A New Radical Based Approach to Offline Handwritten East-Asian Character Recognition

Kumar Chellapilla, Patrice Simard

A New Radical Based Approach to Offline Handwritten East-Asian Character Recognition

Kumar Chellapilla
Microsoft Research
kumarc@microsoft.com

Patrice Simard
Microsoft Research
patrice@microsoft.com

Abstract

East-Asian characters possess a rich hierarchical structure with each character comprising a unique spatial arrangement of radicals (sub-characters). In this paper, we present a new radical based approach for scaling neural network (NN) recognizers to thousands of East-Asian characters.

The proposed off-line character recognizer comprises neural networks arranged in a graph. Each NN is one of three types: a radical-at-location (RAL) recognizer, a gater, or a combiner. Each radical-at-location NN is a convolutional neural network that is designed to processes the whole character image and recognize radicals at a specific location in the character. Example locations include left-half, right-half, top-half, bottom-half, left-top quadrant, bottom-right quadrant, etc. Segmentation is completely avoided by allowing each RAL classifier to process the whole character image. Gater-NNs reduce the number of NNs that need to be evaluated at runtime and combiner-NNs combine RAL classifier outputs for final recognition.

The proposed approach is tested on a real-world dataset containing 13.4 million handwritten Chinese character samples from 3665 classes. Experimental results indicate that the proposed approach scales well and achieves a low error rate.

Keywords: East-Asian handwritten character recognition, radicals, convolutional neural networks.

1. Introduction

Offline handwritten East-Asian (Chinese, Japanese, and Korean) character recognition is a challenging pattern-recognition problem. Several aspects of these character sets make the problem difficult for conventional machine learning techniques such as Bayesian methods, neural networks, support vector machines, template matching and nearest neighbor algorithms. The most prominent aspect that contributes to the difficulty is the large number of classes/characters. The number of unique CJK (Chinese, Japanese, and Korean) characters can range from 4,000 to 80,000 [1].

Effective classification approaches that use only a single neural network [6] exist for most Latin languages such as English, French, German, etc that have less than a few hundred classes. However, these single neural network based approaches do not directly scale to the thousands of East-Asian characters. Trying to learn more than a couple of thousand characters using stochastic gradient descent (backpropagation) is a hard problem as the neural network weights do not converge even after several weeks of training.

Radical based approaches to CJK character recognition is not new [1-3]. Common approaches are:

a) extracting features holistically from the character image, or
b) structurally decomposing characters into component parts—usually strokes or radicals (groups of strokes).

The latter is almost always preferred as conventional holistic approaches do not scale to the tens of thousands of CJK characters. The decomposition schemes typically rely on explicit segmentation for strokes or radicals.

Traditionally, radicals are extracted either from the character skeleton or character strokes. Skeleton-based methods [4] first convert the character image into its skeleton graph. The skeleton graph is processed to discover radicals, their spatial relationships, and also character variations. Stroke-based methods [3,5] go one step further and decompose radicals into primitive strokes, usually straight lines with a few possible slopes. Structural analysis is used to integrate the information contained in these strokes and recognize the character. Computationally, stroke-based methods are faster than skeleton-based methods, but can be inferior due to ambiguities resulting from intersecting strokes.

In printed form (say from a font in printed documents) the character decomposition is very clear and one can devise clever approaches to segment the characters into not only radicals (skeleton-based), but also strokes (stroke-based). The hierarchical structure can then be exploited in a bottom up manner to recognizer the character. Efficient dynamic programming (Viterbi) algorithms can be designed to do the composition. Further, not all stroke combinations generate valid radicals, and not all radical compositions produce valid characters. This can be used to further improve the accuracy of the recognizer. Thus, stroke-based and skeleton-based approaches work well on machine printed CJK characters.

However, in handwritten form several of the character strokes merge into continuous curves. This also extends to strokes from different radicals in a character. Such merging makes radical and stroke segmentation becomes difficult, if not impossible. This merging is similar to cursive handwriting in Latin based western languages. The merging of strokes and radicals is so

common place that, common handwritten character forms (for each character) have evolved that do not look like their printed counter parts, but allow for easy reading and writing. Hence, the traditional radical and stroke decomposition approaches perform poorly on cursive characters.

