MEMO #93–06
June 1993

Recognition of Handprinted Digits[1] using Optimal Bounded Error Matching

Thomas M. Breuel
IDIAP, C.P. 609, 1920 Martigny, Switzerland
tmb@idiap.ch

ABSTRACT

This paper describes a system that recognizes hand-printed digits. The system is based on optimal bounded error matching, a technique already in common use in general-purpose 2D and 3D visual object recognition systems in cluttered, noisy scenes. In this paper, we demonstrate that the same techniques achieve high recognition rates (up to 99.2%) on real-world data (the NIST database of hand-printed census forms and the CEDAR database of digits extracted from U.S. mail ZIP codes).

As part of the system, we describe a post-processing step for k-nearest neighbor classifiers based on decision trees that can be used (in place of the usual heuristic methods) for setting thresholds and improves recognition rates significantly.

Keywords: ZIP codes, digit boundary, unconstrained handwritten digit recognition, pre-segmented handwritten digit recognition, nearest neighbor classification, decision trees, classification trees.

[1] A version of this paper has appeared in ICDAR '93 (Breuel, 1993).

1 Introduction

The off-line recognition of handwritten text has been the subject of intense research for many years (for recent reviews, see Chatterji, 1986, Davis and Lyall, 1986, Govindan and Shivaprasad, 1990, Impedovo *et al.*, 1991, R. Allen Wilkinson *et al.*, 1992). An important subproblem is the problem of recognizing handprinted digits. This is not only because there exist a significant number of practical applications for a system that can reliably recognize (strings of) handwritten digits, but also because because it is an important test case for more general recognition problems.

In order to achieve high recognition rates and high throughput, many systems for recognizing handwritten digits have made strong assumptions about the input data. Examples of such assumptions are that the input consists of fully segmented characters and that each character consists only of a single connected region. Such assumptions allow the use of approaches such as global features (moments, topological features, Fourier descriptors) that are not well suited to general visual object recognition problems because of the presence of clutter, occlusions, and noise in more realistic images. In addition, such an approach also makes the resulting systems be not very robust in the presence of violations of the basic assumptions. This can present a serious practical problem, since, for example, segmentation may well be the step that is limiting overall recognition rates, or since cluttered, degraded, or low-quality documents, or manually corrected characters may cause such systems to fail.

In this paper, I present the results of applying a general-purpose vision system to the problem of handwritten digit recognition. The vision system used in this work (Breuel, 1992b) has already been successfully applied to the recognition of 3D objects, and has shown to be able to cope with occlusions, clutter, and considerable noise and variation (often induced in 3D scenes by changes of illumination and shadowing). The vision system relies only on very simple, low-level features. Basically, objects are represented by their edges or boundaries (these can, of course, be curved), and recognition is achieved by identifying the model view for which the largest fraction of its boundary can be brought into correspondence with boundaries in the image, subject to given error bounds. Essentially the same system is being used for the character recognition experiments described here.

Our goal in this paper is to show that one can achieve excellent recognition rates with a general purpose recongition system, without having to rely on domain specific features or representations.

We ultimately hope to be able to take advantage of the ability of such systems to cope with clutter and occlusion in order to improve recognition in complex or degraded documents. On the other hand, character recognition (in context) is a good task for assessing quantitatively and improving the ability of general purpose recognition systems to cope with multiple objects and clutter.

2 System Overview

2.1 Matching

Small amounts of noise were removed from the raw character images using morphological processing. A bounding box was calculated. The image was resampled uniformly in x and y such that the bounding box fit into a 48×48 box. After resampling, the image was convolved with a Gaussian of width 2.0. The Laplacian of the resulting smoothed image was computed. The zero crossings of the Laplacian were approximated using a polygonal chain of segments of fixed length (2 pixels). These chain segments were used as input to the matching algorithm.

Note that no slant normalization or thinning was carried out. Inspection of the database suggested that character identity was not complete invariant under either kind of transformation. For example, certain types of digits "7" and "1" seem distinguishable mainly by a slightly larger slant of the digit "7". Not correcting for slant avoids this problem (another approach would have been to carry out slant correction and make the amount of the correction available to the classifier). Similarly, a thinning step eliminates all information relating to stroke width. But certain pairs of digits, for example, a digit "6" with a very small, filled loop at the bottom and a slightly curved digit "1" are indistiguishable after thinning. The boundary representation used in the system avoids this problem.

