Research Feature

# Neural and Fuzzy Methods in Handwriting Recognition

Handwriting recognition has challenged computer scientists for years. To succeed, a computing solution must ably recognize complex character patterns and represent imprecise, commonsense knowledge about the general appearance of characters, words, and phrases.

*Paul D. Gader*
*James M. Keller*
*Raghu Krishnapuram*
*Jung-Hsien Chiang*
*Magdi A. Mohamed*
University of Missouri

Character recognition is a classical computing problem, dating back to neural computing's infancy. One of Frank Rosenblatt's first demonstrations on the Mark I Perceptron neurocomputer in the late 1950s involved character recognition.[1] The Perceptron was one of the first computers based on the idea of a neural network, which is a simplified computational model of neurons in a human brain. It was the first functioning neurocomputer, and it was able to recognize a fixed-font character set. As with many artificial intelligence applications, the difficulty of handwriting recognition was greatly underestimated. Significant progress was not achieved until the late 1980s and early 1990s, when many technologies converged to enable rapid increases in recognition rates for digits, characters, and words so that reliable commercial systems could be developed.

Handwriting recognition problems are either online or offline. Online recognition systems use a pressure-sensitive pad that records the pen's pressure and velocity, which would be the case with, for example, a personal digital assistant. In offline recognition, the kind we are concerned with here, system input is a digital image of handwritten letters and numbers.

Handwriting recognition requires tools and techniques that recognize complex character patterns and represent imprecise, commonsense knowledge about the general appearance of characters, words, and phrases. Neural networks and fuzzy logic are complementary tools for solving such problems. Neural networks, which are highly nonlinear and highly interconnected for processing imprecise information, can finely approximate complicated decision boundaries. Fuzzy set methods can represent degrees of truth or belonging. Fuzzy logic, one of several fuzzy set methods, encodes imprecise knowledge and naturally maintains multiple hypotheses that result from the uncertainty and vagueness inherent in real problems. By combining the complementary strengths of neural and fuzzy approaches into a hybrid system, we can attain increased recognition capability for solving handwriting recognition problems.

This article describes the application of neural and fuzzy methods to three problems:

- recognition of handwritten words,
- recognition of numeric fields, and
- location of handwritten street numbers in address images.

These problems and methods were part of research we conducted on US Postal Service data and on problems of interest to the USPS.

## HANDWRITTEN WORD RECOGNITION

Handwritten word recognition involves matching a digital image of a handwritten word to a lexicon. Successful recognition, however, cannot rely on simply isolating and recognizing individual characters: To arrive at the best interpretation, recognition techniques must also evaluate information about the context of the word.

Even examined character by character, handwriting is ambiguous, as Figure 1 shows. Many handwriting recognition methods segment an image into subimages before applying recognition algorithms.

Figure 2 shows how segmentation can cause additional ambiguities. Here, the word "Cowlesville" is incorrectly segmented into subimages, several of which are plausible characters when viewed in iso-

0018-9162/97/$10.00 © 1997 IEEE

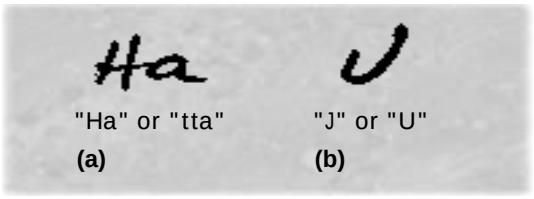


*Figure 1. Ambiguities in handwritten word recognition. (a) A single pixel pattern can represent single or multiple characters. (b) A single pixel pattern can represent different characters.*

lation. A word recognition system could therefore assign a high score to the word "Avenue" for this image. Moreover, isolated character recognition cannot take into account information regarding spatial and lexical compatibility of adjacent character images. Even though the fifth and sixth segments in Figure 2b look like "u" and "e" when taken individually, humans would probably not think so because the "u" is much larger than the "e." Conversely, a correctly segmented image can contain legitimate characters that do not look legitimate. (See if you can read the word in Figure 2c; the answer appears in Figure 6.)

