QuickStroke: An Incremental On-line Chinese Handwriting Recognition System

Nada P. Matić John C. Platt* Tony Wang†
Synaptics, Inc.
2381 Bering Drive
San Jose, CA 95131, USA

Abstract

This paper presents **QuickStroke***: a system for the incremental recognition of handwritten Chinese characters. Only a few strokes of an ideogram need to be entered in order for a character to be successfully recognized. Incremental recognition is a new approach for on-line recognition of ideographic characters. It allows a user to enter characters a factor of 2 times faster than systems that require entry of full characters. Incremental recognition is performed by a two-stage system which utilizes 68 neural networks with more than 5 million free parameters. To enable incremental recognition, we use specialized time-delay neural networks (TDNNs) that are trained to recognize partial characters. To boost the recognition accuracy of complete characters, we also use standard fully-connected neural networks. Quickstroke is 97.3% accurate for the incremental writer-independent recognition of 4400 simplified GB Chinese ideograms.*

1 Introduction

As computer technology improves and becomes more widespread, writers of ideographic characters need more friendly human-machine interfaces. The keyboard is not a friendly interface for entering the characters of ideographic languages like Chinese, whose alphabet consists of a very large number of symbols — there are 4400 simplified GB Chinese ideograms. In order to enter Chinese characters using a keyboard, one has to memorize and execute complex key sequences or, alternatively, enter equivalent symbols for phonetic representation. An ideal input device for ideographic text would use on-line handwritten input. Common touch-sensitive input devices like TouchPads, tablets, or PDAs are all capable of capturing such on-line handwriting data in the form of pen or finger trajectories.

*Current address: Microsoft Research, 1 Microsoft Way, Redmond, WA 98052, USA
†Current address: Nortel Networks, 4555 Great American Parkway, Santa Clara, CA 95054, USA

This paper describes **QuickStroke**, a writer-independent on-line Chinese character recognition system for printed and partially cursive characters. This system is designed to provide an ideographic input method that is both intuitive and very fast. QuickStroke recognizes handwritten characters with very high accuracy and is robust to stroke number and stroke order variation in these characters.

Numerous methods have been previously proposed for on-line Chinese character recognition [1]. In most cases, previous work did not lead to practical high-accuracy systems either because experiments were performed on a reduced number of classes, or because limitations on writing order were imposed to obtain an acceptable level of accuracy for the writer-independent task.

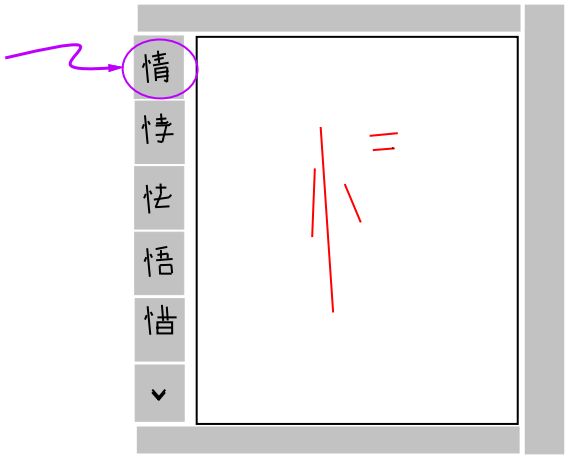

Figure 1. The user interface for QuickStroke. Quickstroke displays its hypothesis list sorted by probability. The user can select out of the hypothesis list, or add more strokes to the same character if no candidate is correct.

QuickStroke improves on previous work by performing *incremental* recognition. Incremental recognition allows a character to be accurately recognized after only a few strokes of the input ideogram have been drawn (see Fig. 1). Incremental recognition increases character entry

speed twofold by reducing the number of required strokes on average by a factor of 2.

QuickStroke builds upon previous work in neural networks for on-line handwriting recognition. Examples of such neural networks for Latin text include time-delay neural networks (TDNNs) [2], convolutional neural networks [3], or standard multi-layer perceptrons [4].

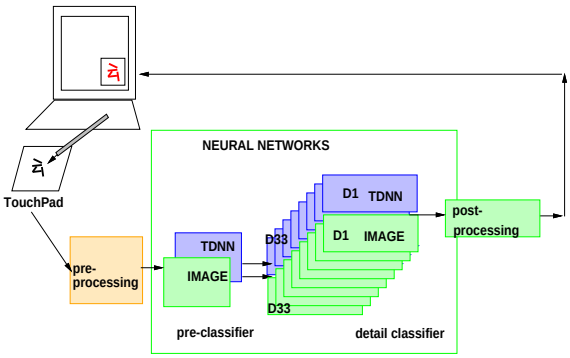

Figure 2. Our system consists of 68 neural networks that are divided into pre-classifier and 33 detail classifiers. Each classifier consists of a pair of neural networks. There are more than 5 million free parameters in all of the neural networks.