The matching method used by the system is that of optimal bounded error matching. Optimal bounded error matching has been used extensively in 2D and 3D recognition systems (see Grimson, 1990 for an extensive bibliography). The optimal bounded error matching problem can be solved by a variety of algorithms. One of the algorithms that is best suited when matching involves a large number of simple features is the recently developed RAST algorithm (Breuel, 1992a), which was used in the system described here.

To illustrate the approach, let us formalize the optimal bounded error matching problem. Let a *feature* be a point in $\mathbb{R}^2$ with associated information such as orientation (we call this associated information the feature *label*). We say that two features are *compatible* if their associated information satisfies some prespecified constraints.

Given a set of model features $\{m_1, \ldots, m_k\}$, a set of image features $\{b_1, \ldots, b_l\}$, and a set of error bounds $\{\epsilon_1, \ldots, \epsilon_l\}$, the RAST algorithm finds a translation T such that a maximal number of compatible image and model features are brought into correspondence under the given error bounds:

$$\|T\, m_{\alpha_i} - b_{\beta_i}\| \leq \epsilon_{\beta_i} \qquad (1)$$

Here, α and β are 1-1 functions from $\mathbb{N}^r$ to $\mathbb{N}^k$ and $\mathbb{N}^l$, respectively, where r is the maximal number of features that can be brought into correspondence. Note that the RAST algorithm can also be used to find optimal transformations in more general cases, such as matching under equiform transformations or 3D transformations.

The recognition system used as features the locations and associated orientations of the segments of the polygonal approximation returned by the Laplacian edge extractor. The collection of these features is therefore a *representation of the boundary* (or edges) of the input character, augmented by additional information. Two such features were considered compatible if their orientation was within 23 degrees of one another. The error bounds were chosen to be 5 pixels. Note that none of the system parameters have yet been optimized by exploring the parameter space. Some improvement in recognition rate from choosing better parameter values is to be expected.

An example of an optimal bounded error match is shown in Figure 1.

Each match was assigned a *quality of match*, which was the minimum of the fraction of the model matched by the image and the fraction of the image matched by the model. Note that this quality of match measure tends to degrade gracefully as more of the instance of a model in an image becomes occluded or as spurious features are added to the image.

2.2 Classification

Classification was carried out in two stages. The first stage was a k-nearest neighbor classifier. The k outputs from the k-nearest neighbor classifier were then used as input to a decision tree algorithm, which made the final decision about the identity of the input character.

The database for the k-nearest neighbor classifier was constructed by "compression". That is, for the construction of the database, the algorithm started with an empty database and sequentially considered examples from a training set. If a character was classified correctly using the examples in the database built so far, no action was taken. If the character was misclassified, it was added as a new prototype to the database. The database of models was constructed entirely before the second stage classifier, the tree classifier, was built; the classification that was used during the construction of the database was simply nearest neighbor classification. Nearest neighbor classification tends to have an error rate that is about 2–4 times as large as that of the tree based classifier. This means that the database is probably somewhat larger than it needs to be. The number of characters in the database is shown in column "#proto." in Table 1.

For the experiments, k was chosen to be 10. That is, for each character, the 10 nearest neighbors (under the quality of match measure described above) from the database were output. Among these, the two most frequent classifications were determined, and for each of the classifications, the number of matches corresponding to that classification, and the minimum, maximum, and average quality of match was determined.

This information was used to build a decision tree (Breiman et al., 1984) that could make one of three choices: (1) the input character belongs to most frequent classification (among the k nearest neighbors), (2) the input character belongs to the second most frequent classification, or (3) the character belongs neither to the first or the second most frequent classification. When the output of the decision tree was (3), then the input character could either be rejected, or it could be assigned arbitrarily to the most frequent vote. Relatively few characters were in this third class, and, for simplicity, for the experiments described in this paper, we therefore arbitrarily assigned characters in class (3) to class (1) (resulting in somewhat higher error rates), rather than reporting separate rejection rates.