### Word recognition solution

We have written a computer program that recognizes handwritten words. The inputs to the program are a digital image of a word and a lexicon, which is a list of strings representing the possible identities of the word. The program structure is shown in Figure 3. The program segments[2] the input word "Richmond" into 25 primitives and applies a matching process that finds the best way to assemble the primitives. Each assembly of primitives is a segmentation to match each string in the lexicon. As Figure 3 shows, the program assigns a confidence value between 0 and 100 to each segment that corresponds to a match.

The confidence value, which represents the degree to which an image segment looks like a character, must accurately reflect the ambiguities between characters like "J" and "U." Furthermore, segments that represent noncharacter images, such as the first subimage in the segmentation corresponding to "Edmund," should be assigned low confidence values to avoid false interpretations. Finally, these confidence values must be combined to produce an overall match score. The program uses the score to rank the strings in the lexicon according to how much they match the image. The top-ranked string is the recognition result.

### Neural and fuzzy techniques

We experimented with a variety of novel techniques for applying neural and fuzzy methods to the handwriting recognition problem. They are as follows:

- *We assigned character confidence values* to individual segments using fuzzy sets and multilayer feed-forward neural networks (MLFNs).
- *We gave the segments contextual information* by assigning a confidence that character pairs were spatially compatible, again using MLFNs.
- *We aggregated character confidence values* from a segmentation, by means of fuzzy integrals.

### Assigning character confidence

Character confidence values can be assigned using either a fuzzy or crisp neural network. A crisp set is an ordinary set, while a fuzzy set is a set for which set membership can be any number between 0 and 1. A crisp net determines whether a person is tall or not tall, for example; a fuzzy net determines that a person is tall to a certain degree. Fuzzy sets naturally represent pixel patterns that can have degrees of membership in different character classes. Character-class membership values correspond to character confidence values, if we can approximate the membership

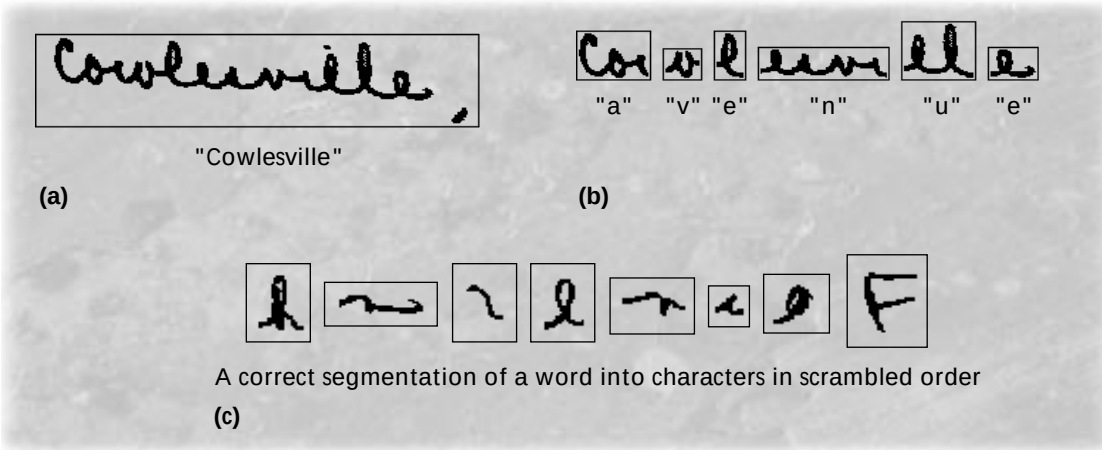


*Figure 2. Ambiguities caused by segmentation. (a) A word that is (b) incorrectly segmented can lead to a bad match, even when the subimages are plausible characters. (c) A correctly segmented word but with characters in scrambled order to illustrate that no context is available.*

function. To do this, we used the fuzzy *k*-nearest neighbor (*k*-nn) algorithm to assign character-class membership values, which were then used as the training set for an MLFN neural net.[3,4] MLFNs provide powerful function approximation and thus are excellent for character recognition.

To compute the character-class membership values, we evaluated the number of neighbors of each training pattern (in feature space—a set of feature vectors or ordered sets of measurements made on a pattern) from the character class. The MLFNs were trained to mimic the examples and interpolate between them.

