation to the input to make it learn for more complex patterns. As a typical representative of activation function, rectified linear units (ReLU) can be used in neural networks such as CNN [21, 25, 28, 29]; it is defined as follows:

\[ f(x) = \max(0, x) \]  

(7)
Dropout layers randomly ignore some units in model in order to avoid overfitting and save the model's training time. Fully connected layers calculate the outputs as follows:

\[ y_k = \sum_i W_{ki} x_i + b_k \]  \hspace{1cm} (8)

where \( y_k \) denotes the \( k \)-th output neuron, \( x_i \) denotes the \( i \)-th input neuron, \( W_{ki} \) denotes the weight connecting \( x_i \) with \( y_k \) and \( b_k \) denotes the bias. Usually, a CNN model has several fully connected layers in the end of the model to output the labels like one-hot codes in the classification task or other type of labels in other tasks.

In the CNN model, spatial information will be lost because of flattening operation, thereby it is not suitable for segmentation tasks. Thus, FCN architecture is proposed by eliminating fully connected layers and adding convolutional layers with \( 1 \times 1 \) filter as an alternative. After feature extraction, deconvolutional layers are used to make the output have the same dimensions as the input. The inputs of traditional FCN models are images that can be regarded as a matrix while the outputs remain the same shape. To ensure this feature, FCN needs to be properly designed. Usually, the segmentation maps have a size of \( k \times w \times h \) for \( k \) different objects, where \( w \) and \( h \) are the width and height of input image shape, respectively. Typical FCN uses cross-entropy as a loss function which is shown as follows:

\[ C = -\frac{1}{n} \sum_x [y \ln a + (1 - y)\ln(1 - a)] \]  \hspace{1cm} (9)

where \( n \) denotes the number of output neurons, \( y \) denotes the results of the model output and \( a \) denotes the results of the label. FCN architecture can be used for semantic segmentation. By using the whole images as input and heat maps as labels, the FCN models can be trained end-to-end and can produce full-pixel segmentation results.

As mentioned above, FCN solves the problem of fixed input image size and restoration of the feature image size that CNN suffers. However, FCN can ignore detailed features while its receptive field is increased through the pooling layers. Such a defect will result in the loss of a lot of edge information in the final recognition result of the model, which will lead to the performance degradation of the model. U-net fuses the features in the encoder with the corresponding features of the decoder to reduce the impact of the down-sampling operation. However, some of the image details have already lost in the deep layers of the encoder. Therefore, we proposed a novel network architecture, MSFCN. MSFCN can be divided into two parts, encoder and decoder. The former is mostly composed of convolutional layers for feature extraction. The latter is composed of deconvolution layers and up-sampling layers for feature reduction.

Figure 2 shows our model structure. It can be found that our model is an image-to-image conversion model. In convolutional layers and deconvolutional layers, the ReLU is selected as activation layers. Compared to the sigmoid-like activation function, ReLU is a nonlinear function that is easy to calculate.

Figure 3 shows a simplified structure of MSFCN. The feed-forward process can be divided into three branches, main branch (marked as blue), intermediate feature fusion branch (marked as green) and multiscale input feature extraction branch (marked as blue). The main branch is an encoder-decoder structure composed of a stack of sampling units and up-sampling units. All the sampling units contain a pooling.
layer and several convolutional layers. Correspondingly, the up-sampling units contain an unpooling layer and several transpose convolutional layers. In this branch, input features for all units come from the previous units. Thereby solely using it will lead to a large amount of loss of character edge feature information of the input image. Intermediate feature fusion branch combines the features from the encoder with the input feature of the decoder is added to discount the feature loss. To make our model more sensitive to the features of intervals, an input feature fusion branch is used to concatenate features directly from the input image. The scaled input is concatenated to feature maps both from convolutional layers and deconvolutional layers. Thus, all sampling and up-sampling units will get the low-level edge features of the intervals between characters.

Our original inputs are image with size of 256 × 256 × 3, because an image in such size can be down-sized by three times keeping the integer size. And the scaled image remains a certain resolution, then, we use an algorithm to make the image feature maps, shown by scale ‘n’×. Table 1 shows the size of these feature maps.

