Hindawi
Scientific Programming
Volume 2021, Article ID 5520338, 9 pages
https://doi.org/10.1155/2021/5520338

Research Article

OCR with the Deep CNN Model for Ligature Script-Based Languages like Manchu

Diandian Zhang,[1] Yan Liu,[1] Zhuowei Wang,[2] and Depei Wang[3]

[1]*School of Literature, South China Normal University, Guangzhou 510631, China*
[2]*School of Computers, Guangdong University of Technology, Guangzhou 510006, China*
[3]*School of Automation, Guangdong University of Technology, Guangzhou 510006, China*

Correspondence should be addressed to Depei Wang; depeiwang@qq.com

Received 28 February 2021; Accepted 25 May 2021; Published 1 June 2021

Academic Editor: Antonio J. Peña

Copyright © 2021 Diandian Zhang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Manchu is a low-resource language that is rarely involved in text recognition technology. Because of the combination of typefaces, ordinary text recognition practice requires segmentation before recognition, which affects the recognition accuracy. In this paper, we propose a Manchu text recognition system divided into two parts: text recognition and text retrieval. First, a deep CNN model is used for text recognition, using a sliding window instead of manual segmentation. Second, text retrieval finds similarities within the image and locates the position of the recognized text in the database; this process is described in detail. We conducted comparative experiments on the FAST-NU dataset using different quantities of sample data, as well as comparisons with the latest model. The experiments revealed that the optimal results of the proposed deep CNN model reached 98.84%.

1. Introduction

Optical character recognition (OCR) is the key technology for the digitization of many modern characters, and it is also the mainstream technology in text recognition. Indeed, Tausheck obtained a patent on OCR technology 89 years ago. Around the 1950s and 1960s, research on OCR technology began to increase in countries around the world. The system for identifying postal codes in Japan that was developed then is still in use today. Since the 1970s, Chinese character recognition has undergone decades of research and development. In 1984, Japanese researchers designed a device capable of recognizing Chinese characters in multiple typefaces. The recognition rate was as high as 99.98%, with a recognition speed greater than 100 characters per second. At present, the methods and technologies for Chinese character recognition research have matured, and this has been applied to product design.

Used by ethnic minorities such as the Manchu and Xibe in China, Manchu is a spoken and written language with a phonetic script. Due to the specificity of letter concatenation and deformation in the Manchu alphabet system, the writing rules are completely different from modern Chinese and more closely resemble Mongolian and ancient Chinese. Manchu is read and written from top to bottom and from left to right, and it can also be written in Pinyin characters that are divided into monophthongs, diphthongs, consonants, and digraphs, with the lengths of radicals corresponding to these letters. This can, however, differ in different Manchu scripts. During the translation and interpretation of Manchu books, some radicals are difficult to identify due to faults such as consecutive writing, deformation, misidentifications, scratches, and cracks, which make the corresponding text, in turn, difficult to identify quickly and accurately. Some of these faults are caused by preservation measures in the scanned book image, while the error rate of false detection in Manchu images through existing Manchu recognition methods has greatly increased for other reasons as well. Recognition errors often occur when the Manchu source text is handwritten, so using OCR on handwritten Manchu remains inconvenient.

Manchu recognition methods often require segmentation of Manchu into basic units (e.g., letters) first, followed by recognition [1]. Improvements in Manchu recognition are often simply improvements in segmentation accuracy, which do not solve the fundamental problem of low recognition accuracy caused by letter concatenation and deformation. Manchu words are composed of one or more letters connected along the vertical axis. There is no gap between letters in the same word [2]. The initial point of the letters is located on the central axis in the Manchu word image, so it is difficult to recognize quickly and accurately the Manchu characters formed based on the positioning image or handwriting through traditional segmentation methods [3].

This article presents a methodology and system for Manchu text recognition. The recognition method can quickly recognize parts of letters in the text image without word segmentation. Based on this recognition component, we extended the text retrieval system to the full database to find similar texts. The system identified all of the similar letters in turn in the images of Manchu text through the sliding window. The partial recognition of letters and the partial adjustment image in the sliding window are compared according to the standard images of the letters. Following letter recognition, it is possible to find the associated Manchu characters in the database quickly. The system can output the corresponding letters with their database counts, as well as locate the standard image of all letters marked as associated letters in the database. The sliding window was proposed to reduce the computational complexity and improve the recognition accuracy; it can index part of the letter area with higher accuracy, and this local recognition ensures the reliability of letter identification and reduces the probability of false detection.

2. Background and Existing Work

To create a fully functional OCR system, it is necessary to understand the scripting background of the language and its methodology. The following sections describe the deep learning model in detail, with various convolution network (CNN) models used in image classification. The glyph structure and preprocessing method for Manchu are also analyzed in detail, according to the latest methods of recognizing Manchurian scripts.