Both the k-nearest neighbor classification and the decision tree were cross-validated. The nearest neighbor classifier was cross-validated by simply not allowing a character to match against itself in the database (the "leave one out" technique). For the decision tree, five-fold cross validation was used (for more details on the cross validation procedure, the reader is referred to the book by Breiman et al., 1984). This allows us to give statistically valid performance estimates of the recognizer using the whole training set, rather than explicitly setting aside a certain part as "testing data". However, as a "sanity check", the data was also split manually into training and test data in several cases.

2.3 Database

For training and testing the system, the NIST (National Institute of Standards and Technology, Garris and Wilkinson, 1992) and CEDAR (Center for Excellence in Document Analysis and Recognition, CEDAR, 1992) databases were used. Both databases contain presegmented, preclassified alphanumeric, handprinted characters. In both cases, there were no constraints on the writing instrument or writing style. However, the writers who provided the data for the NIST database did so specifically for the purpose of building a database for testing systems for the recognition of handprinted characters. The CEDAR database, in contrast, was collected from actual envelopes processed by the US post office. For the work reported here, only the presegmented digits contained in these databases were used.

The NIST database used the same acquisition, segmentation, and classification procedure for the whole dataset (a separate set of test data collected from a different writer population has recently also become available). The NIST database contains a total of 223125 digits representing 2100 writers. For the experiments reported here, an arbitrarily selected subset of 20790 digits from the database representing 2079 writers was used, one digit of each class per writer.

It is important to note that as a consequence of this selection, the cross validation (of the nearest neighbor classifier and the tree classifier) also took place across writers. That is, during the cross validation, digits written by one writer were never classified based on information derived from digits of the same class written by the same writer.

The CEDAR database consisted of three parts. These parts were acquired, segmented, and/or classified differently from one another at CEDAR: a training set of 18468 characters from 4000 address blocks, and a test set of 2711 characters, acquired and processed at a later date. The test set was present in two copies. The first copy (henceforth referred to as "CTEST") was obtained by automatic segmentation of ZIP codes. The second copy (henceforth referred to as "CGTEST"), consisting of 2213 digits, was obtained by manually removing ambiguous, ill segmented, or misclassified digits. For further details of the segmentation procedure and the manual preparation and classification of the data, the reader is referred to the database documentation.

In reporting results (Table 1), characters in the range 0–20790 refer to characters from the NIST database, while characters in the range of 20790–35500 refer to characters from the CEDAR database (all ranges are exclusive at the top).

3 Results

System recognition rates. The results of applying the recognition system to the NIST and CEDAR databases of handprinted digits are shown in Table 1. The "parts used" columns in the table show what parts of the database were used to build the database of prototypes for the k-nearest neighbor algorithm, for the training of the tree-based classifier, and for testing. Note again (see above) that even in the cases in which the "testing" set and the "knn" or "tree" sets overlap, the recognition rates are cross validated and constitute a good estimate of the performance of the recognition system on new data.

In experiments E1 and E2, training and testing was carried out on parts of the same database, i.e., on characters that were acquired and segmented under identical conditions. In experiments E3 and E4, training was carried out on a database that consisted to approximately 60% of digits from the NIST database and to approximately 40% of digits from the CEDAR database training set. These sets of digits are generally well segmented and classified (probably less than 0.5% serious segmentation errors or misclassifications for the NIST database). Furthermore, testing was carried out on sets of digits that were acquired and processed under conditions identical to those for large numbers of digits in the training set. This is reflected in the high rate of correct classification of 98.4%–99.2%.

In both cases, performance was slightly worse when the testing set was disjoint from the set of characters used for building the database and the decision tree, but this does not appear to be due to a failure of cross validation. Rather, in the case of E3 vs. E4, this is probably due to the different amounts of training data used for building the tree (10000 examples in the case of E3 vs. 20500 samples in the case of E4). In the case of E1 vs. E2, separate tests suggest that the digits numbered 18790–20790 are somewhat more difficult to recognize than the digits numbered 10000–18790.