To fuse the semantic features from different level, we combine feature maps from encoder, decoder and scaled input to a concatenate layer \( \text{Concat}(\cdot) \) which can be represented as:

\[
\text{Output} = \text{Concat}(F_e, F_d, \text{Input})
\]

\[
\text{Concat}(F_e, F_d, \text{Input}) = F_e \oplus F_d \oplus P(\text{Input})
\]

where \( \oplus \) denotes the channel-wise concatenate operation and \( P(\text{Input}) \) refers to max pooling operation. The kernel size of pooling layer is not always the same and is demonstrated in Table 2.

There are two concatenate layers. Table 2 shows the shape of input and output of concatenate layers.

After these two concatenate layers, we add another two convolutional layers, a max pooling layer to turn the feature maps into size of 128 × 128 × 128, after that, two convolutional layers and a max pooling layer are added and finally the features maps become the size of 64 × 64 × 256. Table 3 shows the detailed process.

Table 4 shows the detailed information of decoder module. Finally, the output’s size is the same as the label, it will be origin width × origin height × class numbers in our model. we calculate the loss and update it to the whole model.

The FCN model we used is not the most original one [24]. It is more like a Segnet based on the encoder-decoder structure proposed by Badinayarayan [28] or a U-net proposed by Ronneberger [29]. The FCN models will keep the two-dimensional feature maps with a deconvolutional process. In MSFCN, based on the FCN model, we concatenate detailed texture features with semantic features and in both the encoder part and decoder part to fuse more features of intervals. So, the irregular intervals between multilingual characters can be captured by multiple receptive fields. It makes MSFCN more sensitive in the intercharacter intervals such as the distance between two characters.

†TABLE 1. The shape of scale ‘n’×

| Feature name | Feature size |
|--------------|-------------|
| Scale 1×     | 256 × 256 × 3 |
| Scale 2×     | 128 × 128 × 3 |
| Scale 3×     | 64 × 64 × 3  |

TABLE 2. The parameters of fuse route

| Layers | Encoder feature | Decoder feature | Output size |
|--------|-----------------|-----------------|-------------|
| Concat 1 | 128 × 128 × 32 | 128 × 128 × 32 | 128 × 128 × 64 |
| Concat 2 | 64 × 64 × 64   | 64 × 64 × 64   | 64 × 64 × 128 |

TABLE 3. The parameters of encoder module

| Layers | Input size | Output size |
|--------|------------|-------------|
| Conv 1 | 256 × 256 × 3 | 256 × 256 × 128 |
| Conv 2 | 256 × 256 × 128 | 256 × 256 × 128 |
| Max pooling | 256 × 256 × 128 | 128 × 128 × 128 |
| Conv 3 | 128 × 128 × 128 | 128 × 128 × 256 |
| Conv 4 | 128 × 128 × 256 | 128 × 128 × 256 |
| Max pooling | 128 × 128 × 256 | 64 × 64 × 256 |

TABLE 4. The parameters of decoder module

| Layers | Input size | Output size |
|--------|------------|-------------|
| Upsampling | 64 × 64 × 256 | 128 × 128 × 256 |
| Conv 1 | 128 × 128 × 256 | 128 × 128 × 256 |
| Conv 2 | 128 × 128 × 256 | 128 × 128 × 256 |
| Upsampling | 128 × 128 × 256 | 256 × 256 × 256 |
| Conv 3 | 256 × 256 × 256 | 256 × 256 × 128 |
| Conv 4 | 256 × 256 × 128 | 256 × 256 × 128 |
| Max pooling | 128 × 128 × 256 | 64 × 64 × 256 |
3.3 | Multiscale feature extraction

The essential problem of the character segment is that it's very difficult to define an appropriate length threshold for the intervals between two neighbouring characters in the same row. Moreover, compared to characters of Western language such as English, Chinese characters have unique structures where an interval can exist inside one character. To train a model that is sensitive to the boundary of one whole character, we need to let the model learning more information about intervals of two characters, rather than the intervals inside a character.