2.1. CNN Models. As a subfield of machine learning, deep learning has achieved good results in image classification. A variety of CNN models have sprung up that show promising ability in shallow and deep feature mining, which continuously improves the classification accuracy. These models have a wide range of applications, such as face [4], speech [5], and scene text recognition [6]. A general CNN is usually composed of an input layer, a convolutional layer, a pooling layer, and a fully connected network. The depth of CNN-related models has gradually deepened over the past decade, and the models have gradually become larger. AlexNet [7] has 60 million parameters and 650,000 neurons; it consists of five convolution layers, with the largest pool layer, and three fully connected layers with 1000-way softmax. VGG [8], an architecture with a very small (3×3) convolution filter, is used to evaluate the network with increased depth. Significant improvements to the prior art configuration can be achieved by pushing the depth to the 16–19 weight layer, while some studies have focused on reducing model parameters, maintaining high classification accuracy, and improving equipment ease of use. SqueezeNet [9] proposed reducing the size of the model while achieving the same results; it employs three main strategies: a reduced filter, decreasing the input channel filters, and downsampling. ResNet [10] uses a residual learning framework in which the layer is changed to learn the residual function of the layer input instead of learning the unreferenced function. MobileNet [11] uses the depth separable convolution based on a streamlined structure and introduces two simple hyperparameters to compromise between speed and accuracy; it is suitable for mobile terminals and embedded devices. DenseNet [12] introduces the dense convolution network to shorten the connection between the input and output layers, and to strengthen the feature connection between layers.

2.2. Manchu Language OCR. To explain the text processing procedure during text recognition, we start with a brief overview of the structure and alphabet of Manchu. According to the National Standard of the People's Republic of China, Information Technology Universal Multi-Eight Coded Character Set Sibe, Manchu Character type, the same letter in Manchu generally has four different forms for the independent, initial, medial, and final shapes. A Manchu word is composed of one or more letters.

Research on Manchu character recognition is still in its infancy, and most studies are based on character segmentation. According to the structural characteristics of the text, Manchu words are divided into individual characters by projection methods or the stroke-by-stroke growth method. Then, backpropagation neural network, statistical pattern recognition, and support vector machine (SVM) methods [13] are used to recognize the characters or strokes. Finally, the individual characters or strokes are combined into characters according to specific rules. The advantage of the method is that the size of the dataset is smaller than that of the word-level dataset, which further compresses the training data to reduce the number of calculations [14]. A classifier with a relatively simple structure can also be used for recognition to improve efficiency [15]. However, due to the complexity of the Manchu word structure, the correct segmentation of Manchu letters cannot be fully realized, which restricts the accuracy of subsequent character recognition.

The character recombination technology, after recognition, also needs to be resolved. For example, Yi et al. built an offline handwritten Manchu text recognition system in 2006 [16]. They first extracted and preprocessed the recognition target, and then they segmented the extracted Manchu text to form Manchu stroke primitives. Next, they performed statistical pattern recognition on the stroke primitives to obtain the stroke sequence. The stroke sequence was then converted into a radical sequence, and the fuzzy string matching algorithm was used to achieve Manchu-Roman transliteration as an output, with a recognition rate of 85.2%. Finally, implicit Markov model method was used to process the recognition results for the stroke primitives and further improved the recognition rate to 92.3% [17]. In 2017, Akram and Hussain et al. proposed a new Manchu stroke extraction method [18]. After preprocessing the recognition target, the main text is determined, and the text growth method is used to extract the Manchu strokes automatically; the stroke extraction accuracy was 92.38%, and the stroke recognition rate was 92.22%. In 2019, Arafat and Iqbal et al. worked on the recognition of handwritten Manchu text [19]. First, the handwritten like Manchu text is scanned, and the image is preprocessed. The full-text elements are then segmented. Next, the projection features, endpoints, and intersections are extracted from the Manchu text elements. After the feature and the chain code features, the three types of features are classified and recognized, and the combinations of the three types of features are recognized simultaneously. Finally, to further improve the recognition rate, the hidden Markov algorithm is used to process the recognition results, which yielded the highest recognition rate (89.83%) [20].

These articles on Manchu character recognition are all based on character segmentation and used manual design to extract shallow features. However, the segmentation-based OCR method is based on accurate character segmentation technology [21]. For historical document images, due to the complexity of various typefaces and styles, inconsistent lighting during image capture, noise caused by capture equipment, and variable background colors (among other idiosyncrasies), correct segmentation of characters becomes more challenging. However, a hand-designed feature extractor requires significant engineering skills and domain expertise. Recognition methods based on character segmentation restrict the recognition accuracy for Manchu words because of errors in processing character segmentation. This paper therefore proposes a method for the recognition of Manchu words without segmentation using a CNN to recognize and classify unsegmented Manchu words. The proposed method also includes improvements to the traditional CNN so that it can train on any size of unsegmented Manchu word images, thereby reducing the influence of normalization preprocessing on the recognition rate.

3. Methodology

In our present work, we use the deep CNN model to recognize the text and then build a Manchu recognition system. The deep CNN model uses four convolution layers to mine different image features. In the Manchu recognition system, the sliding window method is used to identify the same characters in the database. These two approaches are discussed below.

3.1. Manchu Recognition Algorithm. This paper builds a CNN to identify and classify Manchu words without segmentation [22, 23]. The CNN architecture proposed in this article is shown in Figure 1.

The CNN model constructed in this paper includes nine layers in total: four convolutional layers, two maximum pooling layers, and the classification layers, which consist of a flattening layer, a fully connected layer, and the output layer [24]. The first layer is a convolutional layer; the number of convolution kernels is set to 32, and the size of each convolution kernel is set to $n \times n$ ($n = 5$ or 3 or 2: the appropriate convolution kernel is selected through experimentation). This layer convolves the data of the input layer to obtain 32 feature maps. Each feature map consists of $(28-n+1) \times (28-n+1)$ neurons, and each neuron has an $n \times n$ ($n = 5$ or 3 or 2) acceptance domain. The activation function used is ReLU.

