Memory Conscious Sketched Symbol Recognition

Caglar Tirkaz
Sabanci University
caglart@sabanciuniv.edu

Berrin Yanikoglu
Sabanci University
berrin@sabanciuniv.edu

Metin Sezgin
Koc University
mtsezgin@ku.edu.tr

Abstract

Automatic sketch recognition is used to enhance human-computer interaction by allowing a natural/free form of interaction. It is a challenging problem due to the variability in hand drawings, the variation in the order of strokes, and the similarity of symbol classes. Since sketch recognition requires real time processing, the speed of the classifier is important. Another important issue is how to deal with very large data sets and/or large number of classes, as these also effect training and testing speed, making certain approaches infeasible. In order to deal with these issues, we present a memory conscious sketch recognition system that processes the data to retain only a few templates per class as prototypes; and furthermore, the query and prototypes are subsampled without loosing important information. The system also uses a cascaded combination of classifiers, to improve speed, as well as recognition accuracy. Results obtained using the public COAD and NicIcon databases are comparable to previous results obtained for these databases.

1. Introduction

Sketching is the freehand drawing of shapes and is a natural mode of interaction. Automatic sketch recognition refers to the recognition of pre-defined symbols (e.g. a resistor, transistor) or free-form drawings (e.g. an unconstrained circuit drawing); and is often used in the fields of education, engineering and design. There are many approaches in the literature for sketched symbol recognition. These include gesture-based approaches that treat the input as a time-evolving trajectory [6, 12], image-based approaches that rely only on image statistics (e.g., intensities, edges) [9], or geometry-based approaches that attempt to describe objects as geometric primitives satisfying certain geometric and spatial constraints [3, 2].

In this paper, we present a classifier architecture that achieves high recognition rates while using a small subset of the available training data. While training data may be abundant, using all available data for training a classifier might not be feasible in certain cases, due to very large training or test durations. Instead, it would be more preferable, if a small subset of the training samples could be used to summarize most of the training samples within the classes. This would not only decrease training and test durations, but also reduce the space required to store the classifier; hence the name "memory conscious".

2. System Architecture

Our system architecture is illustrated in Fig. 1. In the preprocessing step, each symbol is scale normalized and subsampled. For feature extraction, we use gradient histogram features. The prototype selection step during training reduces the number of training samples used to train a classifier. During recognition, this first stage classifier is used for narrowing down the candidate classes and a single template for each candidate class is selected from the previously chosen prototypes, to be used in the second classifier. The final decision is obtained by combining the two classifiers at score level.

2.1 Preprocessing

The two main goals of the preprocessing step is to normalize the symbols so as to improve matching and to reduce the number of points in the shape so as to increase recognition speed. The reduced number of points improves speed both during feature extraction and second stage classification.

The preprocessing starts with scale normalization where the input symbol is rescaled while preserving the aspect ratio. Then, the points in the shape are resampled such that only points of high curvature are retained. Preprocessing applied to input symbols is illustrated in Fig. 2. At the top row, the input shapes are

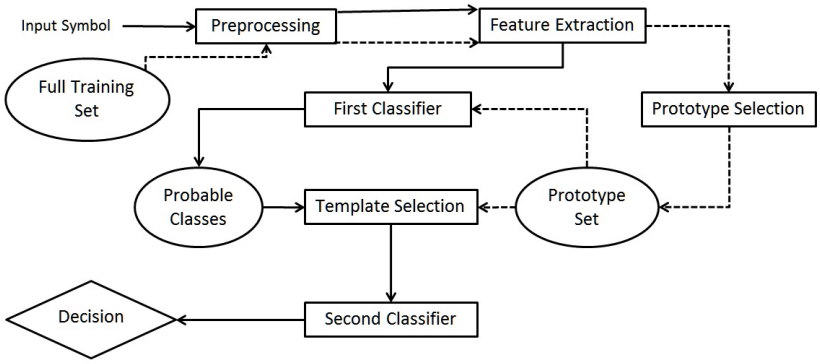
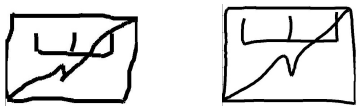


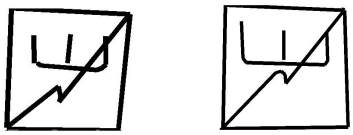
Figure 1. Online sketched symbol recognition system. The flow of data during training is shown by the dashed lines, whereas the data flow during recognition is shown by the solid lines.

displayed while the outputs are shown in Fig. 2b. In this case, the preprocessing step removes around 97% of the points from both shapes while successfully preserving the shape features.

In the two databases that we tested our system, COAD [1] and the NicIcon [8] databases, the preprocessing step, used at the same setting as in the experimental setup, removes 95.9% and 75.7% of the points on average, respectively.