On the CGTEST set, the system achieved a recognition rate of 98.3%. Many of these errors seem to be due to characters with very unusual aspect ratios or slants. On the CTEST set, the recognition rate was significantly lower: 97.0%. Not surprisingly, since the input was segmented automatically, on examination of the misclassified characters, we find that this is to a significant degree due to poorly segmented input characters (often resulting in unrecognizable or ambiguous shapes).

The fact that the test set CTEST has a significantly larger proportion of ill-formed characters is likely responsible for the difference in recognition rates between E7 and E8. We can view the tree-based post-processing of the output of the k-nearest neighbor classifier as an automatic way to set "thresholds", but if the training set does not contain enough malformed characters, a statistically valid decision tree cannot set these thresholds accurately.

From Table 1, we can see that in 1.6% of the cases, neither the first nor the second choice of the k-nearest neighbor classifier was correct. Because of this, class (3) of the output of the tree classifier had a significant size. If we allow the system to reject characters, the overall performance for E7 is the following: 96.9% correct, 2.3% error, 0.8%, rejected.

Of these 0.8% of rejected digits, 0.7% are digits that actually need to be rejected (since they are neither the first nor the second choice of the output of the k-nearest neighbor algorithm), and 0.1% are errorneously rejected first or second choices.

Tree-based decision rules. It is interesting to ask how important the effect of the two classification methods (k-nearest neighbor, tree-based decision rules) is on the overall performance of the system. We can see this by examining the columns "majority" and "first or second" in Table 1. The column "majority" simply gives the performance of a k-nearest neighbor classifier for $k = 10$, while the column "first or second" describes the theoretical upper limit for the performance of the tree-based classifier. We can see from this that an increase in the recognition rate of around 0.5% is average. In several cases, the recognition rate was increased by almost 1% (E2, E4, and E7). In the case of E2, this resulted in cutting the error rate in half.

Examples of Errors Figure 2 shows the complete set of characters for which neither the first nor the second choice of the k-nearest neighbor classifier was correct in experiment E1. From this, we can see that several errors were due to malformed digits or segmentation errors (7, 9, 10, 11). One error was due to a misclassified digit in the database (4). Another error was due to an unusual slant (13). Two errors were due to qualitative variations in style that were apparently underrepresented in the training set (the "barred 1", examples 2 and 5). The remaining five digits, while perhaps still classifiable by a human reader, involve rather subtle distinctions in character shape. For example, in the case of digit number 8, if the intersection between the two strokes occurred just a little bit higher, the digit would be classified as a "7" by a human reader as well.

4 Discussion

Let us try to relate the approach to character recognition described in this paper to other existing approaches.

Template matching is one of the most commonly used recognition algorithms. And, indeed, the method described here is quite similar to template matching: as in template matching, the input character is compared to a number of prototypes ("templates"), and the best matches are used to classify the input.

The crucial difference between template matching and the method described in this paper lies in the representation of characters and the similarity measure used. Template matching commonly uses the correlation (or matched filtering) of the input and prototype (possibly after some signal processing). If we want to achieve larger error bounds, we have to dilate the input character. This means that character shape and the effects of shape variation can be related in complicated ways.

Optimal bounded error matching using boundary representations, on the other hand, lets us set error bounds directly (in fact, we can choose different error bounds for different parts of a model or image). Furthermore, boundary representations let us express, through the use of labels, important constraints such that (for example) approximately vertical and/or convex parts of a model boundary should only match approximately vertical and/or convex parts of an image boundary.

At the other end of the spectrum, we find *feature based* methods[2] Such methods rely on the extraction of "characteristic properties" of characters. Examples of features are global topological properties, the presense and absence of a variety of local configurations of lines (e.g., junctions and terminators), moments, Fourier boundary descriptors, and various projections and crossings (Chatterji, 1986, contains an extensive review of different kinds of features used for character recognition).

The driving principle behind feature-based recognition methods is data reduction. That is, variation that is inessential to the identity of a character is to be discarded during the feature extraction process. Unfortunately, it seems that *all* the input data describing a character can be essential for determining its identity and no information should be discarded before the classification process.