The second layer is another convolutional layer, with the same settings as the first layer. The activation function used is ReLU. The third layer is the pooling layer, which plays the role of subsampling and local averaging. The receptive field size of each neuron is set to 2×2, with one trainable bias, one trainable coefficient, and a sigmoid activation function. The fourth and fifth layers are additional convolutional layers, with the same settings as the first layer. The activation function used is ReLU. The sixth layer is another pooling layer, with the same settings as the third layer.

The seventh layer is the flattening layer. Because the input image may be two-dimensional, the flattening layer is used to reduce the dimensionality of the multidimensional matrix output by the previous layer to obtain a one-dimensional feature vector. The eighth layer is a fully connected layer, which contains 256 neurons, and each neuron in this layer is fully connected to the previous layer. The ninth and final layer is the output layer, which outputs a one-dimensional vector of length 671 with the softmax activation function. Because the size of the word image in the multilevel Manchu dataset is arbitrary in length, the size of the original image must be scaled to a fixed size of 28×28.

3.2. Manchu Recognition System. The proposed Manchu recognition system is divided into ten units, and the process is illustrated in Figure 2.

(1) The letter image-reading unit reads the standard image of each letter

(2) The Manchu image acquisition unit collects the Manchu images and creates binarized images

(3) The image-preprocessing unit filters the binarized image, extracts the salient area after filtering, and performs edge detection on the salient area to obtain the image for recognition

(4) The parameter initialization sets the initial values of variables i and j

(5) The sliding image extraction searches for the image to be recognized using the sliding window method

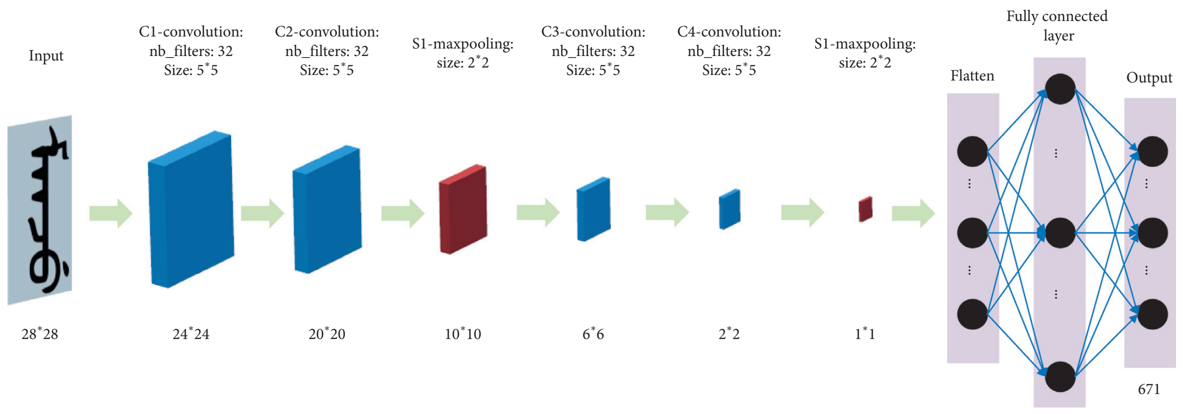
Input
C1-convolution: nb_filters: 32 Size: 5*5
C2-convolution: nb_filters: 32 Size: 5*5
S1-maxpooling: size: 2*2
C3-convolution: nb_filters: 32 Size: 5*5
C4-convolution: nb_filters: 32 Size: 5*5
S1-maxpooling: size: 2*2
Fully connected layer
Flatten
Output
28*28
24*24
20*20
10*10
6*6
2*2
1*1
671

FIGURE 1: CNN framework for Manchu OCR.

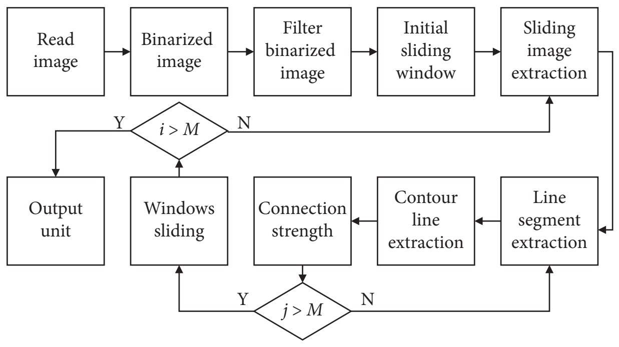

FIGURE 2: Manchu recognition system process framework.

(6) The standard line segment extraction unit filters the binarized image of the *j*-th letter standard image
(7) The contour line extraction unit scales the *i*-th contrast image
(8) The connection strength output unit calculates the connection strength between the vector contour line of the *i*-th contrast image and the *j*-th letter standard image and then jumps according to the value of *i*
(9) The window sliding unit increases the value of *i*, *j*, and *W* and then jumps according to the value of *j*
(10) The result output unit renders the characters and numbers of the letters corresponding to all of the letter standard images marked as associated letters in the database

The first step in our proposed Manchu recognition system is to read the standard image of each letter. Currently, the standard letter image is an image of the Manchu alphabet prestored in the database, which contains 114 images of Chinese alphabets in total; the database also stores the characters and their numbers of the alphabets corresponding to the alphabetic standard images, as well as the component numbers of the constituent letters of each Manchu image. The corresponding output can be based on multiple one or more letters in Manchu characters, and the standard image of each letter contains different glyphs.

