Handwritten English Word Recognition based on Convolutional Neural Networks

Aiquan Yuan, Gang Bai, Po Yang, Yanni Guo, Xinting Zhao
College of Information Technical Science
Nankai University
Tianjin City, China
{*yuanaiquan123, yangpo, 2120110373, zxtrebecca*}*@mail.nankai.edu.cn, baigang@nankai.edu.cn*

***Abstract*—This paper presents a novel segmentation-based and lexicon-driven handwritten English recognition systems. For the segmentation, a modified online segmentation method based on rules are applied. Then, convolutional neural networks are introduced for offline character recognition. Experiments are evaluated on UNIPEN lowercase data sets, with the word recognition rate of 92.20%.**

***Keywords*- Handwritten English Word Recognition; Modified Word Segmentation; Convolutional Neural Networks;**

I. Introduction

Handwriting is one of the most important means of daily communication. During the last years, many popular studies and applications merged for bank check processing, mailed envelops reading, and handwritten text recognition in documents and videos [1] [2] [3].

This paper aims at a segmentation-based and lexicon-driven handwritten English recognition systems, as shown in Figure 1. After online segmentation of words, offline character recognition based on Convolutional Neural Networks (CNNs) are conducted.

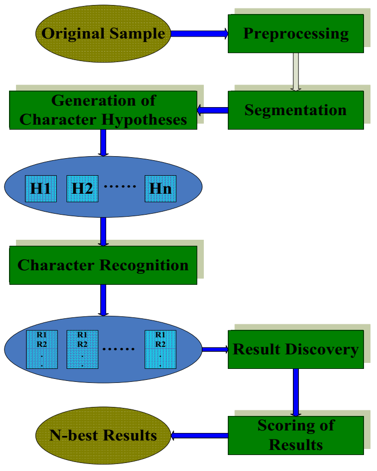


Figure 1. The basic architecture of *LeNet-5*

CNNs was brought about by LeCun [4] and caused huge attention immediately. It accepts a 2-D image as its direct input, thus the feature extraction is avoided. Many experiments with the CNN have seen moderately good performance.

Section II shows the preprocessing of original samples. A modified segmentation algorithm are described in Section III. In Section IV, character recognition based on CNNs are conducted. Section V interprets the process of word recognition. Experiments and discussions are provided in section VI.

II. Preprocessing

A. Resizing and Smoothing

As the values of sampling points of original samples are not in the same scale, they are resized to the range of 1-199 for the height, with the width in the same ratio. At the same time, the strokes is smoothed. With the consideration that, the location of current sampling point in the 2-D image is affected its neighbors, X coordinate of one sampling point is calculated as below:

$$X_p = \frac{\sum_{i=1}^{N} X_{p+i} * w_{p+i} + X_{p-i} * w_{p-i}}{\sum_{i=1}^{N} w_{p+i} + w_{p-i}} \tag{1}$$

N is the radius of neighborhood. The Y coordinate is obtained in the same way.

B. Slant Correction

As the strokes of samples are slant, adjusting slant angles of all samples to the same angle is necessary. Slant estimation is based on the statistics of slant angle of the sampling points with a certain distance. That is, points p_i and p_{i+5} are lined and the slant angle of this line is calculated. The mean slant angles is seen as the slant angle of the sample. The standard slant angle is set with $80°$. Improvements of slant correction are shown in section V.

C. Detection of Baselines

Information of baselines (AD-info) are important for consequential segmentation of word. This paper tries two solutions for detection of AD-info, as below.

1) by the sampling points: Most of the sampling points locate in center region (CR), while the sampling points in ascender region (AR) and descender region (DR) are much less. OTSU method [5] is applied for the location of upper baseline and lower baseline. To alleviate the inaccuracy, short strokes in AR or DR are filtered when detecting AD-info.

2) by the stationary points: Along the strokes, stationary points as maxima are mostly located near to upper baseline and those as minima are near to lower baseline. The algorithm applied in this paper can be in Table I. In experiments, T and H are initialized to 0.67 and 1 respectively.