In this paper, we propose a new holistic method that exploits the hierarchical nature of East-Asian characters through a divide-and-conquer approach.

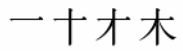
2. Radical Decomposition of CJK Characters

The East-Asian character system is very hierarchical. Each character is made up of one or more strokes (as large as 20). Some of the constituent strokes form commonly occurring sub-characters (parts of characters) called radicals. The hierarchical decomposition has the following properties

a) Characters are made up of one or more radicals
b) Constituent radicals occur in specific locations that are unique to the character (for example, left, top, or left-top quadrant, or as a band along the left and top parts of the character, etc)
c) Radicals are made up of strokes

Some radicals are also characters, i.e., there exist some characters that comprise a single radical, but not every radical is a character. Two examples of Chinese radicals are presented below:

i) 木 (tree) is a radical made up of 4 strokes and is also a character that represents a tree. The order of strokes in the character is:



ii) 目 (eye) is a radical made up of 5 strokes. It is also a character that represents an eye. The order of strokes in the character is:

Three Chinese character examples are given below:

a) 林 is the Chinese character for forest. It is made up of two tree radicals placed next to each other.
b) 相 is the Chinese character that means “each-other” or mutual. It is made up of a tree radical to the left and an eye radical to the right
c) 煞 is the Chinese character for “evil-spirit.” It is made up of three radicals 刍 攵 灬, with 刍 showing up in the left-top corner, 攵 showing up in the right-top corner, and 灬 showing up at the bottom of the character.

In the current paper, the term radical will be used to refer to any collection of strokes (with one or more strokes) that repeatedly occur as parts of characters. Under this definition, each of the 214 KangXi radicals (defined in the UNICODE standard) is a radical, but it is possible for several other custom radicals to be defined and used.

3. Character layout

Each character is composed of one or more radicals arranged in a special layout. The arrangement takes on very few selective forms. This layout of a character can be effectively captured by a tree representation of the character. An example decomposition of the Chinese character (煞) is presented in Figure 1.

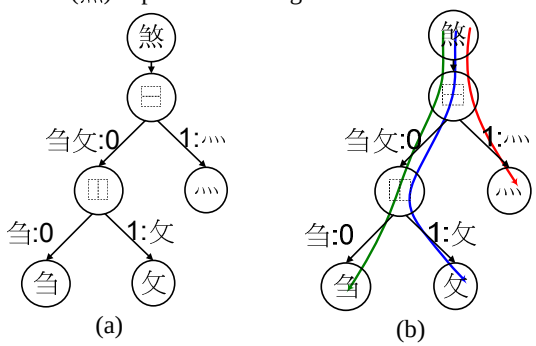
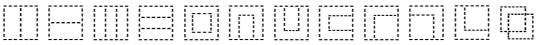


Figure 1. (a) Tree Representation of an example Chinese Character (煞) depicting the radical decomposition, and (b) three example radical paths.

This tree contains internal nodes (nodes that have children) and leaf nodes. Each of the leaf nodes is a radical. All possible selective forms can be represented using a few internal nodes called templates. These templates do not represent character pieces, but character layout. Interestingly, though there are tens of thousands of East-Asian characters, with several hundred radicals, the number of templates is only a handful (about a dozen or so). Here are 12 commonly used templates:



The first template defines the left and right parts of the character, the second defines the top and bottom parts of the character, etc. It is conceivable that other useful templates can be designed on an as needed basis. The tree representation provides interesting ways of

a) decomposing characters into radical-paths, and
b) grouping characters based on their (radical, template-path) pairs

3.1. Radical-Paths and Template-Paths

We now define “radical-path” and “template-path” through examples. Three example radical paths in the character 煞 are presented in Figure 1(b). These three radical paths are represented as

a) 0⿱\0⿰\刍 (green),
b) 0⿱\1⿰\攵(blue), and
c) 1⿱\灬(red)

Each node along the path is delimited by a back-slash (\). The different parts of each template are sequentially numbered starting at zero. For example, the template node, ⿱, contains two children corresponding to the top-part (0⿱), and bottom-part (1⿱).