(a) Inputs



(b) Outputs

Figure 2. Sample preprocessing result.

2.2 Feature Extraction

In order to represent sketched symbols, gradient histogram features are employed in out system. In order to extract these features, four orientation maps (horizontal, vertical and two diagonals) are calculated, where each map indicates a high response at locations in which the pen orientation coincides with the map orientation. Then, the the orientation maps are divided into 8×8 cells, using global elastic meshing of the symbol which are constructed through equally dividing the horizontal and vertical histogram of the symbol [4]. The elastic meshing is found to be a better strategy than fixed sized grids, especially when there are large variations among individual drawing styles [13]. The final feature vector for the input symbol is obtained by summing the elements inside each grid cell, separately for each orientation map. Each sum thus denotes the total response obtained at each cell, for a particular orientation and the result is a feature vector of size 256. The major advantage of this feature representation is that it is independent of stroke direction and ordering.

2.3 Prototype Selection

The purpose of selecting prototypes for each class is two-folds. Firstly, prototype selection reduces the time and memory required during training of a classifier. Secondly, during testing, the prototypes are employed in template selection which plays an important role in our architecture.

One method for selecting prototypes for a class is to cluster the training instances and retain the instances that are closest to the cluster centers as prototypes. This method can be used to separate training instances according to writing styles and locating representative samples for different styles.

In this work, we use the k-means clustering algorithm to select K training instances from each class as prototypes. We refer to the selected samples as prototype set.

2.4 First Stage Classifier

The goal of this classifier is to select the top-P most probable classes, among all the C classes ($P \ll C$, set to 8 in this work). When the input sketch is obtained, it

is compared to the prototypes using a nearest neighbour (NN) classifier, and the closest P classes are selected. This effectively prunes the search space of the second classifier, with only a small loss of accuracy.

The pruning and the cascaded use of two classifiers allows the system to focus its efforts to increase the overall classification speed. Another advantage of this architecture is that, if the supervised classifier produces similarity scores for each class, then these scores can be used for classifier combination.

We produce similarity scores using the NN classifier during tests using an exponential score function:

$$S(t,i) = \exp\left(\frac{-D(t,i)^2}{d_{max}^2}\right) \quad (1)$$

where $D(t,i)$ demotes the Euclidean distance between the feature vectors of the test and the i th shape, while d_{max} is the distance of the test shape to its P th nearest neighbour.

2.5 Template Selection

The template selection step aims to choose a single template from the prototype sets, for each of the most probable P classes. It is done by choosing the nearest neighbour of the input symbol from among the prototypes of a class, using the same nearest neighbour algorithm as described above. Although this is a simple method for template selection, it is very fast and proved to produce satisfactory results during our experiments.

The selected templates are used in the second classifier that compares two symbols (input and the template) to produce a similarity score.

2.6 Second Classifier

Motivated by the work of Lakämper and Sobel [5], and as an extension to our previous work [10], we designed an alignment algorithm that is able to align two input sketched symbols, independent of stroke direction and ordering. The alignment process produces a score which indicates the level of similarity between the symbols. Although we are unable describe the alignment strategy in detail due to lack of space, we can briefly state that the alignment score is affected by the total number of correspondences and the local similarity between corresponding points.

2.7 Classifier Combination

In order to produce the final recognition result, similarity scores of the nearest neighbour classifier and the alignment scores are combined. The current combination uses a simple multiplicative combination, though more sophisticated methods are available. Since the nearest neighbour classifier is trained with global shape features and the alignment score depends mostly on local feature similarity, this combination of classifiers reduces the error rate significantly.

Top-P Accuracy	K samples	Aligner Accuracy	NN Accuracy	Combin. Accuracy
100%	5	97.64%	92.92%	98.64%
100%	**10**	**97.64%**	**93.40%**	**99.55%**

Table 1. Recognition results for the COAD database.

3 Experiments

We tested our system on two databases, the COAD and the NicIcon. Since they are not very large, the COAD and the NicIcon databases might not seem to be good candidates to test our architecture. However, these databases are public and constitute a good benchmark for sketch recognition applications. During all the tests, we set the remaining class count after pruning to 8, $(P = 8)$. We tested the system with various number of prototypes, K. The recognition results are presented in Tables 1 and 2.

The COAD database contains 620 samples from 20 symbols drawn by 8 users. In the literature, Tumen et.al. [11] report a recognition accuracy of 98% on this dataset. Our system achieves a better recognition accuracy of 99.55% using 10 prototypes per symbol (the row in bold) which is the best recognition accuracy achieved in the literature.