We train MLFNs using *supervised learning*: learning by using specific examples. We assign each training pattern a desired output for each output unit (usually one unit per character class). The standard training approach assigns a high value (usually a 1) to the desired output and a low value (usually a 0) to all other outputs. The standard approach is appropriate for generating a crisp set but not to approximate fuzzy sets.

For example, the standard approach would have assigned a high value to the "U" output node for the "U" in Figure 1b. This could complicate training if there were a "J" in the training set that looked very similar to the "U." The MLFN would then try to solve the almost impossible task of producing two very different outputs for two almost identical inputs. The network might have resolved this conflict by producing the desired value for only one of the inputs.

We thus trained MLFNs to approximate fuzzy sets instead of crisp sets. This approach assigns fuzzy desired outputs on both the "U" and the "J" nodes for each of two training patterns. The fuzzy desired outputs corresponded to how well the given pattern represented both a "U" and a "J." As a result, the MLFN learned and represented the training data more realistically by incorporating the uncertainty associated with the patterns. This technique can be applied to any network that uses class-coded supervised training.

We compared networks trained with standard and supervised approaches on US Postal Service data sets. For isolated character recognition, the crisp networks achieved superior character recognition rates: 89.2 percent for the crisp networks and 83.8 percent for the fuzzy networks. However, the fuzzy networks achieved word recognition rates of 75.5 percent, compared to 70.1 percent achieved with crisp networks.[3,4] We can interpret this result as an affirmation of the Principle of Least Commitment, stated by David Marr for the design of intelligent computer-vision algorithms[5]: *Don't do something that may later have to be undone.* In the case of handwritten word recognition, if we use crisp class memberships, then the MLFN makes irreversible decisions concerning the class membership of an input segment. If the identity is not correct, the mistake will lead to an erroneous output. Fuzzy class memberships are not definite; the network represents multiple class memberships and can more accurately make the final decision at a later stage of processing.