The second step is to collect the Manchu images and obtain binarized images. The Manchu image is acquired by using a line-scan camera, handwriting input, or scanning Manchu books with a scanner.

The third step is to filter the binarized image and, after filtering, to extract the salient area and perform edge detection to obtain the image to be recognized. The filtering method used could be mean, median, or Gaussian filtering; the method for extracting the salient area could use the AC [25], histogram-based contrast [26], LC [27], or frequency-tuned saliency detection [28] algorithms. The edge detection of the salient area determines the pixel-area boundaries of the Manchu text in the image.

The fourth step is to set the initial value of variable i to 1, where the value range of i is $[1, N]$, i is a natural number, and N is the ratio of the size of the image to be recognized to the size of the sliding window. Then, it is necessary to set the initial value of variable j to 1, where the value range of j is $[1, M]$, M is the total number of letter standard images, and the width of the sliding window is W.

The fifth step is to search for the image to be recognized using the sliding window method. The longest line segment is searched in the sliding window through the Hough transform as the i-th line segment, and the i-th line segment and the vertical axis (y-axis) of the image matrix are calculated. The angle in the clockwise direction is the i-th angle. When the length of the i-th line segment is greater than or equal to K, the image in the sliding window is rotated counterclockwise by the i-th angle to obtain the image in the sliding window as the i-th contrast image (when i is less than K, it means that there are no Manchu letters to be recognized in the area of the sliding window), where $K = 0.2 * W$.

The sliding window method searches the image to be recognized through a window that slides, and the step amount for each slide is the size of the sliding window; the size of the sliding window is between [0.01, 1] times the size of the image to be recognized; that is, the height and width (size) of the sliding window are set as $W * W$, and the value of W is [0.01, 1] times the width of the image to be recognized. This can be adjusted according to the full number of characters in the image to be recognized, and the distance of each slide is the width of the sliding window. Each time a row is swiped on the image matrix, it automatically jumps to the next row according to the height of the sliding window (scanning the image horizontally from the pixel area in the image matrix that has not been selected by the sliding window). Note that the height and width are also called the length and width, in pixels. Figure 3 illustrates a schematic diagram of the scanning slide described in this step.

The sixth step is to filter the binarized image of the j-th letter standard image. After filtering, the salient area is extracted, and edge detection is performed on the salient area to obtain the j-th letter standard image to be recognized and to search for the j-th letter by the Hough transform. The longest line segment in the image requiring recognition is taken as the standard line segment.

The seventh step is to scale the i-th contrast image according to the ratio between the standard line segment and the i-th line segment and to extract the vector contour lines of the i-th contrast image and the j-th letter standard image. When the i-th contrast image is scaled according to the ratio between the standard line segment and the i-th line segment, the edge of the i-th contrast image leaves at least 8 pixels blank to obtain an 80×80 size image. The vector contour line of the i-th comparison image can be extracted in the following way: in the i-th comparison image, start from the end point of the i-th line segment with the closest distance to the ordinate axis, and calculate the curvature value of the i-th line segment at each edge point; then, calculate the average value of all of the curvature values—all of the curvature edge points with values greater than the average value are used as corner points to form a large curvature point set; and then, connect, in turn, each corner point in the large curvature point set with a curvature value greater than the average value. See Appendix A for more details.

The eighth step is to calculate the connection strength between the vector contour line of the i-th comparison image and the j-th letter standard image. To do so, first increase the value of j by 1; when the value of j is greater than M, go to Step 9, when the connection strength is greater than the strength threshold, mark the j-th letter standard image as an associated letter and go to Step 9, and when the j value is less than or equal to M, go to Step 6. The intensity threshold value is 0.5–0.8 times PNum or 0.5–0.8 times the number of corner points on the contour line in the vector contour line of the j-th letter standard image. See Appendix B for more details.

The ninth step is to increase the value of i by 1, set the value of j to 1, and slide the sliding window at the distance of W. When the value of i is less than or equal to N, go to Step 5. When the value of i is greater than N, go to Step 10.

The tenth step is to output all of the alphabet standard images marked as associated letters in the database corresponding to the characters and numbers of the letters. The output characters and their numbers are used to output Manchu characters, including more than one character corresponding to the alphabetic standard image marked as part of an associated alphabet in the database.

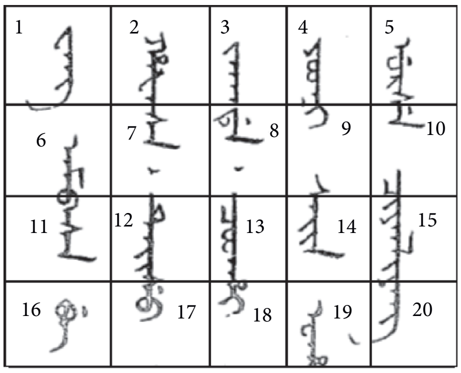

Figure 3: Schematic diagram of the sliding window.