Consider the green radical-path (Figure 1(b)) that the character 煞 belongs to. This radical-path is a particular instance of the template-path given by

0□\0□\?

where ? denotes a wild-card radical. This template path is a concrete way of capturing the property, "the top-left part of the character." Note that the corresponding radical path obtained by replacing the wild-card '?' with 刍 captures the property, "the top-left part of the character contains the radical 刍." The number of nodes in a template-path or a radical-path is its length. Here are a few example template paths of length 3

a) TOP-LEFT = 0□\0□\
b) TOP-RIGHT = 0□\1□\
c) BOTTOM-LEFT = 1□\0□\
d) BOTTOM-RIGHT = 1□\1□\

Using radical-paths one can decompose a character into its constituent radical paths. There exists one radical path for each radical present in the character.

3.2. Grouping Characters by Template-Paths

Given a radical path = (radical, template-path) pair one can ask the question: Which are all the characters that contain this (radical, template-path) pair? The answer would be a group of characters that share a particular radical in a given location in the character. A template path can then be viewed as a group of characters containing all possible radicals at a given location. This grouping of characters using (radical, template-path) pairs or simply a (template-path) is key to the radical based architecture of the East-Asian recognizer presented in this paper. Note that these template-path groups are not necessarily disjoint over different template-paths.

4. Distribution of Character Layouts

4.1. Data set

We used a large database of Chinese handwritten characters collected using a tablet-PC for our experiments. The dataset contained a total of 13.4 million samples of 3665 frequently used characters from GB2312 (simplified Chinese) written by 6668 users. The dataset is not uniformly distributed over all 3665 characters, but contains a natural distribution of characters. Character samples were collected in both print and cursive forms. Print data comprised 88% (11.8 million samples) of the data while cursive data comprised 12% (1.6 million samples) of the data.

In our offline recognition experiments, the ink samples are converted to images by rendering them to 29x29 bitmaps using a rectangular guide box that was used during collection. The size choice of 29x29 (same as in [6]) is somewhat arbitrary. It is conceivable that CJK characters with several strokes might benefit from larger bitmap sizes. Figures 2 and 3 present Chinese character ink samples rendered as 29x29 bitmaps. Print characters are easy to recognize, the cursive forms are much more difficult.

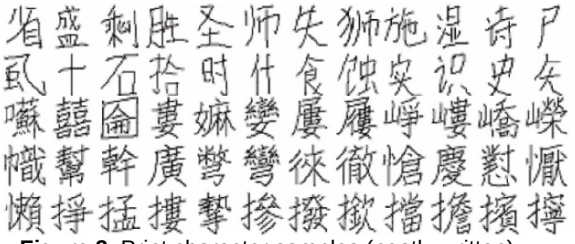


Figure 2. Print character samples (neatly written).

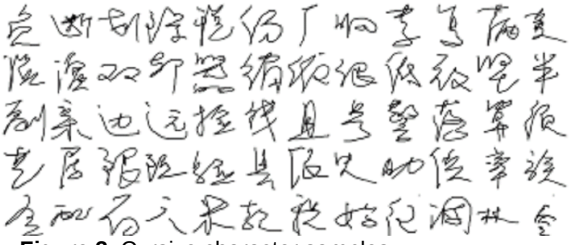


Figure 3. Cursive character samples.

The dataset was split into three sets: train (80%), validation (10%), and test (10%). The train dataset was used for training recognizers, while the validation set was used to monitor progress and determine when to stop training. All accuracies reported are on the test dataset.

4.2. Labeling Radical-Paths

Each of the 3665 characters was manually labeled with its unique tree decomposition. Printed forms of the characters as present in popular UNICODE fonts were used. First, all radicals were identified along with their radical paths. These radical paths were then collectively represented through a depth-first-search traversal of the character layout (see Figure 1).