In order to obtain high accuracy for a classification task that has 4400 classes, we use a two-stage architecture that first identifies a subset of classes that are possible, and then recognizes an individual class out of that subset [5] (see Fig. 2). QuickStroke is unique because it uses only the first three strokes as a basis for the first stage of classifiaction. This allows it to perform incremental recognition on a large number of classes.

Furthermore, in order to have high accuracy for both partial and full characters, each classifier consists of an ensemble of two neural networks, each with a different input representation and architecture that is tuned for either partial or complete characters. These different input representations make errors more orthogonal, hence the overall error rate is reduced [6].

2 Description of the System

The first stage of QuickStroke (called the *pre-classifier*) performs coarse classification by placing the character into one of 33 possible character subsets. The second stage is then responsible for final classification. The second stage consists of 33 separate *detail classifiers*: D1-D33.

Every classifier in the system (e.g. pre-classifier and all detail classifiers) is implemented as a combination of two neural networks.

The first neural network in each pair is a two-layer, time-delay neural network (TDNN)[2] which exploits the sequential, time-varying nature of the recognition problem (e.g. stroke order). In our system, the TDNN is designed and trained in order to optimize the recognition of a partial character. The second network is a two-layer feed-forward perceptron, trained with back-propagation [7] and optimized to recognize complete characters. Each component classifier provides probability estimates for each individual class in the form of an output vector of confidences. QuickStroke integrates the information from two component classifiers by simply averaging individual probabilities. It picks the class with the maximum probability as the final answer of a particular classifier.

2.1 Coarse Classification

The pre-classifier performs coarse classification: it limits the number of character candidates from the original set of classes to a much smaller group. Coarse classification speeds up recognition and concencentrates the neural network resources to disambiguate more confusable characters.

In QuickStroke, we partition the 4400 GB classes into 33 groups. These groups are generated by a bootstrapping procedure. The pre-classifier uses only the first three strokes from each character. We choose an initial set of classes for each group based on the similarity of these first three strokes. We then train a first prototype of the pre-classifier on this limited set of classes and use bootstrapping (similar to [8]) to label all available training data into groups. A final pre-classifier is trained using input-output pairs from all the available training data. The output of the pre-classifier is a vector of 33 probabilities, one per pre-classifier class.

To accommodate for the natural variability in writing, we allow the 33 groups to overlap: each character class can be assigned to one or more groups. For example, the first three strokes of a particular character can be written using different stroke order by different writers, which in turn leads to one variant of a class belonging to one group and another variant belonging to an alternate group. We assign these multiple groups by presenting 40 training samples from different writers to the pre-classifier and assigning a class to each group output by the pre-classifier if that output occurs more than a once in the 40 samples.

2.2 Detail Classification

Each pre-classifier class represents a group of characters that consists of a subset of the Chinese alphabet. Each detail classifier was trained to distinguish between classes in a particular group. The number of classes n that each detail classifier was trained on ranges from 29 to 567. A pair of neural networks performs a detail classification for each one of these 33 groups. The output of the detail classifier is a vector of n probabilities, one per detail classifier.

To improve the accuracy of the system, we evaluate two detail classifiers corresponding to the two top pre-classifier answers. We combine two probability vectors into one and sort it into a hypothesis list.

Finally as a simple post-processing step, the incremental recognizer modifies the hypothesis list. Characters that have fewer strokes than the user entered are deleted from the list. Characters that are subsets of other characters are promoted towards the front of the list.

If the correct character appears on the hypothesis list, the user can select it as a final answer of the system, otherwise additional strokes can be provided to QuickStroke for the new recognition attempt. Notice that the group identity will not change after the first three strokes, therefore there is no need to call the pre-classifier after additional stokes have been entered.

3 Pre-processing

After simple noise removal, we scale a group of strokes that can represent either a partial character or a complete character so that they lay in the box $[0, 1] \times [0, 1]$. Each group of strokes is scaled so that the maximum of its height or width becomes 1. This makes our recognizer independent of size variation. Afterward, scaling input strokes are re-sampled so that sample points are spaced evenly along the arc length of each stroke [2].