We can view the boundary representations used in this work as a way to perserve a complete representation of character shape, which is additionally enriched and annotated by additional information that captures many of the same shape properties that are used by feature based representations. As such, it follows the design principles suggested by Marr, 1982.

Another important aspect of the system is the use of an automatically generated decision tree for post-processing of the output of the k-nearest neighbor classifier. This provides an automated alternative to manual and/or heuristic choices of thresholds and voting strategies and seems to result in noticeably better performance. We expect that this

[2]The "features" of "feature based methods" usually represent larger scale characteristic properties of characters and are to be distinguished from the "features" used in representing boundaries representations in this system, which are local and individually carry virtually no information about character identity. This is an unfortunate convergence of terminology.

technique will be useful for other systems based on k-nearest neighbor classification as well. Results with the decision tree learning algorithm also suggest that using training data of higher quality than testing data may not be an optimal choice. While manually verified training data may constitute a good data set to choose character prototypes from, rejection thresholds should be set on data with the same fraction of "bad" characters as the testing data (and the data in the application of the system, of course).

As we already noted before, another important aspect of the approach described here is that boundary representations and bounded error matching have explicitly been designed to cope with clutter and occlusions. Note that little can be inferred about the degree to which the current system can deal with clutter and occlusion from the performance of the system on the CTEST database, which actually does contain a significant number of poorly segmented characters. The reasons are that (1) the segmentation algorithm, when it fails, often introduces much graver mutilation than are to be expected from other degradations of the input, and (2) the majority of characters (even in the CTEST database) contain no clutter or occlusions, and the system chooses prototypes and sets parameters for the undegraded and uncluttered case. This aspect of the system needs to be explored experimentally in future work, using training data that contains contextual information.

There have been a few attempts, similar in spirit to the one described in this paper, at applying more general representations of shape together with matching algorithms from computer vision to the character recognition problem. Edelman *et al.*, 1990, use a variation on the alignment method (Huttenlocher and Ullman, 1987) for the problem of recognizing cursive handwriting. Recognition by alignment is very similar to the bounded error recognition method described here, but it requires the extraction of alignment points. Mitchell and Gillies, 1989, are concerned with the recognition of complete ZIP codes from given address blocks; the digit recognition component of their system is "painstakingly crafted using an iterative refine and test methodology". Both systems show the promise that the use of more general techniques from computer vision hold for the recognition of handwriting, but, unlike the work presented here, they have not yet demonstrated competitive performance on standard recognition tasks.

The above results demonstrate that optimal bounded error recognition using labeled boundary features can achieve excellent recognition rates for handwritten digit recognition. This constitutes a first step towards building a high recognition rate recognizer for the considerably harder problem of off-line recognition of unrestricted, unsegmented handprinted characters.

References

Breiman et al. L., 1984, *Classification and Regression Trees*, The Wadsworth statistics/probability series, Wadsworth, Belmont, CA.

Breuel T. M., 1992a, Fast Recognition using Adaptive Subdivisions of Transformation Space, In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition.*

Breuel T. M., 1992b, View-Based Recognition, In *MVA '92, IAPR Workshop on Machine Vision Applications.*

CEDAR , 1992, CEDAR CDROM 1: USPS Office of Advanced Technology Database of Handwritten Cities, States, ZIP Codes, Digits, and Alphabetic Characters, 5.25" CD-ROM with documentation, available from: Jonathan J. Hull, Associate Director, CEDAR, 226 Bell Hall, State University of New York at Buffalo, Buffalo, NY 14260-000, `hull@cs.buffalo.edu`.

Chatterji B., 1986, Feature extraction methods for character recognition, *IETE Technical Review*, vol.3, no.1:9–22.

Davis R., Lyall J., 1986, Recognition of handwritten characters - a review, *Image and Vision Computing*, vol.4, no.4:208–18.

Edelman S., Ullman S., Flash T., 1990, Reading cursive handwriting by alignment of letter prototypes, *International Journal of Computer Vision*, 5:303–331.