4.3. Distribution

The labeled radical-path data was used to compute the distribution of characters over possible template-paths. Tables 1 and 2 present their distribution over common template paths of length 1. Note that these observations don't take into consideration characters that comprise only a single radical (template-paths of length 0). While Table 1 presents frequencies, Table 2 presents percentages. Each entry in the table has the form

{path}: (A,B) + ext(a,b)

where

{path} is a template-path of length one (root node),
A and a = number of radical paths,
B and b = number of characters containing radical paths with a given root node

Each template path begins with a template node which can have 2 or 3 children. Each child node can be a terminal node or a sub-tree. The (A,B) numbers indicate the number of children that were terminal nodes, while the extension numbers ext(a,b) indicate the number of characters that had sub-trees. It is more common to have

terminal nodes (radicals) as children rather than sub-trees.

Table 1. Distribution of characters and radical over template paths of length 1 (root node of the tree).

0 (267,833)+ext(49,62) 1 (359,814)+ext(87,102)	0 (255,2195) + ext(33,36) 1 (776,2014) + ext(156,221)
0 (3,16) 1 (16,16)	0 (6,80) 1 (76,77)+ext(6,6)
0 (23,12) + ext(2,2) 1 (34,61) + ext(4,4) 2 (47,63)	0 (16,35) 1 (29,35) 2 (20,33)+ext(4,4)
0 (5,31) 1 (26,29) + ext(4,4)	0 (3,4) 1 (4,4)
0 (14,133) 1 (100,113) + ext(38,42)	0 (11,31) 1 (26,29) + ext(4,4)
0 (2,10) 1 (10,10)	0 (2,2)+ext(2,2) 1 (2,2)+ext(2,2)

Table 2. Normalized distribution of characters and radical over template paths with percentages rounded to the nearest integer

0 (11%,12%)+ext(2%,1%) 1 (14%,11%)+ext(3%,1%)	0 (10%,31%) + ext(1%,1%) 1 (31%,28%) + ext(6%,3%)
0 (0%,0%) 1 (0%,0%)	0 (0%,1%) 1 (0%,1%)+ext(0%,0%)
0 (1%,0%) + ext(0%,0%) 1 (1%,1%) + ext(0%,0%) 2 (2%,1%)	0 (1%,0%) 1 (1%,0%) 2 (1%,0%)+ext(4%,0%)
0 (0%,0%) 1 (0%,0%) + ext(0%,0%)	0 (0%,0%) 1 (0%,0%)
0 (1%,2%) 1 (4%,2%) + ext(2%,0%)	0 (0%,0%) 1 (1%,0%) + ext(4%,0%)
0 (0%,0%) 1 (0%,0%)	0 (0%,0%)+ext(0%,0%) 1 (0%,0%)+ext(0%,0%)

Some interesting observations can be made from the percentages in Table 2:

1. The characters composed of left-right and top-bottom parts (marked in red) make up over 82% of the 3665 characters. Further, these parts contain mostly two radicals (one for each child node) as indicated by low ext(a,b) percentages.
2. The ratio (B/A) gives the average number of characters corresponding to each unique radical path.

$$\frac{B}{A} = \frac{(\#\text{chars containing root template-node})}{(\#\text{radical-paths})}$$

Examining (B/A) we note that

a) For , if we know the right radical (and that the character is of type), then on average we reduce the search space from 3665 characters down to nearly 1.5 characters i.e., recognition is almost done!

b) For , if we know the left radical (and that the character is of type), then we reduce the search space from 3665 characters down to nearly 6 characters.

c) For , if we know the top or bottom radical (and that the character is of type), then we reduce the search space from 3665 characters to nearly 2 characters.

These observations strongly indicate that character-parts (radicals or strokes) that are important for recognition are not uniformly distributed over all regions of the character. Further, this distribution varies over which radicals are present in which locations. For example, knowing the right radical of a type of character is more useful than knowing the left radical.