Table I
DETECTION OF AD-INFO BASED ON STATIONARY POINTS

Step 1	initial T; get the sum of stationary points: $sNum$
Step 2	$H = 1$, go step 4, get the range of baselines: AC,CD;
Step 3	if AC and CD have no overlap, then return AC and CD; end; else $T = T \times 0.8$; go step 2;
Step 4	slide the window over the 2-D image in vertical direction; if the number of stationary points in the window: num; if $num > T \times sNum$ return current location of the window; elseif the window reaches the bottom of the 2-D image $H = H + 1$;

III. SEGMENTATION

A. Origin Segmentation

Paper[6] provided a rule-based methodology for online segmentation of handwritten English words. To begin with, all the local minima of strokes of one sample are seen as potential segmentation points. Then, they are checked with five features to get the final segmentation points. Thereafter, the origin strokes are segmented into several segments. The segmentation way applied in this paper is modified based on this rule-based method.

However, the internal defects of this method call our attentions. With it, some adjacent characters with certain traits can not be segmented apart, thus the word they are in will never be recognized correctly. Fortunately, after origin segmentation, some additional steps may alleviate this problem and decline the rate of over segmentation to some extent.

B. Adding New Segmentation Points

Error segmentation means that some legal segmentation points are missed by the old segmentation algorithm. These lost segmentation points are tried to find out and added to other segmentation points. Figure 2 shows one common stroke in which segmentation points are lost.

To detect and locate the segmentation point, some critical assistant points are defined. Thus the detection of new segmentation points consists of the detection of the sub-segment between P_o and P_b and the confirmation of the sub-segment between P_b and P_d.

C. Merging Fragmentary Segments in Multi-Strokes

Fragmentary segments are short segments or those which have no stationary points. A fragmentary segment itself can not compose a legal character. They are usually the secondary strokes of some characters or the result of over segmentation. For the fragmentary segments in original strokes that are split into several parts in segmentation (multi-strokes), they are merged into their neighbors. The overlaps of current fragmentary segment and its two neighbors are calculated. It will be merged into the neighbor with which the overlap is wider.

D. Merging Intimate Segments

Another way to alleviate over segmentation is to merge intimate segments that belonging to one character. The detection of intimate segments is a question to discuss. In this paper, the overlap between two adjacent segments in one multi-stroke are seen as their intimacy. As slant correction has been done,the overlap in horizontal direction could reflect whether the two segments are from one character or not. In experiments, if the overlap of two adjacent segments is wider both than 50% of them, they will be merged.

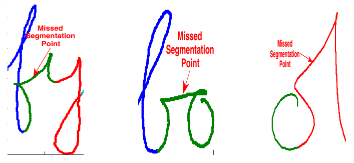
E. Disposing Fragmentary Segments in Isolated-Strokes

The isolated fragmentary segments are isolated-strokes that are not split by original segmentation. Some of them brings about additional difficulties for latter grouping of segments. Meanwhile, characters of some alphabets, such as i,j,t,etc, usually consist of two discontinuous strokes. The second strokes are usually the isolated fragmentary segments with irregular written styles. This paper only disposes isolated fragmentary segments that locate higher than the center line of CR. There are usually three disposals for them:

1) : If current segment is adjacent to several other segments, it will be split into several parts, each of which are merged into the those segments, as in Figure 3(a).

2) : If it is adjacent to only one other segment, it will be merged into the segment, as shown in Figure 3(b).

3) : If their are no other segments that adjacent to it, it will be removed, as shown in Figure 3(c).


Missed Segmentation Point
Missed Segmentation Point
Missed Segmentation Point

(a) 'by' (b) 'bo' (c) 'ol'

Figure 2. Samples that segmentation points are missed.

F. *Ordering of Segments*

The origin order of segments only shows the written sequence, but not their relations of spatial location. As the inputs for latter character recognition are from the grouping of segments according their spatial locations, ordering the segments according their horizontal locations in 2-D image is necessary. The horizontal location of one segment is defined by its centroid, which is the mean X-axis coordinates of all its sampling points.

IV. Character Recognition

With the segments after word segmentation, character recognition are based on two parts: grouping of segments and offline character recognition based on CNNs.

A. *Generation of Character Hypotheses*

Because of over segmentation, one segment is usually a part of rather than a complete character. So, it's essential to group several adjacent segments into one character hypothesis for latter character recognition. After reviewing the results of segmentation we find out that, almost all of legal characters are split into no more than three segments. This also shows over segmentation is controllable. Thereafter, generation of character hypotheses has three ways: one segment itself (G1), two adjacent segments together (G2), or three adjacent segments together (G3) as one character hypothesis. However, there still few characters that are split into four segments (G4). Theoretically, words including these like characters will have no possibility to be correctly recognized.