Garris M. D., Wilkinson R. A., 1992, NIST Special Database 3: Binary Images of Handwritten Segmented Characters, 5.25" CD-ROM with documentation, available from: Standard Reference Data, national Institute of Standards and Technology, 221/A323, Gaithersburg, MD 20899.

Govindan V., Shivaprasad A., 1990, Character recognition-a review, *Pattern Recognition*, vol.23, no.7:671–83.

Grimson E., 1990, *Object Recognition by Computer*, MIT Press, Cambridge, MA.

Huttenlocher D. P., Ullman S., 1987, Object Recognition Using Alignment, In *Proceedings of the International Conference on Computer Vision*, pages 102–111, London, England, IEEE, Washington, DC.

Impedovo S., Ottaviano L., Occhinegro S., 1991, Optical character recognition-a survey, *International Journal of Pattern Recognition and Artificial Intelligence*, vol.5, no.1-2:1–24.

Marr D., 1982, *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*, W.H. Freeman and Company, San Francisco.

Mitchell B., Gillies A., 1989, A model-based computer vision system for recognizing handwritten ZIP codes, *Machine Vision and Applications*, vol.2, no.4:231–43.

R. Allen Wilkinson *et al.* , ed., 1992, *The First Census Optical Character Recognition System Conference*, U.S. Department of Commerce, National Institute of Standards.

	parts used			correct/error			
	knn	tree	testing	system	majority	1st or 2nd	#proto.
E1	0–18790	10000–18790	18790–20790	98.6/1.4	98.2/1.8	99.4/0.6	612
E2	0–20790	10000–20790	10000–20790	99.1/0.9	98.2/1.8	99.6/0.4	664
E3	0–30000	20000–30000	30000–35500	98.4/1.6	98.3/1.7	99.7/0.3	910
E4	0–35500	15000–35500	30000–35500	99.2/0.8	98.3/1.7	99.7/0.3	1038
E5	0–35500	CGTEST	CGTEST	98.2/1.8	97.9/2.1	99.5/0.5	1038
E6	0–35500	15000–35500	CGTEST	98.3/1.7	97.9/2.1	99.5/0.5	1038
E7	0–35500	CTEST	CTEST	97.0/3.0	96.1/3.9	98.4/1.6	1038
E8	0–35500	15000–35500	CTEST	96.7/3.3	96.1/3.9	98.4/1.6	1038

Table 1: Recognition rates.

```
if first-votes <5.5
  if second-max <8017.5
    decide 1 (889/950)
  if second-max >8017.5
    if first-max <8326
      decide 2 (76/88)
    if first-max >8326
      if second-avg <8349.12
        decide 1 (82/83)
      if second-avg >8349.12
        if first-max <8866.5
          decide 2 (7/7)
        if first-max >8866.5
          decide 1 (4/5)
if first-votes >5.5
  decide 1 (7642/7657)
```

Table 2: Example of the kinds of decision rules generated by the CART algorithm.

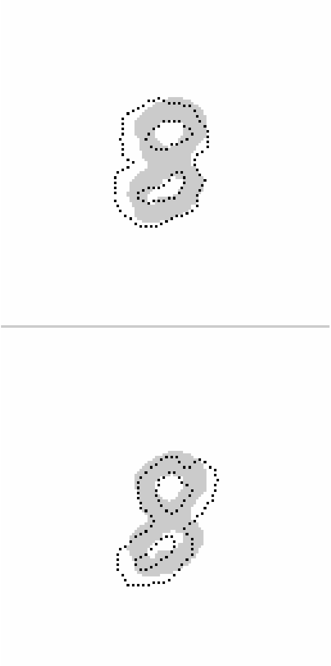

Figure 1: An example of the boundary representation of the top two matches of prototypes (dotted outline) against an input character (grey).