The above distribution was also computed using only 1000 of the 3665 characters. Very little change in the distribution of the template paths was found between the 1000 and 3665 characters. We conjecture that this distribution is likely to be similar, if recomputed over a much larger set of Chinese characters.

5. Radical based recognizer design

The end-to-end recognizer comprises neural networks arranged in a graph. Each NN is one of three types: a radical-at-location (RAL) recognizer, a gater, or a combiner.

5.1. Radical-At-Location (RAL) recognizer

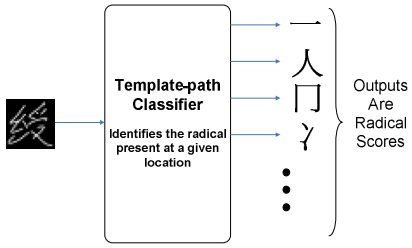


Figure 4. An example template-path classifier.

Each Radical-At-Location classifier, such as the one shown in Figure 4, has a unique template-path associated with it. It examines the whole character, and recognizes the radical present at the location specified by its template-path. Its input is a character (image/ink) and its output is a list of radical scores. Passing the whole character image as input avoids the need for explicitly segmenting the character to find the radical. Note that since not all (radical, template-path) pairs are valid, only those radicals that can actually appear at the location specified by the template-path are present as output classes.

5.2. Template-node gater

The template-node gater is a classifier that determines which template node is present at a specific depth and along a specific path from the root node. In the simplest case, this could be the root template node itself (one of 12 or so possible template nodes or a subset thereof). For example, a root template node of would imply that the character has a left- and a right-part. It is conceivable that for characters with very deep hierarchies a template-path gater can be designed for internal nodes at higher depths. Figure 5 presents an example template-path gater for the root node. The root template-node gater examines the whole character to determine the root node template

for the character tree. Not all possible output templates (about 12 of them) need to be present – any subset of the templates would also suffice.

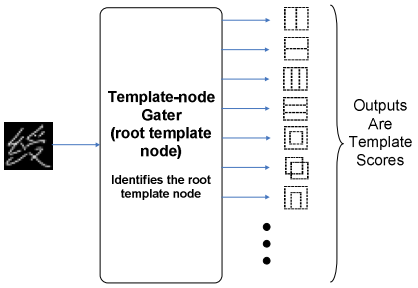


Figure 5. An example template-node gater.

5.3. Combiners and End-to-End recognizer

The template-path combiner combines the outputs of two or more RAL recognizers. Figure 6 presents an example gater—RAL—combiner architecture for recognizing characters made up two halves (left-right or top-bottom). Each individual part (left, right, top, or bottom) is recognized by a different RAL. All four RAL outputs are combined to produce a final distribution of probabilities over the supported characters. Note that only those characters that contain a left- and a right-part show up at the output of the combiner. Depending on the settings the individual RAL classifiers can be further decomposed. For example, the right-part of the fourth output character 涸 can be further decomposed to two more levels.

6. Experiments

Two recognizers with slightly different architectures were trained to recognize 3,665 Chinese (CHS) characters. The first one, shown in Figure 7, does not use any gaters, while the second one, shown in Figure 8, uses an LR vs TB gater. 3,076 (84%) of the 3,665 characters have an LR (⿰, left-radical, right-radical) or a TB (⿱, top-radical, bottom-radical) form. The rest of the 589 characters are either base radicals or have one of the other 10 root template nodes. In both recognizers, a single CNN was built to recognize these 589 characters. In the figures, this classifier is labeled as the "Radical+Misc" Classifier. The four RAL classifiers have the following template-paths:

a) L-Classifier = 0⿰\
b) R-Classifier = 1⿰\
c) T-Classifier = 0⿱\
d) B-Classifier = 1⿱\

Each RAL-NN was a convolutional neural network with two convolutional layers, one hidden layer and one output layer [6]. The first and second convolutional layers contained 5 and 50 nodes, respectively. The convolutional kernels was 5x5 in size with each layer down-sampling the convolved output by a factor of 2. The number of hidden nodes was 200. The number of output nodes varied with the RAL (Table 4).