To find out the patent of the intervals of characters, we collected a large number of text images and scaled them. MSFCN extracts the intervals features by scaling down the images with different sizes. By adding these feature maps into the network model, MSFCN takes more intercharacter interval information. Figure 4 shows images that are scaled-down with three different sizes. The image sizes of them from left to right are $256 \times 256 \times 3$, $128 \times 128 \times 3$ and $64 \times 64 \times 3$. The first and second numbers denote the width and height of images, and the last number means the channel numbers of these images. As shown by Figure 4, some intervals inside a character disappear in the down-scaled images. However, intervals between the two characters are still retained. That means, using input images of different scales can enable our model to extract more useful feature information. So we propose a novel scaling method called adaptive multiscale representation, which can be implemented by the following equations:

$$M(x) = \text{MaxPooling}(x, \text{kernel} \_\text{size} = 2) \quad (12)$$
$$A(x) = \text{AvgPooling}(x, \text{kernel} \_\text{size} = 2) \quad (13)$$
$$S(x) = \text{StackPooling}(x, \text{kernel} \_\text{size} = 2) \quad (14)$$
$$C(x) = \text{Conv2D}(x, \text{kernel} \_\text{size} = 2 \times 2, \text{stride} = 2) \quad (15)$$
$$U(x) = \text{ReLU}(M(x) \oplus A(x) \oplus S(x) \oplus C(x)) \quad (16)$$

where $x$ denotes the input feature maps. In this module we add two novel modules to scale origin input image. Stack-pooling is a way to pool the original image without losing any information of pixels and two-stride convolution can pool information from input image adaptively with its variable kernels. With the ASMR module, the down-sampling module is more adaptable to various input images. Figure 5 shows the details of two sampling modules.

![Figure 4](image4.png) **Figure 4** Images scaled down with different sizes: (a) $160 \times 127$(origin), (b) $80 \times 64(1/2 \times)$ and (c) $40 \times 32(1/4 \times)$

![Figure 5](image5.png) **Figure 5** The effect of stack-pooling and two-stride convolution is demonstrated in the figure: (a) in the stack-pooling module, the size of the output will be halved while the number of channels will quadruple and (b) in the two-stride convolution module, the size of the output will be the same as max-pooling and average-pooling but the pooling result includes all factors in the input image

3.4 | Searching connected component

The pixel-wise classification result indicates whether the area of each pixel belongs to text or the background. The adjacent pixels of text characters are grouped to form a heat map of characters. However, such a result cannot directly represent the final character segmentation result. To get the specific spatial information of each character, these components need to be concatenated together. Image regions consisting of foreground pixels with the same pixel value and adjacent positions in an image were called connected components. We can get all characters after searching all connected components candidates in an image. The detailed information of the two-pass algorithm is shown in Table 5.

After all the connected parts are labelled as groups, we can colour each group to observe the experimental results. The result of the connected component searching algorithm is shown in Figure 6. From Figure 6, we noticed that most characters can be segmented well except for the last two. The
The reason is that the pixel-wise classification results from the MSFCN model are not accurate enough. However, these errors can be made up by improving the algorithm of connected domain search. To solve this problem, we proposed a novel method named boundary expansion algorithm.

The boundary expansion algorithm contains three parts: image erosion, connected component search and boundary expansion. To demonstrate the whole procedure of the algorithm, we use a simple example in Figure 7. In this example, we input a pixel-wise segmentation result with eight character regions. Our goal is to find all character region without being affected by the adhesive pixels. First, we use a $5 \times 5$ kernel for image erosion, the small adhesive part between the last two characters is separated. Then we use the connected components search algorithm to define the character region. The regions labelled with different colours in Figure 7 denote the connected components result of different characters. After that, we use the boundary expansion algorithm to expand the region of each connected component to restore them to the former shape. Finally, we can regard the output image as the result of character segmentation tasks.

The reason why we chose the $5 \times 5$ kernel is that in the semantic segmentation result image, the only part we are interested in is the character region. And the connection between the character regions is noise, which will seriously affect the segmentation results. Therefore, we chose the $5 \times 5$ kernel instead of the $3 \times 3$ kernel to remove noises as much as possible.