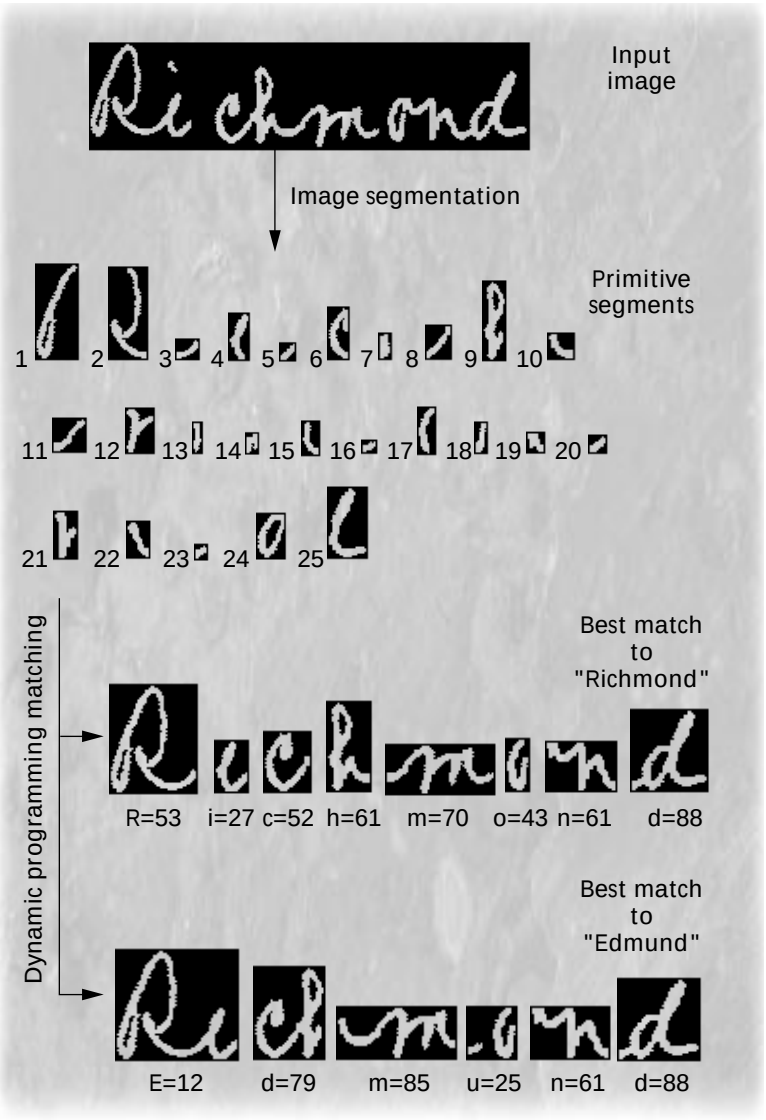


***Figure 3. Segmentation approach to handwriting recognition. We segment the input word "Richmond" into primitives. Then a matching process finds the best way, via dynamic programming (an optimization algorithm), to assemble primitives to match the string with a word in the lexicon. The best matches to the words "Richmond" and "Edmund" are shown. Each subimage of a segmentation corresponding to a match is assigned a scaled confidence value between 0 and 100.***

## Assuring spatial compatibility

As Figure 2 illustrates, individual segments can look like characters even though they aren't. However, when adjacent segments having this property are viewed together, differences in size and spatial location can reduce the likelihood that they might be char-

To find an optimal match between a word in the lexicon and segmentations of the word, we used dynamic programming.

acters. For example, the fifth and sixth segments in Figure 2b should receive high character-class membership values in the classes "u" and "e," respectively. The relative segment sizes are incompatible with this interpretation, however. Thus, the interpretation of that segment sequence as the pair of characters "ue" is incompatible with the fact that both "u" and "e" are lowercase and should be approximately the same size.

Compatibility is, of course, a matter of degree. We used MLFNs to measure how compatible the relative spatial locations of pairs of adjacent segments were with interpretations as character pairs. When memberships in compatibility classes (pattern classes that we defined to represent different configurations of adjacent characters) were employed to augment the fuzzy character confidence assignment networks, we improved word recognition performance from 79.5 percent to 85.3 percent.[6]

### Aggregating confidence values

To find an optimal match between a word in the lexicon and segmentations of the word, we used dynamic programming (an optimization algorithm). Once we'd assigned confidence values and determined character compatibilities, we could come up with the standard objective function, which is the average of the character confidence values. However, in our word recognition system the objective function is the sum of two terms: the average of the character confidence values and the sum of the compatibilities between pairs of characters.

Averaging is an integration process. Fuzzy measure theory provides alternatives to standard integrals, leading to more effective objective functions than averaging. Fuzzy measures are generalizations of probability measures differing in one significant aspect: with probability measures, the "whole" is equal to the "sum of the parts" (if the parts are disjoint). With fuzzy measures, the whole may be more than, less than, or equal to the sum of the parts (even if the parts are disjoint).

Fuzzy integrals generally combine a function with a fuzzy measure in a nonlinear fashion. We used a fuzzy integral called the Choquet integral.[7-9] The Choquet integral is a weighted sum that depends on the character confidence value ordering. In our word recognition application, we determined experimentally that those weights that decrease exponentially as the confidence values increase work well, and so we assigned small weights to large character confidence values. This implies that the network prefers a match with several "medium-sized" character confidence values over a match with one extremely high value. Thus, the Choquet integral reduces the effects of outliers—single segments in the wrong character class that receive a very high score from the neural network.

In our work with word recognition, the Choquet integral reduced to a linear combination of order statistics. The computation was performed as follows: Let $c_1, c_2, \ldots, c_n$ be the character confidence values from a segmentation of length $n$, and let $c_{(1)}, c_{(2)}, \ldots, c_{(n)}$ denote a permutation of these values such that $c_{(1)} \leq c_{(2)} \leq \ldots \leq c_{(n)}$.

We computed the word confidence as a linear combination of these sorted character confidence scores:

$$\text{Fuzzy Integral Word Match Score} = \sum_{i=1}^{n} w_i c_{(i)}$$

This is a nonlinear function of the character confidence scores because the weight applied to a confidence value is determined by the value's position in the sorted confidence values. Different weight schemes result in well-known operators. As an alternative, for example, we could have chosen $n$ to be odd, say $n = 2k + 1$, and $wk = 1$ with all other weights 0 to obtain the median.