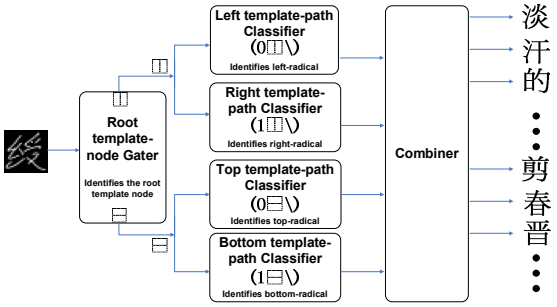


Figure 6. An example radical based classifier for characters made up of two radicals.

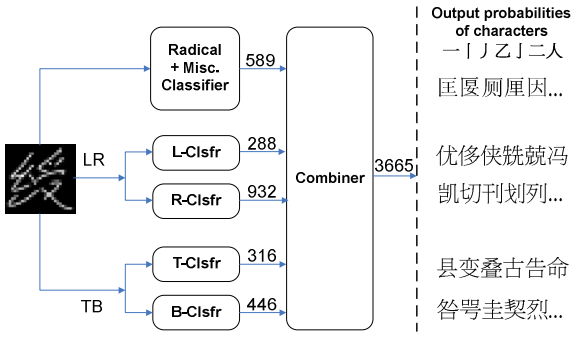


Figure 7. An example recognizer without gaters.

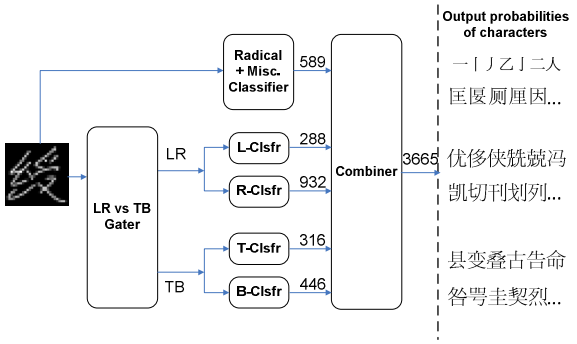


Figure 8. The example recognizer with a gater.

Simple linear combiners with sigmoid nonlinearity (one layered neural networks) were used. Each combiner computed a weighted sum of its inputs and applied the nonlinearity to obtain each output. The large number of inputs (2,570) and outputs (3,665) result in a very large combiner weight matrix. However, since not all RAL output combinations are feasible, the weight matrix is very sparse and contains very few non-zero weights. For example, a dense weight matrix would have 2,570 x 3,665 = 9,419,050 weights. However, given that only 3,076 of these combinations result in actual characters, we expect only about 2 to 10 non-zero weights per output character, producing (2 to 10) x 3076 = 6,152 to 30,760 non-zero weights. Several combiners were built for combining the left and right RALs, the top and bottom RALs, all four RALs, and also the combiners shown in Figures 7 and 8.

The gater in Figure 8 has the same architecture as the RALs, but is much smaller with only 50 hidden and 2 output nodes. The gater is 4-5 times faster and can improve throughput by up to 40%, at a small decrease in accuracy (see Table 3). A gater with 3 outputs was also trained to tell the LR characters apart from the TB characters and also the rest of the 589 "Radical+Misc" characters. All networks were trained using stochastic gradient descent (backpropagation). For the combiners, weight decay was used as a simple regularization approach to ensure that most of these weights were zero or had very small values.

7. Results

Tables 3, 4, and 5 present the accuracies of the Gater, RAL, and Combiner classifiers, respectively. The two output gater achieves a high accuracy of 98.72% and is very successful at telling whether the character has a left-right or a top-bottom form. However, the three output gater has a much lower accuracy due to confusions between the (LR+TB) character set and the rest of the 589 (Radical+Misc) character set. Thus the LR vs TB gater was used in the final recognizers (Fig. 8).