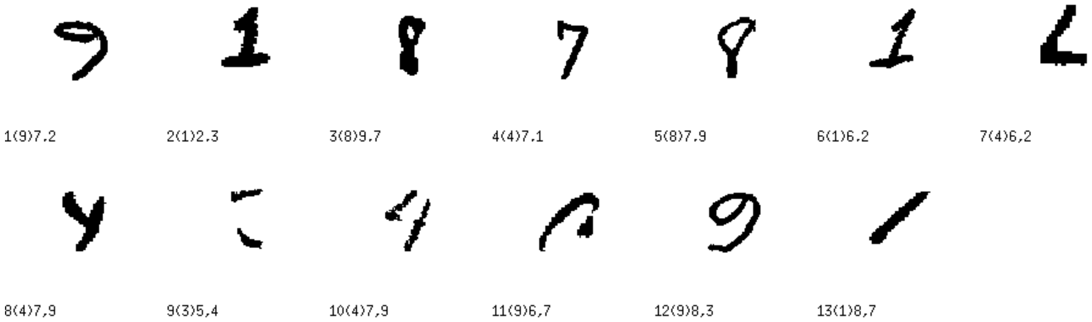

Figure 2: All characters in E1 that were misclassified by both the first and the second choice. The format of the annotation under each character is "*number*(*correct-classfication*)*first-choice*,*second-choice*".

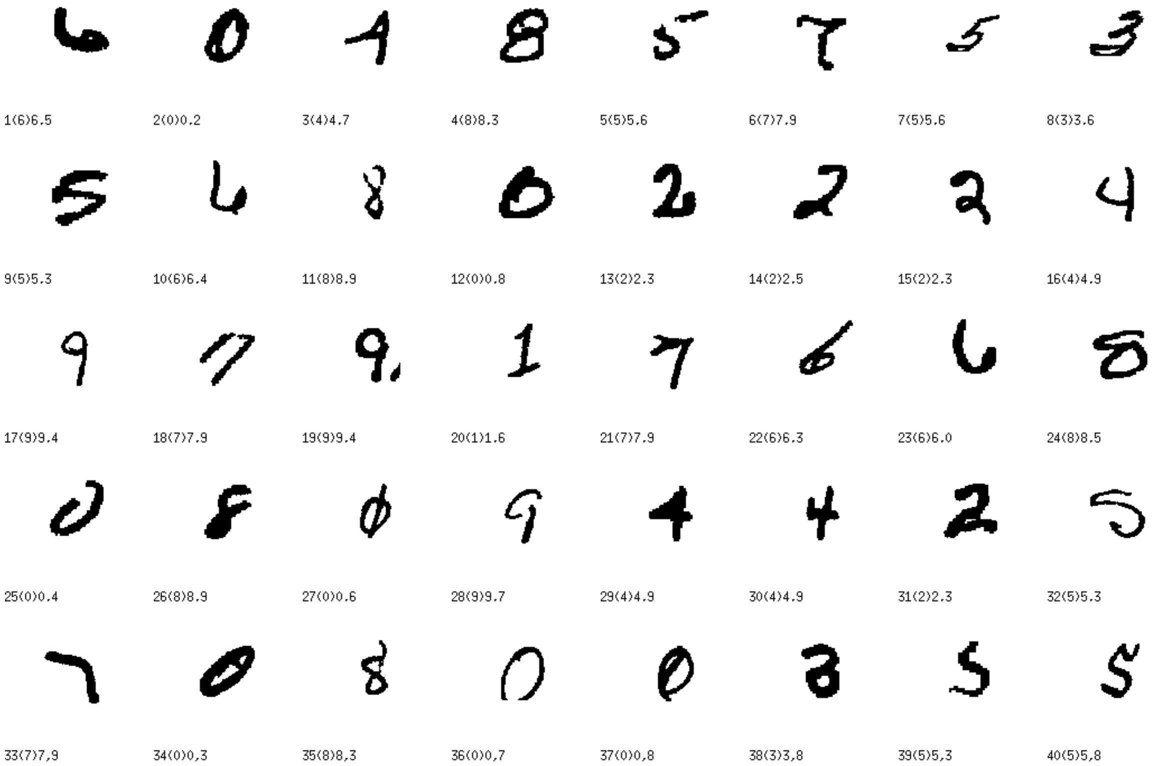

Figure 3: Some characters in E1 that were classified correctly by the k-NN method as the first choice. The format of the annotation under each character is the same as in Figure 2.