4. Experiments

We tested the proposed system experimentally. First, the experiment verified the recognition effect of the CNN and deep CNN on different types of unsegmented Manchu word data. The unsegmented Manchu word dataset contains 671 categories; each category has 1000 samples, so the total size is 671,000. In the training process, 1,000 sample images were shuffled, then 900 images were randomly selected for training, and the remainder was used for testing. The CNN model and deep CNN model were used for 100, 200, 300, 400, 500, 600, and 671 categories. When using the CNN to recognize Manchu words, the original image was normalized to a uniform size of 28×28. We tested three networks with convolution kernels of 5×5, 3×3, and 2×2, and the convolution kernel of 3×3 yielded the best results. The convolution kernels of the two networks were therefore set to 3×3, the sliding window size of the maximum pooling layer was set to 2×2, the number of filters was set to 32, the dropout ratio was set to 0.25, and there were 4 convolution layers. The experimental results are shown in Table 1.

Under the same parameters, in the recognition and classification of different categories of unsegmented Manchu words, the recognition rate of the deep CNN is 0.76%, 0.61%, 1.00%, 0.89%, 0.82%, 0.73%, and 1.23% higher for each category count than that of the traditional CNN. This indicates that the CNN was improved by the spatial pyramid pooling layer, which has a certain inhibitory effect on the influence caused by image normalization. At the same time, it can be seen that, for 671 types of data, the CNN can obtain a high recognition rate.

We collected papers on Manchu text recognition and summarized the classification accuracy in Table 2. Li et al. [29] used a spatial pyramid pooling layer in the CNN to replace the last maximum pooling layer, and a classifier of any size is proposed to recognize Manchu words without word segmentation. Zheng et al. [30] put forward the idea of nonsegmentation to identify and understand Manchu and replace Manchu roles with words; an end-to-end nine-layer CNN was also proposed to extract the deep features of Manchu text images automatically. Xu et al. [31] improved the traditional projection segmentation method, effectively improving the accuracy of segmentation. The method

Table 1: Experimental results of the CNN and deep CNN on different types of datasets.

Number of categories	Data size	Deep CNN recognition rate (%)	CNN recognition rate (%)
100	100,000	99.85	99.09
200	200,000	99.88	99.27
300	300,000	99.24	98.24
400	400,000	99.35	98.44
500	500,000	99.13	98.31
600	600,000	98.98	98.25
671	671,000	98.84	97.61

Table 2: Comparison with the existing technique for Manchu OCR.

Reference	Accuracy (%)
Li et al. [29]	97.68
Zheng et al. [30]	95
Xu et al. [31]	87.4
Present paper	98.84

Table 3: Experimental results of different numbers of convolutional layers.

Convolutional layers	Deep CNN recognition rate (%)	CNN recognition rate (%)
2	98.04	97.54
4	98.84	97.61
6	98.91	97.66

proposed in this paper is more accurate than these other methods by 1.16%, 3.84%, and 11.44%.

Next, we considered the influence of different numbers of convolutional layers on recognition and classification. The number of convolutional layers was set to 2, 4, and 6 for both CNN and deep CNN. The convolution kernel of the two networks was set to 3×3, the sliding window size of the maximum pooling layer was set to 2×2, the number of filters was set to 32, and the dropout ratio was set to 0.25. The experimental results are shown in Table 3.

As can be seen from Table 3, as the number of convolutional layers increases, the accuracy increases. At the same time, when different numbers of convolutional layers are set, the recognition rate of the deep CNN is higher than that of the traditional CNN by 0.50%, 1.23%, and 1.25% for 2, 4, and 6 layers, respectively. This further shows that the CNN is improved by using the spatial pyramid pooling layer.

The experimental results show that, under the same parameters, the recognition rate of the deep CNN model designed in this paper for the recognition and classification of different categories of unsegmented Manchu words is higher than that of the traditional CNN. Compared with the methods proposed in other papers, the method we proposed here has great advantages. When setting different numbers of convolutional layers, the recognition rate of the deep CNN is higher than that of traditional CNNs, and the deep CNN model proposed in this paper avoids the feature expression problem caused by image normalization. It was tested in a recognition experiment with Manchu words of different lengths and obtained higher recognition accuracy than the traditional CNN model.

5. Conclusion

Traditional CNNs require input images of a consistent size. Manchu is a phonetic text, and its word length is not fixed. Preprocessing to ensure the uniform size is therefore required before recognition and classification, and such preprocessing reduces the recognition rate. To reduce the effect of image normalization preprocessing on the recognition rate, this paper improves the traditional CNN and constructs a new nonsegmented Manchu word recognition network model: deep CNN. We also proposed a Manchu identification system, which locates the position of the recognized text in a database. The model solves the image normalization problem, realizes the extraction of deep features from the unsegmented Manchu word images of any size, and recognizes and classifies different types of unsegmented Manchu word data. Using the deep CNN to recognize and classify unsegmented Manchu words in categories of 100, 200, 300, 400, 500, 600, and 671, the recognition rates were 99.85%, 99.88%, 99.24%, 99.35%, 99.13%, 98.98%, and 98.84%, respectively. The experimental results indicated that the deep CNN reduced the effects caused by image normalization preprocessing and obtained higher recognition accuracy than the traditional CNN model.