B. *Character Recognition*

1) Architecture of CNNs: A common model of CNNs is the LeNet-5 model [7], as shown in Figure 4. Each unit in it is connected to a local neighborhood in the previous layer, thus it can be seen as a local feature detector. The outputs of the units in the same position in different feature maps can be thought as a feature vector of the same area. Increasingly complicated features are extracted by neurons in the successive layers. Weight-sharing reduces the number of free parameters greatly. CNNs produces an output vector in each layer, each dimension of which detects features from different parts of feature maps in the previous layer. In experiments, except one input and 17 outputs, there are 6 neurons in layer C2 and layer S3, 16 neurons in layer C4 and layer S5, 120 neurons in layer C6, 200 neurons in layer F7.

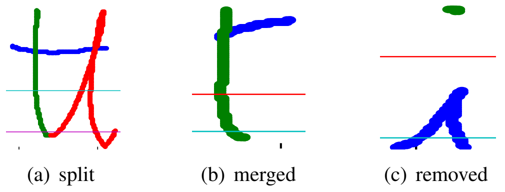

(a) split (b) merged (c) removed

Figure 3. Samples with isolated fragmentary segments.

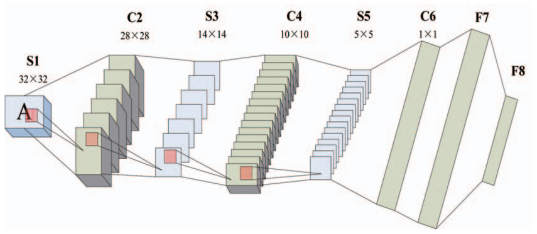


Figure 4. The basic architecture of *LeNet-5*

2) Character Recognition based on CNNs: As CNNs take 2-D image as its input, it's indispensable to produce 2-D images from grouped character hypotheses. First, adjacent points are connected. Then, strokes are extended to the width of 3 pixels. Finally, anti-aliasing and resizing the image to 32×32 are carried out. Rejection to results of character recognition is certainly important in handwritten recognition, for many samples are quite irregular, unconstrained, or illegal. In paper [8], the outputs of LeNet-5 are set with error-correcting codes (EC codes), thus LeNet-5 has the ability to reject illegal samples. EC codes are also applied in our experiments. When the feed-forward propagation of a sample finishes, the outputs of the CNNs are converted to a vector of EC code. Then, the Hamming distances between this EC code and all standard EC codes of 26 alphabets are calculated respectively. If one of the 26 Hamming distances is greater than a predefined rejection distance, its corresponding recognition result will be rejected. In the end, the left recognition results are sorted by Hamming distances in ascending order and TOPX results are thus obtained.

V. Word Recognition

After character recognition, we have the recognition results for character hypotheses of one word. Extracting the recognition results for the word from the recognition results of character hypotheses is the next step. Meanwhile, with more than one recognition result, it is quite essential to calculate the possibilities of them to be the label of the word. Moreover, scoring and sorting of these recognition results are also necessary for TOPX results.

A. *Result Discovery*

After word segmentation, a group of adjacent segments is seen as one character hypothesis and given to CNNs. So the relation between the word recognition result for current word and the character recognition results of its characters can be shown by a combine-tree. The word recognition results can be discovered by the traversal of the combine-tree. This paper applies two methods for discovering word recognition results: *recognition-after-segmentation* and *recognition-with-segmentation*. They differ mainly in the precedence relationship of word segmentation and character recognition. The first one means that character recognition begins after the finish of word segmentation, while the other one represents that character recognition is in progress every time one segment is segmented out. Both of the two ways are based on recursive implementation, as shown in Table II and Table III. Recognition-with-segmentation is a full traversal of the combine-tree. If each character is recognized correctly, then the right word recognition result will not be ignored. Comparing with recognition-after-segmentation way, recognition-with-segmentation is faster, as it can get the real-time recognition results and reject them immediately. However, if the right result for one character hypothesis is not in its R, this character will not be recognized correctly, so as the word.