The main idea of this algorithm is to collect adjacent pixels of origin segment results and expand the boundary of the current region. This algorithm is based on the breadth-first search algorithm so we do not need to worry about a region expanding too quickly. We describe the details of the boundary expansion algorithm in Table 6. In this description, $M$ is an intermediate variable used to store the labelled result in each iteration. And the function adjacent() denotes a function to get four adjacent pixels near the target pixel.

| TABLE 5  Two-pass connected components searching algorithm |
|----------------|----------------------------------|
| **Input:** Eroded semantic segmentation result: | $I = \{ i_{h,w} \mid h < \text{Height}, w < \text{width} \}$ |
| **Output:** Connected components: | $C = \{ c_{h,w} \mid h < \text{height}, w < \text{width} \}$ |
| // First Pass | |
| 1. Label = 1 | |
| 2. for each $i_{h,w} \in I$ do | |
| 3. if $i_{h,w} > 0$ | |
| 4. if $i_{h,w-1} > 0$ and $i_{h,w-1} > 0$ | |
| 5. add $(i_{h,w-1}, i_{h-1,w})$ to Equal_Label | |
| 6. if $c_{h-1,w} > 0$ | |
| 7. $c_{h,w} = i_{h-1,w}$ : continue | |
| 8. if $c_{h,w-1} > 0$ | |
| 9. $c_{h,w} = i_{h,w-1}$ : continue | |
| 10. $c_{h,w} = \text{Label}++$ : continue | |
| 11. end for | |
| // Second Pass | |
| 12. for each $i_{h,w} \in I$ do | |
| 13. if $i_{h,w} > 0$ | |
| 14. let $i_{h,w}$ be the minimal value in Equal_Label | |
| 15. end for | |

**Figure 6** The result produced by connected components searching algorithm. The first row shows samples from the dataset and the result image obtained by using MSFCN model. The second row shows the matrix result.
TABLE 6 Boundary expansion algorithm

| Input: Connected components: |
|-----------------------------|
| \( C = \{ c_{h,w} | h < \text{height}, w < \text{width} \} \) |

Semantic segmentation result
\( S = \{ s_{h,w} | h < \text{height}, w < \text{width} \} \)

Output: Character segmentation result:
\( R = \{ r_{h,w} | h < \text{height}, w < \text{width} \} \)

1. for each \( c_{h,w} \in C \) do
2. if \( c_{h,w} > 0 \)
3. \( \text{Enqueue}(M, c_{h,w}) \)
4. \( r_{h,w} = c_{h,w} \)
5. end for
6. while \( M \neq \emptyset \)
7. \( p_{h,w} = \text{Dequeue}(M) \)
8. for each \( q \in \text{Adjacent}(p_{h,w}) \) do
9. if \( q \in S \)
10. \( q = p_{h,w} \)
11. \( \text{Enqueue}(M, q) \)
12. \( r_{q, \text{label}} = q \)
13. end for
14. end while

![Diagram](image1)

**FIGURE 7** The result produced by connected components searching algorithm. CC refers to connected component searching algorithm and Ex refers to the boundary expansion algorithm.

![Diagram](image2)

**FIGURE 8** Samples of MLCS dataset: (a) training data and (b) labels mapping to data.

of characters that include random numbers of three types of characters. We set a random interval size in each of these characters. Also, we add some random noise in images to increase the practicability of the trained model. The labels are rectangles, and each rectangle can cover a character mapping from training data. Though we show our labels in red colour, in the training process, we convert our labels into matrices which have a size of 256 x 256 x 2. The reasons for choosing this size are described in Section 3.2. The first and second number denotes width and height of the labels and the last number denotes the category of the segment task. There are two classes, the character part and the background part. We create 3000 images, half for training and another half for testing.

4.2 Results and analysis

The loss function we used in our model is cross-entropy. Since each pixel of images has a class label, we can compute the loss between outputs and labels data and get a number between 0 and 1; and then update all parameters based on loss value.
Usually, the optimize function of CNN-like models is stochastic gradient descent (SGD). Recently, Adam [18] has been proved to be a well-optimized function, and the fitting speed of Adam is better than SGD. In our experiment, we utilize Adam and set the learning rate at 0.001.

An HP Z840 workstation is used to train the model. The workstation is equipped with 64 GB memory, two Nvidia Tesla K40c graphics cards and 12 GB video memory. To train the model, iteration is set to 300 rounds, and after 8 h we get the model on MLCS. Figure 9 shows the loss curve during the training procedure.