We found that a good choice for weights was to compute them as follows: Let $g$ be a parameter between 0 and 1. Let $wn = g$ and

$$w_{n-1} = g(1 + \lambda g)^i$$

Here $\lambda > 0$ is the unique nonzero root of the equation (assuming $g \neq 1/n$)

$$\lambda + 1 \prod_{i=1}^{n} (1 + \lambda g)$$

These weights decrease exponentially as the subscript increases, which implies that high confidence values get smaller weights. Thus, if there is only one high value, the word confidence does not place much importance on that high value. This value is determined by the parameter $g$ and is important in word recognition because sometimes a single noncharacter can look like a perfect character, greatly influencing that final word confidence if standard averaging were used.

## NUMERIC FIELD RECOGNITION

The second problem we discuss for the Postal Service application involved numeric field recognition: We were able to identify a sequence of handwritten digits. Many of the problems from handwritten word recognition occur with numeric field recognition as well. First, the number of digits is unknown; second, it's unclear if something is a digit or part of a digit. Thus we used a similar approach: We segmented an image of a numeric field into primitives and applied dynamic programming to find the best way to assemble them into a sequence of digits, as Figure 4 shows.

Our algorithm assigned low digit-class membership values to the subimages that represented pieces of digits. To do this, it uses a hybrid neural system: a Kohonen self-organizing feature map plus an MLFN. A Kohenen SOFM is a neural network that represents training data in an array of weight vectors. The weight vectors are prototypical of the patterns, and weight vectors that are close in the array look similar.

### Multilayer Feed-Forward Neural Networks

Normally, when MLFNs are applied to handwritten digit recognition, researchers assume that the input is a digit. But, as we've shown, handwritten numbers can be segmented into nondigit fields.

MLFNs can learn highly complex decision boundaries; they are discriminant functions. However, their behavior toward nondigits that vastly differ from the training set is difficult to control without developing a uniform representation of nondigits. We can alleviate this difficulty with a *typicality measure*, which decreases the distance of the unknown test feature vector from the training set. If we interpret membership values as degrees of typicality, which we did, then fuzzy sets are a powerful means to design robust systems that function well in the presence of outliers such as noncharacters. When used in conjunction with the discriminatory capability of the MLFN, typicality—a central theme of fuzzy set theory—is a powerful tool for rejecting noncharacters.

### Self-Organizing Feature Map

We developed a typicality measure through the use of the Kohonen SOFM, which augmented our standard feature sets for digit recognition. The Kohonen SOFM accurately represents the fuzziness of the character classes. The SOFM tries to find a representation for a large number of high-dimensional feature vectors in terms of a given number of prototypes or weight vectors. Each prototype can be viewed as a representative of all feature vectors in its neighborhood. The SOFM performs mapping so that spatial locality is preserved as much as possible—feature vectors in adjacent neighborhoods map to adjacent prototypes. One node is associated with each prototype, and the output of a particular node corresponding to an input can be viewed as the distance of the input vector from the prototype associated with the node.

Figure 5 shows the weight vectors of an SOFM that we trained using $24 \times 18$ binary images of handwritten digits as input feature vectors. While forming the map, we chose a total of 100 nodes such that each node had four adjacent nodes. The figure depicts each weight vector in the form of $24 \times 18$ gray-level images in the location of the corresponding node. The spatial organization ability of the SOFM is quite striking in this example. Most sets of weights appear as gray-level images of characters themselves. Furthermore,