The NicIcon database contains 26163 symbols representing 14 classes collected from 32 individuals. For this database, the training set contains a dedicated portion of 9416 instances. In the literature, the best recognition rates for the NicIcon database are reported by Willems et.al.[12] as 99.2%, using an SVM classifier trained with all the training data and selecting the best features from a large feature set. They also report the recognition accuracy of a classifier using the dynamic time warping (DTW) algorithm [7], using *all* the training samples as prototypes, which in turn achieves 98.5% recognition accuracy. In our tests, while selecting 10 prototypes per symbol and using only a single template for alignment, our alignment algorithm obtained a recognition accuracy of 95.64%. Considering that there are more than 650 instances per class, the reduction in testing time and memory requirement is sig-

Top-P Accuracy	K samples	Aligner Accuracy	NN Accuracy	Combin. Accuracy
99.98%	**10**	**95.64%**	**94.70%**	**97,45%**
99.98%	20	95.97%	94.70%	97.45%

Table 2. Recognition results for the NicIcon database.

K samples	Memory Requirement	Training Time	Average Test Time
10	**80 MB**	**167 sec.**	**0.42 sec.**
20	122 MB	346 sec.	0.67 sec.

Table 3. Memory and timing requirements for the NicIcon database during training and average test time per instance during testing.

nificant, at the cost of a loss in accuracy as displayed in Table 3.

For the NicIcon database, another point worth mentioning is the reduction of error due to classifier combination. The reduction of errors when $K = 10, 20$ are about 58% and 63% respectively, compared to the best performing single classifier. So, the two classifiers are indeed good candidates for classifier combination.

4 Discussion and Future Research

In this paper, we presented an online sketch recognition architecture capable of achieving high recognition rates using a small set of training data. This kind of architecture is most useful when the number of training samples and the number classes are very large, such as the case for Chinese character recognition which we have recently started working on and was part of the motivation for this work. Indeed, our results show that the overall accuracies obtained in the tested databases surpass (the COAD) or approach the state-of-the-art results (the NicIcon).

There are some design choices that we made in order to make the training and recognition as fast as possible, but that might adversely effect the recognition accuracies. Firstly, K is constant for all classes, while in fact some symbols are more complicated than others and using more training samples from those classes might increase accuracy. Secondly, template selection uses global features to find the nearest neighbor to the test instance. A better approach might be to run the alignment algorithm for a small number of steps and choose the instance that obtains the highest score, as prototype. However, this approach would also be much slower than our current approach. Another possibility that might increase accuracies is to use more than one template for each symbol during recognition. This way the chance that a bad template is selected is lowered, at the cost of runtime speed.

References

[1] 101-5-1, Operational Terms and Graphics.*Washington, DC, Department of the Army 30*, 1997.

[2] F. Brieler and M. Minas. A model-based recognition engine for sketched diagrams. *Journal of Visual Languages and Computing*, 21:81–97, April 2010.

[3] T. Hammond and R. Davis. Ladder, a sketching language for user interface developers. *Computers and Graphics*, 29:518–532, August 2005.

[4] L. Jin and G. Wei. Handwritten chinese character recognition with directional decomposition cellular features. *Journal of Circuits, Systems, and Computers*, 8(4):517–524, 1998.

[5] R. Lakämper and M. Sobel. Using the particle filter approach to building partial correspondences between shapes. *International Journal of Computer Vision*, 88(1):1–23, 2010.

[6] A. C. Long, Jr., J. A. Landay, L. A. Rowe, and J. Michiels. Visual similarity of pen gestures. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '00, pages 360–367, New York, NY, USA, 2000. ACM.

[7] R. Niels, L. Vuurpijl, and L. Schomaker. Automatic allograph matching in forensic writer identification. *International Journal of Pattern Recognition and Artificial Intelligence*, 21:61–81, 2007.

[8] R. Niels, D. Willems, and L. Vuurpijl. The NicIcon database of handwritten icons. In *11th International Conference on the Frontiers of Handwriting Recognition*, Montreal, Canada, August 2008.

[9] M. Oltmans. *Envisioning sketch recognition: a local feature based approach to recognizing informal sketches*. PhD thesis, Cambridge, MA, USA, 2007.

[10] C. Tirkaz, T. M. Sezgin, and B. Yanikoglu. Sketched symbol recognition with few examples using particle filtering. In *ACM International Symposium on Sketch-Based Interfaces and Modeling*, Vancouver, Canada, 2011.

[11] R. S. Tumen, M. E. Acer, and T. M. Sezgin. Feature extraction and classifier combination for image-based sketch recognition. *ACM Symposium on Sketch Based Interfaces and Modeling*, June 7-10, (2010).

[12] D. Willems, R. Niels, M. van Gerven, and L. Vuurpijl. Iconic and multi-stroke gesture recognition. *Pattern Recognition*, 42(12):3303 – 3312, 2009.

[13] Z. Zhang, L. Jin, K. Ding, and X. Gao. Character-sift: A novel feature for offline handwritten chinese character recognition. In *10th International Conference on Document Analysis and Recognition*, pages 763 –767, july 2009.