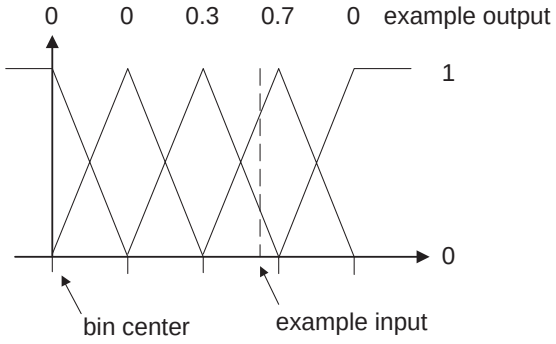

Figure 3. Continuous parameters are encoded by evaluating several triangular basis functions, each centered on a bin center. The output of the encoding is a vector of values.

Features are then extracted for every reparameterized point of the stroke: the horizontal and vertical position of the point and direction of the stroke at the point. The system uses these local features to generate *Directional Feature Maps* (DFMs). A DFM takes this local information and produces a three-dimensional tensor that contains joint directional and spatial information (similar to [9, 3]). We perform an interpolated encoding of each of the continuous variables x, y, and direction (see Fig. 3), and then take the tensor outer product of the resulting encoded vectors. This encoding makes the training of neural networks easier and allows the first layer of each network to compute piecewise linear functions of the original input variables.

We use a different number of basis functions per input for the different neural networks. For the TDNN, we use five membership functions for x and y encoding, and eight for direction which results in a 200-dimensional input vector.

In addition to the DFM, we use *Geometrical Features* (GFs) that represent the relationship between two consecutive strokes. GFs capture the spatial difference between the endpoints of the previous stroke and the endpoints of the current stroke. A total of four continuous quantities are computed and then interpolated and encoded into a single 64-dimensional vector. Again the GF for each pair of strokes is supplied to the corresponding copy of the first layer of the TDNN.

Two feature sets are computed for standard multi-layer networks. For the pre-classifier, a single DFM is generated for the input consisting of the first three strokes. Similarly, a single DFM is calculated for all detail classifiers, where the input to the DFM consists of all the strokes of the partial or complete character. The multi-layer networks use an $8 \times 8 \times 8$ DFM, which provides more spatial resolution than the DFMs for the TDNNs. The second feature set for the standard networks consists of information about the location of endpoints of the strokes that are also interpolated and encoded, leading to a 25-dimensional vector. The input dimensionality of these standard multi-layer networks is thus 537.

Because these standard neural networks receives spatial and no temporal information from all of the strokes, we refer to them as the "image" neural networks.

4 Neural Network Architectures

In the current implementation, the pre-classifier and all 33 of the detail classifiers each consist of a TDNN with a novel architecture and an "image" neural network.

For the TDNN of the pre-classifier, the first (hidden) layer is replicated three times. Each copy of the first layer receives a DFM and a GF from one of the three input strokes. This is in contrast to prior uses of TDNNs [2], where each copy of the first layer receives input from a small time-slice of handwriting. There are 80 neurons in each copy of the hidden layer, which extract higher-level features from each stroke. The output (classification) layer is fully connected to all units in the hidden layer. The output layer has 33 outputs, one for each group.

The TDNN of the detail classifier is quite similar to the pre-classifier TDNN, except that each detail TDNN has 25 copies of the first layer (corresponding to the first 25 strokes of the character), rather than three copies as in the pre-classifier TDNN. Each hidden layer consists of 20 neurons.

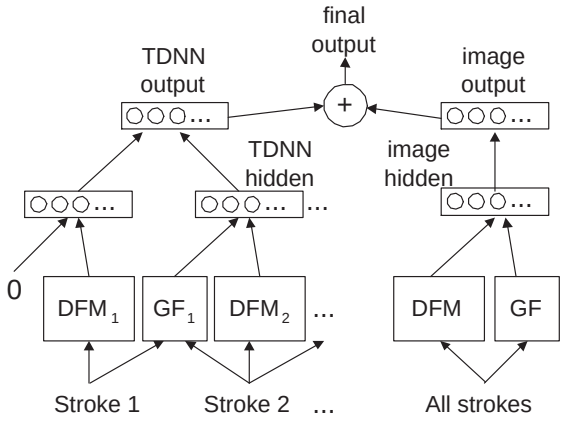

Figure 4. Architecture for a Single Classifier

The "image" neural network of the pre-classifier is a standard multi-layer perceptron, with sigmoidal non-linearities trained with back-propagation [7]. The network has two layers of trainable weights, with 33 output neurons and 350 hidden neurons.

The "image" neural network for each detail classifier has 200 hidden units, and n output units. The 200 hidden units are fully connected to the 537-dimensional input feature vector, representing the "image" of the complete character.

4.1 Neural Network Training

All the networks in the system are trained to recognize partial or complete input characters that belong to a set of 4400 GB Chinese characters. This represents an extended set of the Standard Level 1 GB character set. Our training set consists of samples from 60 writers. Additional samples from a disjoint set of 20 writers were used for validation purposes.