We use area-under-roc-curve (AUC) as a quantitative standard to compare these experimental results. The full name of AUC is area-under-roc-curve, in which receiver operating characteristic (ROC) is a widely used indicator in evaluating binary tasks. However, ROC cannot be calculated as a quantitative standard, so we calculate AUC to make a judgement. To do the comparison experiments, we choose four segmentation models, which are FCNs, GMM, HOG + SVM, [24, 30, 31] and 2D-DWT + GMM (which is the state-of-the-art in character segmentation tasks), and our model, MSFCN. Figure 10 and Table 7 show the AUC results by different methods.

It can be seen from Figure 10 that the proposed method has better performance than FCNs such as U-net and Segnet. It indicates that keeping the character spacing and character edges of the input image as much as possible will help improving performance of the network model. We can find that MSFCN model achieve best results compared to other methods in the test data.

Besides, to judge the complexity of our model, we used smaller FCNs for a reasonable comparison. We force all FCNs to perform down-sampling only twice. Then we compute the results for the trainable parameters of these models. The obtained results are shown in the table below:

From the results in Table 8, we can see that the trainable parameters of our method and U-net have increased compared with other FCNs, and the parameters in our method are slightly higher than U-net in the decoder. This is because the accuracy of the character segmentation results depends on the performance of the decoder, so a slight increase in the trainable parameters is inevitable.

Figure 11 shows some sample segment results by using MSFCN. To gain insight into what features are extracted by the convolutional filter in our model, we randomly select four activation feature maps after each convolutional layer for observation. As shown in Figure 11, MSFCN not only has a capacity locating multilingual character instances with arbitrary shapes, but also is sensitive to text with different character intervals. This proves that MSFCN is robust for MLCS. Figure 12 shows these feature maps.

5 | EXPERIMENT RESULTS ON ICDAR2013 HANDWRITING DATASET

5.1 | Performance evaluation methodology

In the study [32, 33], the metrics used to evaluate the performance of segmentation tasks rebased on visual criteria. However, simply observing these results is time-consuming and can’t be applied in all cases. To avoid user intervention, we use an automatic performance evaluation technique by comparing

![Image](image_url)

**Figure 9** The loss curve of training procedure

**Figure 10** The comparison between these models in test data

| Method                      | AUC |
|-----------------------------|-----|
| GMM                         | 0.8215|
| HOG + SVM                   | 0.8759|
| FCN [24]                    | 0.9240|
| Segnet [28]                 | 0.9279|
| U-net [29]                  | 0.9354|
| 2D-DWT + GMM [30]           | 0.9380|
| **MSFCN**                   | **0.9417**|
the detected segmentation result with preannotated ground truth (GT) [32]. The metric is based on the count of matched pixels between the areas detected and the ones in the label. To compare the performance of detection results by quantitative criteria, we used a table called MatchScore [33] to calculate the intersection-over-union (IoU) of pixel sets for the result and the GT. The definition of MatchScore can be represented as:

$$\text{MatchScore}(i,j) = \frac{T((G_i \cup R_i) \cap I)}{T((G_i \cup R_i) \cap I)} \quad (17)$$

where \(I\) denotes the matrix of image, \(G_i\) denotes pixels inside the \(j\)-th GT region, \(R_i\) denotes pixels inside the \(i\)-th result region, \(T(S)\) is a function used to counts pixels in region \(S\). Table MatchScore\((i,j)\) represents the matching results of the \(j\)-th GT region and the \(i\)-th result region.

**TABLE 8** Trainable parameters of basic FCNs

| Method     | Encoder | Decoder | Total |
|------------|---------|---------|-------|
| FCN        | 1.1 M   | 0.5 M   | 1.6 M |
| DeconvNet  | 1.2 M   | 1.1 M   | 2.3 M |
| Segnet     | 0.9 M   | 1.0 M   | 1.9 M |
| U-net      | 1.0 M   | 1.9 M   | 2.9 M |
| MSFCN      | 1.1 M   | 2.2 M   | 3.3 M |