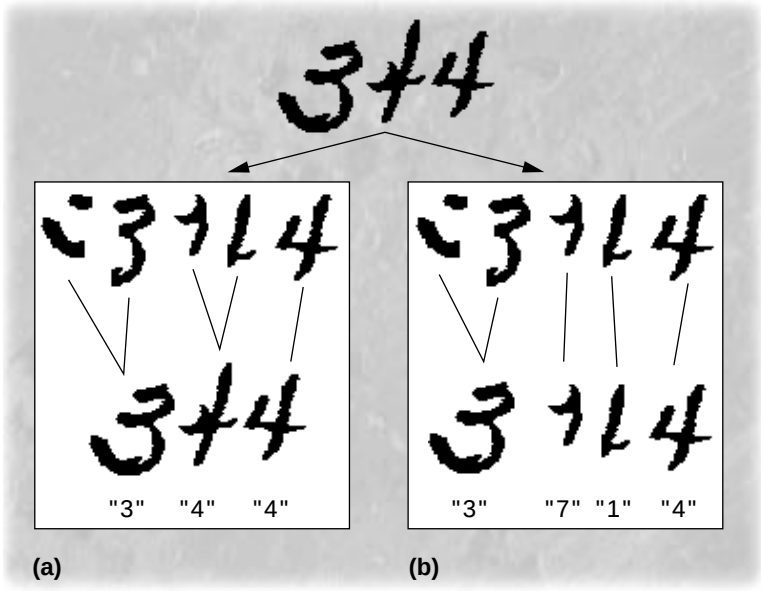


*Figure 4. Numeric field recognition using dynamic programming. For each potential number of digits, (a) and (b), dynamic programming finds the best way to assemble the primitive segments to form a digit string of that length. This enables both hypotheses to be considered by the system for ultimate resolution by higher level processing.*

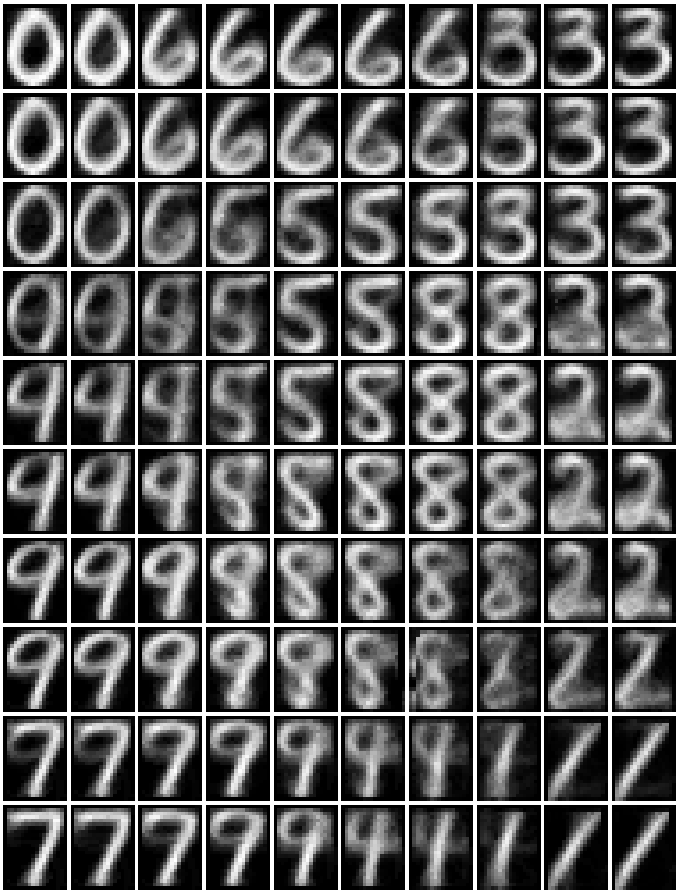

*Figure 5. Each weight vector of this self-organizing feature map is depicted in the form of 24 × 18 gray-level images in the location of the corresponding node. Most sets of weights appear as gray-level images of characters themselves. By traversing the map from left to right, you can see that the weights appear to transition smoothly from one character class to another, which accurately reflects the fact, for example, that 3s, 5s , and 8s are often confused, as are 4s, 7s, and 9s.*