Table II
ALGORITHM OF RECOGNITION-AFTER-SEGMENTATION

Step 1	if recursive conditions can not be met, return;
Step 2	get recognition results for segment N;
Step 3	for each result, dynamic pruning, get pruning result: res; if $res == failure$ skip current result; else N=N+1; go step 1;
Step 4	get recognition results for segment N,N+1;
Step 5	for each result, dynamic pruning, get pruning result: res; if $res == failure$ skip current result; else N=N+2; go step 1;
Step 6	get recognition results for segment N,N+1,N+2;
Step 7	for each result, dynamic pruning, get pruning result: res; if $res == failure$ skip current result; else N=N+3; go step 1;

Table III
ALGORITHM OF RECOGNITION-WITH-SEGMENTATION

Step 1	if recursive conditions can not be met, return;
Step 2	get recognition results for N: resG1; get recognition results for N,N+1: resG2; get recognition results for N,N+2,N+3: resG3;
Step 3	compare resG1,resG2,resG3, and get the best result R;
Step 4	if R==resG1, go step 5; if R==resG2, go step 6; if R==resG3, go step 7;
Step 5	for each result, dynamic pruning, get result: res; if $res == failure$ skip current result; else N=N+1; go step 1;
Step 6	for each result, dynamic pruning, get result: res; if $res == failure$ skip current result; else N=N+2; go step 1;
Step 7	for each result, dynamic pruning, get result: res; if $res == failure$ skip current result; else N=N+3; go step 1;

B. Dynamic Pruning

As traversal of combine-tree is time-consuming, dynamic pruning is fairly essential. Dynamic pruning happens when two conditions met : current result is longer than a predefined maximum of word lengths, or current result is not be found in the predefined lexicon, which means result discovery is lexicon driven. As searching for current result in the lexicon is taken each time dynamic pruning happens, the efficiency of searching algorithm restricts the efficiency of result discovery and the recognition speed.

This paper applies hash method for searching in lexicon. In actual implementation, the hash buckets are stored in a linear list, which provides direct access. The words in the hash buckets have two types: legal-words that are the labels of words, and sub-words which are extracted from legal-words. A sub-word of one word are composed of the prior n (n is less than the length of the word m) characters hypotheses. Thus, we can get $m - 1$ sub-words from the word. All of the sub-words are put into certain hash buckets together with those legal-words. The serial numbers of hash buckets are the hash values of words. The selection of hash functions is important, with consideration of the size of the hash list and the mean length of hash buckets. The hash function used in this paper are shown as below:

$$bucketNo = \sum_{i=1}^{m} C_i^2; \tag{2}$$

C_i is the i_{th} character hypothesis of the word. Additionally, as the serial number of hash buckets are discontinuous when hash buckets are continuous in storage, an index between them is generated. The mean length of hash buckets in this paper is 11.28. With dynamic pruning, if current result can not be found in the lexicon, pruning condition is met and the recursion process aborts. After result discovery finishes, all of the results which are not legal-words will be removed.

C. Scoring of Results

To order the word recognition results, we finally score each word result. The scoring method is based on the hamming distance in prior character recognition. With a word result, there are several Hamming distances for its characters. The shorter the mean Hamming distance of them, the higher the score of the word result is. As the mean Hamming distance ranges from 0 to the rejection distance, all of the scores of the word recognition results are scaled to a range from 60 to 100. Word recognition results are ordered with the scores in descending order. Then, word recognition results can be filtered by a predefined score threshold T.

The word recognition results with scores less than T will be filtered.

VI. Experiments and Discussions

Experiments are based on UNIPEN database [9]. A subset with 1791 samples that are randomly selected from UNIPEN lowercase words are made. UNIPEN dataset is used without any cleaning. All experiments except final word recognition are based on the subset of 1791 samples. Uppercase words or mixed case words are not in our experiments.

A. Preprocessing

In experiments, the standard slant angle is set with 80°. With slant correction, 9.62% percent of error segmented words are correctly segmented later. After the same training iterations, character recognition rate for segments from words of no slant correction and slant correction reach 92.1% and 94.5%, respectively. The error rates for the two ways of detection of AD-info is 8.04% and 2.07%, respectively.

B. Segmentation

There are three results for characters of words by original segmentation method: right-segmented, over-segmented and erroneously-segmented, which mean the stroke of one character is spilt into one segment, into several segments and is not split apart with its neighbors. A word with erroneously-segmented characters is also erroneously-segmented, for it will never be correctly recognized later. Among all of the characters, 51.72% are rightly-segmented and 45.74% are over-segmented. Meanwhile, 3.44% of them are not split apart with their neighbors, which results in 6.80% of words are erroneously-segmented, too.