Appendix

A. Extraction Method

This section briefly describes the extraction method for the vector outline of the j-th letter standard image in the seventh step. Starting from the end of the standard line segment with the closest distance to the ordinate axis, the curvature value of each edge point on the standard line segment is calculated. The average value of all the curvature values is then calculated, and all of the edge points whose curvature value is greater than the average value are identified as the edge. The corner points constitute a large curvature point set. Each corner point in the large curvature point set whose curvature value is greater than the average value is connected in turn, according to the steps outlined in the following:

(1) Let the coordinates of the corner point with the smallest value of the abscissa (x-axis) of the large curvature point set be (Xmin, Ymin), and set the

coordinates of the corner point with the largest value of the x-axis in the large curvature point as (Xmax, Ymax). Set the spacing formed by multiple columns of pixels on the y-axis to span pixels, where the span is an integer between 10 and 100. Set the initial value of variable h to 0, and set the initial value of variable r to 1, where both h and r are natural numbers.

(2) Sort the corner points of the large curvature point set from small to large according to the value of the x-axis, except for the corner points with the largest and smallest x-axis values. There is a connection mark Linkmark and an array mark ArrayMark; both Linkmark and ArrayMark are set to 1, and the connection mark Linkmark of the corner point with the smallest x-axis value is set to 2 (that is, the connection mark of the first corner point in the set of the large curvature points Linkmark = 2); and the connection mark Linkmark of the corner point with the largest x-axis value is set to 3 (that is, the connection mark Linkmark = 3 of the last corner point in the large curvature point set).

(3) Let the value range of the x-axis of the image matrix be $X\text{min}+(h-1) * \text{span}$ to $X\text{min} + h * \text{span}$ as the connected interval of the r-th layer, and let the value range of the x-axis be Xmin. The interval from $X\text{min} + h * \text{span}$ to $X\text{min}+(h+1) * \text{span}$ is the interval to be connected in the $(r+1)$-th layer.

(4) If there is a corner point of Linkmark = 2 in the connected interval of the r-th layer (that is, the first corner point in the large curvature point set, the leftmost boundary point in the image), then use the vector line to change Linkmark = 2. Connect the corner points of 2 and all the corner points of Linkmark = 1 in the $(r+1)$-th layer to be connected, and connect all of the corner points of the r-th connected interval and the $(r+1)$-th layer to be connected. Set the connection mark Linkmark to 0, and operate according to Step 5 (the above steps are only performed at the first corner point); otherwise, if there is no corner point with Linkmark = 2 in the r-th layer connected section, then proceed directly to the next operation.

(5) If there is a corner point of Linkmark = 3 in the interval to be connected in the $r+1$ layer (that is, the last corner point in the large curvature point set, the rightmost boundary point in the image), then the vector line in the connected interval of the r-th layer and all of the corner points of the connection mark Linkmark = 0 are connected to the corner points of Linkmark = 3; set the connection mark Linkmark of the corner points of Linkmark = 3 to 0 (the above steps are only in the last corner point executed), and go to Step 10 (that is, the connection process ends). If there is no corner point with Linkmark = 3 in the interval to be connected in the $r+1$ layer, then judge whether the connection mark Linkmark of all of the corner points in the large curvature point set is equal to 0 (that is, the connection process ends), and if so, go to Step 10; if not, go to Step 6.

(6) Input the corner points of Linkmark = 0 in the connected interval of the r-th layer into the array as a connected array; input the corner points of Linkmark = 1 in the to-be-connected interval of the r+1 layer into another array as the array to be connected; the corner points in the connected array and the to-be-connected array are sorted according to the ordinate value from small to large.

(7) Take the corner point with the largest ordinate value among the corner points of the array mark ArrayMark = 1 in the connected array as the first starting point. Connect the first start point and the first end point with a vector line, and set the array mark ArrayMark of the first start point and the first end point to 0 (this step is to set the corners with the highest y-axis value sequentially, see above).

(8) Set the corner point with the smallest ordinate value among the corner points of the array mark ArrayMark = 1 in the connected array as the second starting point, and mark the corner points of the array to be connected with ArrayMark = 1. The corner point with the smallest ordinate value is used as the second end point. The second start point and the second end point are connected by a vector line, and the array marks of the second start point and the second end point are set to 0 (this step connects the corner points with the lowest y-axis values in turn).

(9) When the array mark ArrayMark of all of the corner points in the connected array or the to-be-connected array is equal to 0 (that is, all of corner points in any range within the distance between the two coordinate axes are connected), increase the variables h and r by 1, set the array mark ArrayMark of all corner points in the array to be connected to 1, and set the Linkmark of all the corner points in the array to be connected and the connected array to 0, and go to Step 3 (i.e., connect the next set of coordinate axis spacing ranges); otherwise, go to Step 6 (i.e., continue to connect the corner points in the connected array and the array to be connected).

(10) Output the vector outline formed by all of the corner points connected by the vector line (connect the corner points to form the frame of the letter).

B. Calculation of Connection Strength

This section briefly describes the method for calculating the connection strength between the vector contour lines of the

i-th contrast image and the j-th letter standard image in the eighth step.

(1) Let the vector contour line of the i-th contrast image be P and the vector contour line of the j-th letter standard image be Q; superimpose P and Q with the center of gravity of P and Q as the center. Of these, $P = \{p_1, p_2, \ldots, p_k | k > 0\}$, k is the number of corner points on the contour line in the vector contour line of the i-th contrast image; $Q = \{q_1, q_2, \cdots, q_n | n > 0\}$, n is the number of corner points on the contour line in the vector contour line of the j-th letter standard image; p_k and q_n are the corner points on the contour line; and p_1 and q_2 are the distances on P and Q. The corner points with the smallest distance on the ordinate axis, p_1, $p_2, \ldots, p_k$ and q_1, $q_2, \ldots, q_n$, are the corner points in the two sets of sequences with the increasing distance of the index number from the ordinate axis in sequential increments.