The evaluation metric includes the MatchScore table and searches pairs of one-to-one, one-to-many or many-to-one matches. A one-to-one match \((o2o)\) is a pair with a MatchScore higher than threshold \(th\). A g_one-to-many match \((go2m)\) denotes pairs with one GT word matches with more than one detected words. And a g many-to-one match \((gm2o)\) denotes pairs with more than one GT words match with one detected word. In the same way, a d_one-to-many match \((do2m)\) denotes pairs with one detected word matches more than one GT words. Finally, a d many-to-one match \((dm2o)\) denotes pairs with more than one detected words match with one GT words. Let \(N\) be the count of segments from GT, \(M\) be the count of segments from result and \(\omega_1, \omega_2, \omega_3, \omega_4, \omega_5, \omega_6\) be predetermined weights, we calculate the DR and RA are defined as follows:

$$DR = \omega_1 \frac{o2o}{N} + \omega_2 \frac{go2m}{N} + \omega_3 \frac{gm2o}{N} \quad (18)$$

$$RA = \omega_4 \frac{o2o}{M} + \omega_5 \frac{do2m}{M} + \omega_6 \frac{dm2o}{M} \quad (19)$$

Now, F-Measure (FM) can be defined by combining the value of DR and RA. The definition is shown as the following:

$$FM = \frac{2 \times DR \times RA}{DR + RA} \quad (20)$$

**FIGURE 11** The segmentation results obtained by using MSFCN model. The first row shows the samples from the MTST dataset, the second row shows the ground truth (GT) of the corresponding sample in the first row and the third line shows the result image obtained by using MSFCN model.

**FIGURE 12** Feature maps of each layers.
5.2 ICDAR2013 handwriting dataset

To test the practicability of the MSFCN model, we also test MSFCN on a set of handwritten images from the ICDAR2013 handwriting segmentation contest [33]. There are 150 images written in Greek and English language as well as 50 images written in Indian Bangla language included in this dataset.

According to the study, we compare the effect of some of different segmentation algorithms (CUBS, GOLESTAN etc.) on this data set. The methods mentioned in study are the best results in ICDAR2013 handwriting character segmentation contest [33]. There are still some other methods that have the similar performance, so we won’t list them all.

We evaluated the performance of all participating algorithms for word segmentation using Equations (8-11). The

![Sample input images of ICDAR2013 dataset: (a) training data and (b) labels mapping to data](image)

![The segmentation results: (a) samples, (b) ground truth and (c) result image](image)

![The loss curve of training procedure](image)

![The comparison between these models in test data](image)

| Method  | AUC  |
|---------|------|
| GMM     | 0.7258 |
| HOG + SVM | 0.8260 |
| FCN [24] | 0.8735 |
| Segnet [28] | 0.8924 |
| U-net [29] | 0.8990 |
| MSFCN   | 0.9176 |

| Method     | M  | o2o | DR (%) | RA (%) | FM (%) |
|------------|----|-----|--------|--------|--------|
| CUBS       | 23,782 | 20,668 | 87.86 | 86.91 | 87.38 |
| GOLESTAN-a | 23,322 | 21,093 | 89.66 | 90.44 | 90.05 |
| GOLESTAN-b | 23,400 | 21,077 | 89.59 | 90.07 | 89.83 |
| INMC       | 22,957 | 20,745 | 88.18 | 90.36 | 89.26 |
| LRDE       | 23,473 | 20,408 | 86.75 | 86.94 | 86.85 |
| MSHK       | 21,281 | 17,863 | 75.93 | 83.94 | 79.93 |
| NUS        | 22,547 | 20,533 | 87.28 | 91.07 | 89.13 |
| QATAR-a    | 24,966 | 20,746 | 88.19 | 83.10 | 85.57 |
| QATAR-b    | 25,693 | 20,688 | 87.94 | 80.52 | 84.07 |
| NCSR (SoA) | 22,834 | 20,774 | 88.31 | 90.98 | 89.62 |
| ILSP (SoA) | 23,409 | 20,686 | 87.93 | 88.37 | 88.15 |
| TEI (SoA)  | 23,259 | 20,503 | 87.15 | 88.15 | 87.65 |
| MSFCN      | 23,158 | 21,246 | 90.31 | 91.74 | 91.02 |
acceptance threshold used was $\text{th} = 0.95$ for word segmentation. For the convenience of comparison, we set $\omega_1 = \omega_4 = 1, \omega_2 = \omega_5 = \omega_6 = 0$. In addition, to satisfy the input requirements of our model, we preprocessed the dataset images to generate training pictures of three sizes which are shown by Scale ‘n’ ×. Table 1 shows the size of these feature maps.