The aim of adding new segmentation points is to decline the error rate of word segmentation. After this step, 47.06% of prior erroneously-segmented words are rightly-segmented and the error rate of word segmentation declines to 3.60%, which is a quite excellent performance. Rightly segmented, 3.20% of words thus have the opportunity to be rightly recognized. Of course, a tolerable cost emerges simultaneously that part of characters of 11.40% of words are more over-segmented.

Next, operations with fragmentary and intimate segments try to decline the rate of over segmentation. The mergence of fragmentary segments in multi-strokes alleviates the over segmentation of about 4.10% characters. Then, the mergence of intimate segments in multi-strokes removes over segmentation for 1.70% of all characters. Finally, what reduces both the rate of over segmentation and error segmentation greatly is the mergence of fragmentary segments in isolated-strokes, which prevents 11.39% of all characters from over segmentation. More importantly, 32.35% of words erroneously-segmented by original segmentation method are correctly.

C. Character Recognition

One problem that can not be ignored in the generation of character hypotheses is a few characters are split into more than three segments. With grouping ways of G1, G2 and G3 above, one character like this may be recognized as two characters. In this case, the possibility of being correctly recognized for the word with it will be much lower. After investigation of dataset, there are about 1.40% of words that have these like characters.

In experiments, the rejection distance is set with 6. The CNNs classifier used in experiments provides a mean recognition rate of all 26 classes is 97.28%, with 92.86% and 99.71% as the minimum and maximum of recognition rates. For TOPX results, the mean recognition rate increases to 99.26% in TOP2. The minimum recognition rate for one class is sometimes more critical than the mean recognition rate, because if the label of one character of a word does not shows in the TOPX results, the word therefore has no opportunity to be recognized correctly.

D. Word Recognition

In our experiments, the word recognition rates reach 92.20% with recognition-after-segmentation method and 73.16% with recognition-with-segmentation method respectively, which is feasible and convincing. There are several reasons for the wrongly-recognized words: wrong segmentation of words, the incomplete ways of grouping of segments and CNNs's limited recognition capability. When the score threshold T is set with 80, 0.87% of words are negatively influenced, while the average number of recognition results for one word descends to 2.29 from 11.76.

VII. Conclusion

As this word recognition system is segmentation dependent, exploring segmentation methods with better performances is considerately critical. Meanwhile, when online information of words is useless in character recognition by CNNs, a hybrid classifier based on both online and offline information needs to explore in the future.

Acknowledgment

Experiments in this paper are conducted on the UNIPEN dataset. Here, we acknowledge the organizer, and all participators and volunteers of UNIPEN project.

References

[1] R. Plamondon and S.N. Srihari. Online and off-line handwriting recognition: a comprehensive survey. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(1):63–84, 2000.

[2] L. Zhou, Y. Lu, and C. Tan. Bangla/english script identification based on analysis of connected component profiles. *Document Analysis Systems VII*, pages 243–254, 2006.

[3] A.L. Bianne-Bernard, F. Menasri, R.A.H. Mohamad, C. Mokbel, C. Kermorvant, and L. Likforman-Sulem. Dynamic and contextual information in hmm modeling for handwritten word recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(10):2066–2080, 2011.

[4] Y. LeCun and Y. Bengio. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361, 1995.

[5] N. Otsu. A threshold selection method from gray-level histograms. *Automatica*, 11:285–296, 1975.

[6] L. Zhang, G. Bai, and C. Xuan. A new methodology of online handwritten english character segmentation based on rules. In *Intelligent Computing and Intelligent Systems (ICIS), 2010 IEEE International Conference on*, volume 1, pages 670–674. IEEE, 2010.

[7] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[8] H. Deng, G. Stathopoulos, and C.Y. Suen. Error-correcting output coding for the convolutional neural network for optical character recognition. In *Document Analysis and Recognition, 2009. ICDAR'09. 10th International Conference on*, pages 581–585. IEEE, 2009.

[9] I. Guyon, L. Schomaker, R. Plamondon, M. Liberman, and S. Janet. Unipen project of on-line data exchange and recognizer benchmarks. In *Pattern Recognition, 1994. Vol. 2-Conference B: Computer Vision & Image Processing., Proceedings of the 12th IAPR International. Conference on*, volume 2, pages 29–33. IEEE, 1994.