(2) Connect the nearest corner points on P and Q with vector lines in sequence, and the set of connected edges is the set of connected edges $V = \{(p_{FC}, q_{FC})\}$, where FC is the number of corner points, the value range is [1, PNum], PNum is a constant, and the value of PNum is smaller of the two values k and n.

(3) Calculate the distances from the endpoints p_{FC} to p_1 of all connected edges in the connected edge set (calculate the distances from all corner points of $p_1, p_2, \ldots, p_{\mathrm{PNum}}$ to p_1), and use the calculated distances as the first distance set in turn. Calculate the distances from the endpoints q_{FC} to q_1 of all the connected edges in the connected edge set (calculate the distances from all corner points of q_1, $q_2, \ldots, q_{\mathrm{PNum}}$ to p_1), and use the calculated distances as the second distance set in turn. Calculate the difference between each distance element in the first distance set and each distance element in the second distance set in sequence; count all of the difference values as the number of positive and negative numbers; when the number of positive numbers is greater than the number of negative numbers, go to Step 4; otherwise, go to Step 5; the distance element is each distance value in the set.

(4) Calculate the connection strength S of the connected edge set V, and go to Step 6, where $S = \sum_{FC=1}^{\mathrm{PNum}} \mathrm{Coeffi}\left(|p_{\mathrm{FC}} - q_{\mathrm{FC}}|_x\right)$. The similarity function is

$$\mathrm{Coeffi}\left(\left|p_{\mathrm{FC}} - q_{\mathrm{FC}}\right|_x\right) = \begin{cases} 1, & \left|p_{\mathrm{FC}} - q_{\mathrm{FC}}\right|_x > 0, \\ 0, & \left|p_{\mathrm{FC}} - q_{\mathrm{FC}}\right|_x = 0, \\ -1, & \left|p_{\mathrm{FC}} - q_{\mathrm{FC}}\right|_x < 0, \end{cases} \tag{B.1}$$

where $|p_{\mathrm{FC}} - q_{\mathrm{FC}}|_x$ means the difference between the abscissa value of point p_{FC} (distance value from point p_{FC} to the x-axis) and the abscissa value of q_{FC} (distance value from point q_{FC} to the x-axis).

(5) Calculate the connection strength S of the connected edge set V, and go to Step 6, where $S = \sum_{\mathrm{FC}=1}^{\mathrm{PNum}} \mathrm{Coeff}i\left(|q_{\mathrm{FC}} - p_{\mathrm{FC}}|_y\right)$. The similarity function is

$$\mathrm{Coeffi}\left(\left|q_{\mathrm{FC}} - p_{\mathrm{FC}}\right|_y\right) = \begin{cases} 1, & \left|q_{\mathrm{FC}} - p_{\mathrm{FC}}\right|_y > 0, \\ 0, & \left|q_{\mathrm{FC}} - p_{\mathrm{FC}}\right|_y = 0, \\ -1, & \left|q_{\mathrm{FC}} - p_{\mathrm{FC}}\right|_y < 0, \end{cases} \tag{B.2}$$

where $|q_{\mathrm{FC}} - p_{\mathrm{FC}}|_y$ means the difference of the ordinate value of point q_{FC} (distance value from point q_{FC} to the y-axis) and the ordinate value of p_{FC} (distance value from point p_{FC} to the y-axis).

(6) Output connection strength S.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the Philosophy and Social Science Planning Project of Guangdong Province (GD19CYY13).

References

[1] M. Wahab, H. Amin, and F. Ahmed, "Shape analysis of pashto script and creation of image database for OCR," in *Proceedings of the 2009 International Conference on Emerging Technologies*, pp. 287–290, Islamabad, Pakistan, October 2009.

[2] S. Naz, A. I. Umar, R. Ahmad et al., "Urdu nastaliq recognition using convolutional-recursive deep learning," *Neurocomputing*, vol. 243, pp. 80–87, 2017.

[3] R. Ahmad, S. H. Amin, and M. A. U. Khan, "Scale and rotation invariant recognition of cursive pashto script using SIFT features," in *Proceedings of the 2010 6th International Conference on Emerging Technologies (ICET)*, pp. 299–303, Islamabad, Pakistan, October 2010.

[4] L. Yang, B. Yang, and X. Gu, "Adversarial reconstruction CNN for illumination-robust frontal face image recovery and recognition," *International Journal of Cognitive Informatics and Natural Intelligence*, vol. 15, no. 2, pp. 18–33, 2021.

[5] Z. Zhao, Q. Li, Z. Zhang et al., "Combining a parallel 2D CNN with a self-attention dilated residual network for CTC-based discrete speech emotion recognition," *Neural Networks*, vol. 141, pp. 52–60, 2021.

[6] Z. Zhang, P. Chen, X. Shi, and L. Yang, "Text-guided neural network training for image recognition in natural scenes and medicine," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 5, pp. 1733–1745, 2021.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[8] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015, https://arxiv.org/abs/1409.1556.

[9] F. N. Iandola, H. Song, W. Matthew, K. A. Moskewicz, J. D. William, and K. Kurt, "SqueezeNet: alexnet-level accuracy with 50x fewer parameters and <0.5MB model size," 2016, https://arxiv.org/abs/1602.07360.

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, Las Vegas, NV, USA, June 2016.