Then we generate the corresponding label image. Label images have the size of $256 \times 256 \times 3$. In the label image, each word is marked with different colours. For the word segmentation task, we totally create 2000 images, half of which is training data and another half is test data. Figure 13 shows training data (ICDAR2013) and labels mapping to train data. Figure 14 shows some qualitative segmenting results by MSFCN.

6 | RESULTS AND ANALYSIS

We used the same workstation as described in Section 4. The iteration parameters are set to 700 epochs, and the final model is obtained after 40 h parametric training. Figure 15 shows the loss curve for the training process of MSFCN.

First, we use AUC to evaluate the performance as we do in Section 4.2. Table 9 and Figure 16 show the AUC results in ICDAR2013.

The AUC results show that MSFCN also performs well on ICDAR2013 and has a robust multilingual adaptability. To further analyze the performance of the model, we applied evaluation methods shown in Section 5.1.

The evaluation performance obtained from MSFCN and all algorithms submitted to the contest are presented in Table 10. Figures 17 and 18 are the visual representations of Table 10. And from Figure 18 we can find that MSFCN is superior to other character segmentation methods in both accuracy rate and recall rate. And the FM is pushed to 91.02, proving that our method achieves state-of-the-art.
7 | CONCLUSION

MLCS is challenging, especially when characters of Eastern Asian language such as Chinese are involved. We proposed an MSFCN and a novel connected component searching algorithm to solve this problem. Due to the multiscale feature extraction, intercharacter intervals information can be collected by multiple scaling down modules which make our model more robust for multilingual characters. By using the connected components searching algorithm, we got the marked result from eroding the semantic segmentation result made by MSFCN. The boundary expansion algorithm restores the marked result in the normal size and produces the final segmentation result. To enhance the practicability of MSFCN, we made a new dataset MLCS. Experimental results obtained on two datasets prove that MSFCN achieves state-of-the-art segmentation performance for multilingual characters with arbitrary shapes, and is sensitive for text with different character intervals.

In the future, we will focus on how to obtain more abstract features that can describe the intervals of two characters, and improve our model to accelerate the predicting procedure.

ACKNOWLEDGEMENTS

This work is supported by the National Natural Science Foundation of China (61872231, 61701297).