Each of the RAL classifiers has less than 1000 outputs and thus is able to learn well. The (Radical+Misc.) classifier has the best accuracy of 94.48%, while the R-RAL classifier has an accuracy of only 85.80%. We conjecture that the simplicity of the characters produces such good results for the (Radical+Misc.) classifier while the large number of output classes and right radical variability makes recognition difficult for R-RAL. Overall, it is easier to recognize the left-, top-, and bottom-radicals in Chinese characters than the right-radical.

If the recognition errors of RALs were independent, then during combination we would expect the error rates to add together. For example, with an L-RAL error rate of 7.2% and an R-RAL error rate of 14.2%, a naïve combiner would generate an error rate of over 21.4%. However, the LR-combiner produces an error rate of only 9.5%. Similarly, the TB-combiner produces a low error rate of 5.47%. Both of these results indicate that the radical decomposition approach has potential for scaling well by exploiting radical distribution patterns, like the ones shown in Tables 1 and 2. Lower error rates may be achieved if more powerful combiners are used.

The end-to-end classifier accuracies are presented in Table 6. The accuracy on print characters is over 95.5%. However, the cursive accuracy is only around 70%. Recognizing cursive characters appears to be a much more difficult as identified in several related research efforts [1-3]. From Table 3 we note that almost 82% of the characters are expected to contain LR or TB radical parts, while the remaining 18% of the characters make up the radicals and characters with other forms. Comparing the two end-to-end recognizers with and without the LR vs TB gater, we note that the gater produces an almost 40% increase in throughput with only a marginal reduction in accuracy.

Table 3. Gater accuracies.

Classifier	#Chars	#Outputs	Accuracy (%)
LR vs TB Gater	3076	2	98.72
LR vs TB vs Rest Gater	3665	3	74.95

Table 4. Radical-At-Location (RAL) accuracies.

Classifier	#Chars	#Outputs	Accuracy (%)
L-RAL	2213	288	92.80
R-RAL	2213	932	85.80
T-RAL	863	316	90.68
B-RAL	863	446	90.47
Radical+Rest	589	589	94.48

Table 5. Classifier Combiner accuracies

Classifier	#Chars	#Outputs	Accuracy (%)
LR-Combiner	2213	2213	90.50
TB-Combiner	863	863	94.53
LRTB-Combiner	3076	3076	89.50

Table 6. End-to-end Recognizer accuracies

Classifier	#Chars	Accuracy (%)	
		Print	Cursive
With Gater (Fig. 8)	3665	95.56	69.13
No Gater (Fig. 7)	3665	95.81	70.64

8. Conclusion

We presented a new radical based approach for scaling neural network (NN) recognizers to thousands of East-Asian characters. Experimental results on a database of 13.4 million Chinese character samples from 3665 classes show that this method achieves good scalability by exploiting correlations between spatial components (radicals) in the characters.

References

[1] C-L Liu, S Jaeger, and M Nakagawa (2004), "Online Recognition of Chinese Characters: The State-of-the-Art," *IEEE PAMI*, v. 26, no 2.

[2] D Shi, RI Damper, and SR Gunn (2003), "Offline Handwritten Chinese Character Recognition by Radical Decomposition," *ACM Trans. Asian Lang. Inf. Process.* 2(1): 27-48.

[3] A-B Wang, K-C Fan (2001), "Optical recognition of handwritten Chinese characters by hierarchical radical matching method," *Pattern Recog*. v. 34, no 1, pp. 15-35.

[4] WWS Ip, KFL Chung and DS Yeung (1995), "Offline handwritten Chinese character recognition via radical extraction and recognition", *ICDAR'95*, pp. 185-189.

[5] CW Liao and JS Huang (1990), "A Transformation invariant matching algorithm for handwritten Chinese character recognition", *Pattern Recognition*, v. 23, no. 11, pp. 1167-1188.

[6] P. Y. Simard, D. Steinkraus, and J Platt, (2003), "Best Practice for Convolutional Neural Networks Applied to Visual Document Analysis," in *ICDAR'03*, pp. 958-962.