[11] A. G. Howard, M. Zhu, Bo Chen et al., "MobileNets: efficient convolutional neural networks for mobile vision applications," 2017, https://arxiv.org/abs/1704.04861.

[12] G. Huang, Z. Liu, L. Van Der Maaten, and Q. W. Kilian, "Densely connected convolutional networks." in *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, July 2017.

[13] L. Dinges, A. Al-Hamadi, M. Elzobi, and S. El-etriby, "Synthesis of common Arabic handwritings to aid optical character recognition research," *Sensors*, vol. 16, no. 3, p. 346, 2016.

[14] I. Ahmad, X. Wang, R. Li, M. Ahmed, and R. Ullah, "Line and ligature segmentation of Urdu nastaleeq text," *IEEE Access*, vol. 5, pp. 10924–10940, 2017.

[15] M. Amin Shayegan and S. Aghabozorgi, "A new dataset size reduction approach for PCA-based classification in OCR application," *Mathematical Problems in Engineering*, vol. 2014, Article ID 537428, 14 pages, 2014.

[16] I. Ahmad, X. Wang, Y. h. Mao, G. Liu, H. Ahmad, and R. Ullah, "Ligature based urdu nastaleeq sentence recognition using gated bidirectional long short term memory," *Cluster Computing*, vol. 21, no. 1, pp. 703–714, 2018.

[17] Z. Ahmed, K. Iqbal, I. Mehmood, and M. A. Ayub, "Ligature analysis-based Urdu OCR framework," in *Proceedings of the 2017 International Conference on Frontiers of Information Technology (FIT)*, pp. 87–92, Islamabad, Pakistan, December 2017.

[18] Q. U. A. Akram and S. Hussain, "Ligature-based font size independent OCR for noori nastalique writing style," in *Proceedings of the 2017 1st International Workshop on Arabic Script Analysis and Recognition (ASAR)*, pp. 129–133, Nancy, France, April 2017.

[19] S. Y. Arafat and M. J. Iqbal, "Two stream deep neural network for sequence-based Urdu ligature recognition," *IEEE Access*, vol. 7, pp. 159090–159099, 2019.

[20] U. Hayat, M. Aatif, O. Zeeshan, and I. Siddiqi, "Ligature recognition in Urdu caption text using deep convolutional neural networks," in *Proceedings of the 2018 14th International Conference on Emerging Technologies (ICET)*, pp. 1–6, Islamabad, Pakistan, November 2018.

[21] S. Malik, A. Sajid, A. Ahmad et al., "An efficient skewed line segmentation technique for cursive script OCR," *Scientific Programming*, vol. 2020, Article ID 8866041, 12 pages, 2020.

[22] R. Ahmad, S. Naz, M. Z. Afzal, S. H. Amin, and T. Breuel, "Robust optical recognition of cursive pashto script using scale, rotation and location invariant approach," *PLoS One*, vol. 10, no. 9, Article ID e0133648, 2015.

[23] S. Ahlawat, A. Choudhary, A. Nayyar, S. Singh, and B. Yoon, "Improved handwritten digit recognition using convolutional neural networks (CNN)," *Sensors*, vol. 20, no. 12, p. 3344, 2020.

[24] R. Ahmad, M. Z. Afzal, S. F. Rashid, M. Liwicki, and T. Breuel, "Scale and rotation invariant OCR for pashto cursive script using MDLSTM network," in *Proceedings of the 2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pp. 1101–1105, Tunis, Tunisia, August 2015.

[25] R. Achanta, F. Estrada, P. Wils, and S. Süsstrunk, "Salient region detection and segmentation," in *Computer Vision Systems. ICVS 2008*, A. Gasteratos, M. Vincze, and J. K. Tsotsos, Eds., pp. 66–75, Springer, Berlin, Germany, 2008.

[26] M.-M. Cheng, N. J. Mitra, X. Huang, P. H. S. Torr, and S.-M. Hu, "Global contrast based salient region detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 3, pp. 569–582, 2015.

[27] Y. Zhai and M. Shah, "Visual attention detection in video sequences using spatiotemporal cues," in *Proceedings of the 14th ACM international conference on Multimedia; MM '06*, pp. 815–824, Association for Computing Machinery, New York, NY, USA, October 2006.

[28] R. Achanta, S. Hemami, F. Estrada, and S. Susstrunk, "Frequency-tuned salient region detection," in *Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1597–1604, Miami, FL, USA, June 2009.

[29] M. Li, R. Zheng, S. Xu, Yu Fu, and Di Huang, "Manchu word recognition based on convolutional neural network with spatial pyramid pooling," in *Proceedings of the 2018 11th International Congress on Image and Signal Processing, Biomedical Engineering and Informatics (Cisp-Bmei 2018)*, IEEE, New York, NY, USA, October 2018.

[30] R. Zheng, M. Li, J. He, J. Bi, and B. Wu, "Segmentation-free multi-font printed Manchu word recognition using deep convolutional features and data augmentation," in *Proceedings of the 2018 11th International Congress on Image and Signal Processing, Biomedical Engineering and Informatics (Cisp-Bmei 2018)*, IEEE, New York, NY, USA, October 2018.

[31] S. Xu, M. Li, R.-R. Zheng, and S. Michael, "Manchu character segmentation and recognition method," *Journal of Discrete Mathematical Sciences and Cryptography*, vol. 20, no. 1, pp. 43–53, 2017.