ORCID

Jin Liu

https://orcid.org/0000-0001-7249-698X

REFERENCES

1. Liu, J., et al.: Multi-scale multi-class conditional generative adversarial network for handwritten character generation. J. Supercomput. 75(4), 1922–1940 (2019)
2. Chang, S., Liu, J.: Multi-lane capsule network for classifying images with complex background. IEEE Access. 8, 79876–79886 (2020)
3. Casey, R.G., Lecolinet, E.: A survey of methods and strategies in character segmentation. IEEE Trans. Pattern Anal. Mach. Intell. 18(7), 690–706 (1996)
4. Ryu, J., Koo, H.I., Cho, N.I.: Word segmentation method for handwritten documents based on structured learning. IEEE Signal Process. Lett. 22(8), 1161–1165 (2015)
5. Inkeaw, P., et al.: Recognition-based character segmentation for multi-level writing style. Intl. J. Doc. Anal. Recognit. 21(1-2), 21–39 (2018)
6. Liu, J., Zhang, X., Li, Y.: Deep learning-based reasoning with multi-ontology for IoT applications. IEEE Access. 7, 124688–124701 (2019)
7. Shang, S., Liu, J., Yang, Y.: Multi-layer transformer aggregation encoder for answer generation. IEEE Access. 8, 90410–90419 (2020)
8. Xu, L., Fan, W., Sun, J.: An HMM-based over-segmentation method for touching Chinese handwriting recognition. In: 2016 15th International conference on frontiers in handwriting recognition, ShenZhen, pp. 343–348. (ICFHR) (2016)
9. Bae, Y., et al.: Character region awareness for text detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, Los Angeles, pp. 9365–9374 (2019)
10. Zhang, C., et al.: Look more than once: an accurate detector for text of arbitrary shapes. In: Proceedings of the IEEE conference on computer vision and Pattern recognition, Los Angeles, pp. 10552–10561 (2019)
11. Zhao, J., et al.: The edge detection of river model based on self-adaptive Canny algorithm and connected domain segmentation. In: 2010 8th World congress on intelligent control and automation, Jilin, pp. 1333–1336 (2010)
12. Kavitha, A.S., et al.: A new watershed model based system for character segmentation in degraded text lines. Int. J. Electron. Commun. 71, 45–52 (2017)
13. Seni, G., Cohen, E.: External word segmentation of off-line handwritten text lines. Pattern Recognit. 27(4), 41–52 (1994)
14. Louloudis, G., et al.: Text line and word segmentation of handwritten documents. Pattern Recognit. 42(12), 3169–3183 (2009)
15. Srihari, S., et al.: Handwritten Arabic word spotting using the cedaraar document analysis system. In: Proceedings of the symposium on document image understanding technology, College Park, pp. 123–132. (SDIUT-05) (2005)
16. Wang, T., et al.: End-to-end text recognition with convolutional neural networks. In: Proceedings of the 21st international conference on pattern recognition, Tsukuba, pp. 3304–3308. (ICPR2012) (2012)
17. Chen, L.C., et al.: Deeply semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFS. IEEE Trans. Pattern Anal. Mach. Intell. 40(4), 834–848 (2017)
18. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint (2014) arXiv:1412.6980
19. Dai, J., et al.: Object detection via region-based fully convolutional networks. Adv. Neural Inform. Process. Syst. 379–387 (2016)
20. Wu, Z., Shen, C., Hengel, A.V.D.: High-performance semantic segmentation using very deep fully convolutional networks. arXiv preprint (2016) arXiv:1604.04339
21. Yang, J., et al.: Object contour detection with a fully convolutional encoder-decoder network. Proceedings of the IEEE conference on computer vision and pattern recognition, Las Vegas, pp. 193–202 (2016)
22. Zhang, Z., et al.: Multi-oriented text detection with fully convolutional networks. Proceedings of the IEEE conference on computer vision and pattern recognition, Las Vegas, pp. 4159–4167 (2016)
23. Noh, H., Hong, S., Han, B.: Learning deconvolution network for semantic segmentation. Proceedings of the IEEE international conference on computer vision, Las Condes, pp. 1520–1528 (2015)
24. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. Proceedings of the IEEE conference on computer vision and pattern recognition, Boston, pp. 3431–3440 (2015)
25. Yang, Y.D., Wang, X.F., Zhang, H.Z.: Local importance representation convolutional neural network for fine-grained image classification. Symmetry. 10(10), 479 (2018)
26. Li, X., et al.: Shape robust text detection with progressive scale expansion network. arXiv preprint (2019) arXiv:1806.02559.
27. Lin, T.Y., et al.: Feature pyramid networks for object detection. Proceedings of the IEEE conference on computer vision and pattern recognition, Honolulu, pp. 2117–2125 (2017)
28. Badrinarayanan, V., et al.: A deep convolutional encoder-decoder architecture for image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. 39(12), 2481–2495 (2017)
29. Ronneberger, O., et al.: Convolutional networks for biomedical image segmentation. In: International conference on medical image computing and computer-assisted intervention, Berlin, pp. 234–241 (2015)
30. Basavaraju, H.T., et al.: Arbitrary oriented multilingual text detection and segmentation using level set and Gaussian mixture model, pp. 1–14. Evolutionary Intelligence (2020)
31. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: International conference on computer vision pattern recognition, San Diego, pp. 886–893. (CVPR’05) (2005)
32. Gatos, B., et al.: ICDAR2007 handwriting segmentation contest. In: The 9th international conference on document analysis and recognition (ICDAR). 14(1), 25–33 (2007)
33. Statonopoulos, N., et al.: ICDAR 2013 handwriting segmentation contest. In: 2013 12th International conference on document analysis and recognition, pp. 1402–1406 (2013)