The weights are adjusted during a supervised training session, using back-propagation [7] which performs gradient descent in weight space with a cross-entropy loss function. By minimizing this error, neural network learning algorithms implicitly maximize margins, which explains their good generalization performance despite large capacity (as in the case of our image networks). In the case of TDNNs, the training algorithm must adjust the weights of the convolutional kernels. This is implemented by "weight sharing" [10]. In our present system, the capacity control is handled by "early stopping", detected by cross-validation. We performed cross-validation by computing the performance of the system on a small set of validation patterns.

In order to train a fully incremental recognizer, we place partial characters as well as complete characters with the same target label into the training set. This extended training set requires more training time, but resulted in improved recognition of partial characters, while retaining good recognition for complete characters.

4400 GB classes	*Top1*	*Top2*	*Top3*
partial character accuracy	97.3%	98.25%	98.47%

Table 1. Incremental recognition rate for the 4400 most commonly used GB Chinese characters

5 Experiments and Results

The output of each detail classifier is a vector of probabilities of length n, where n is a number of classes in the particular group. There is one output for each class in the detail classifier group. The probability vectors for each type of classifier (e.g. "image" and "TDNN") are averaged together in order to obtain a single probability vector per detail classifier.

We have tested the performance of QuickStroke on a test set that consists of twenty writers distinct from the writers used for training and validation. Each test character was tested in the "incremental" mode. For test characters having 3 or more strokes, partial characters consisting of $3 \ldots m$ strokes are evaluated and candidate lists generated. If at any point in this process the partial character is recognized as the top candidate the corresponding character is considered recognized. m is a minimum of 25 strokes and number of strokes in the particular test character. Table 1 shows the performance on the test set for the 4400 most commonly used GB Chinese characters. We have also tested the performance of Quick Stroke on complete characters: the top 1 accuracy is 96.3%.

These experiments prove that QuickStroke has excellent accuracy for both partial and complete characters.

Quickstroke provides both excellent accuracy and increased writing speed. Our testing results indicate that, on average, only half of the total number of input strokes need to be entered in order for the system to recognize the character (i.e., 6 strokes out of an average of 12). Thus, users of Quickstroke can enter characters much faster than alternative input methods.

6 Conclusions

In this paper, we present **QuickStroke**, a writer-independent, commercially-successful ideographic character input method that is both fast and very accurate. QuickStroke is trained to perform incremental recognition of Chinese characters which considerably speeds up text input. By combining both TDNNs and "image" neural networks, Quickstroke is robust to variation in both the stroke order and the stroke number of characters, while still retaining the capability of recognizing partial characters.

7 Acknowledgments

We wish to thank Suli Fay and Monte Wang for their suggestions and help during the design and development of the QuickStroke system.

References

[1] S. W. Lee. Special issue: Oriental character recognition. *Pattern Recognition*, 30(1253-1254):1031–1044, 1997.

[2] I. Guyon, J. Bromley, N. Matić, M. Schenkel, and H. Weissman. Penacée: A neural network system for recognizing on-line handwriting. In D. Van Hemmen and et al., editors, *Models of Neural Networks*. Springer-Verlag, 1995.

[3] Y. Bengio, Y. Le Cun, and D. Henderson. Globally trained handwritten word recognizer using spatial representation, convolutional neural networks and hidden markov models. In *NIPS-6*, San Mateo CA, 1994. Morgan Kaufmann.

[4] R. Lyon and L. Yaeger. On-line hand-printing recognition with neural networks. In *5th Intl. Conf. Microelectronics for Neural Networks and Fuzzy Systems*, Lausanne, Switzerland, 1996.

[5] Y. Mori and K. Joe. A large-scale neural network which recognizes handwritten kanji characters. In D. Touretsky, editor, *NIPS*, volume 2, pages 415–422. Morgan Kaufmann, 1990.

[6] L. K. Hansen and P. Salamon. Neural network ensembles. *IEEE Trans. Neural Networks*, 12(10):993–1001, 1990.

[7] C. M. Bishop. *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford, England, 1995.

[8] N. Matic, I. Guyon, J. Denker, and Vapnik V. Computer aided cleaning of large databases for character recognition. In *11th IAPR International Conference on Pattern Recognition*, volume II, pages 330–333. IEEE Computer Society Press, 1992.

[9] J. Tskumo and H. Tanaka. On-line hand-printing recognition with neural networks. In *9th ICPR*, 1988.

[10] Y. Le Cun. Generalization and network design strategies. In R. Pfeifer, Z. Schreter, F. Fogelman, and L. Steels, editors, *Connectionism in Perspective*, Zurich, Switzerland, 1989. Elsevier.