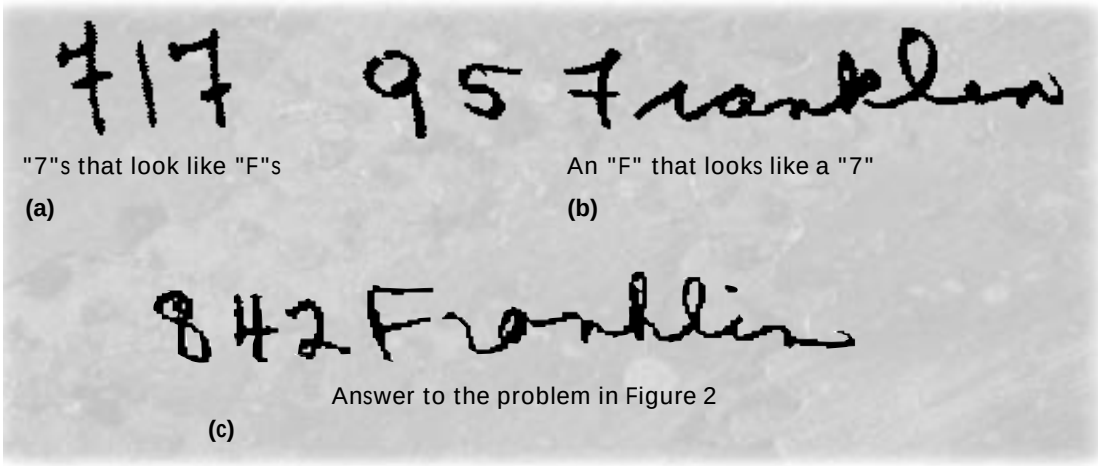


*Figure 6. Typical street location inputs. Characters can look like numbers or overlap with numbers.*

as one traverses the map, the weights appear to transition smoothly from one character class to another. The transitions that appear in the map accurately reflect the common confusion classes in handwritten digit recognition. For example, 3s, 5s , and 8s are often confused, as are 4s, 7s, and 9s.

Since the activation function at an SOFM node represents the distance between the weight vector at the node and the input pattern, the SOFM gives us a way to measure typicality. The scaled output activations of the SOFM can be viewed as typicality or fuzzy set membership values.

The outputs of an SOFM can be used as input features for an MLFN. We used the SOFM to significantly improve the MLFN's capability to assign low confidence values to nondigits while retaining high recognition rates.[10] We extended this work to handwritten word recognition and generalized it to include the fuzzy integral.[4,7] Elsewhere we showed that a possibilistic approach to clustering also overcomes many of the problems associated with outliers.[11]

## STREET NUMBER LOCATION

If handwritten addresses can be interpreted, mail sorting can be automated. If the numeric fields in an address—the street number and zip code—can be identified, the number of possible addresses that match the identified information is significantly reduced. The street number location problem is this: *Given an image block that represents the leftmost block of an address line, determine if it contains a street number and, if so, where*. Figure 6 shows some typical inputs and associated problems.

Our approach to location, as with word and numeric recognition, combined neural and fuzzy techniques. A fuzzy logic rule base combined several measures computed on an image block to produce street number location confidence. Our computer program partitioned an image block into primitive segments, each of which represented the location of a possible ending point of a street number. At each such location, we computed a confidence, using MLFNs and dynamic programming, that the subimage beginning at the leftmost border of the image and ending at that location represented either a numeric field or the phrase "P.O.", and that the next character represented an uppercase capital letter (the beginning of the street name).

In addition, the rule base used features such as sizes of characters and gaps between characters. The fuzzy logic system generated an overall confidence that the street number ended at each location. A typical rule in the fuzzy logic system follows:

```
IF the next primitive is too
  complex to be recognized as a
  digit
AND the numeric field confidence is
  large
AND the gap size between this
  primitive and next primitive is
  medium
THEN the street number confidence
  should be adjusted to positive
  large.
```

The fuzzy logic rule base, consisting of 48 rules,[12] was applied to an independent (blind) test set. The answer produced by the system was deemed correct if there was no street number in the image block and the system responded as such, or if there was a street number in the image block and the system correctly identified its location. The success rate of the system on the blind test data was 86 percent.

We trained a large number of MLFNs to try to match or exceed the performance of the fuzzy logic system. We varied the number of hidden units, the learning rate, and the momentum parameters over a

wide range, with both balanced and unbalanced training. The best success rate we could achieve was only 79 percent, 7 percent below that achieved by the fuzzy logic system.

Why do the MLFNs perform so well for other tasks but not for this task? One interpretation is that the granularity of knowledge required to locate street numbers is "coarser" or more "vague" than that required to perform character recognition. Coarse knowledge that cannot be represented statistically in the training data is difficult or impossible for neural networks or fuzzy systems to learn, but such knowledge can be encoded into fuzzy rules easily. Thus, by combining the neural network techniques that can generate fine decision boundaries and fuzzy logic techniques that can handle coarser knowledge, we can achieve excellent results.

Fuzzy and neural methods can be applied at many levels in complex handwriting recognition systems. They can represent and process ambiguous, low-level information (such as fuzzy character-class memberships), and they support delayed decision making in keeping with the Principle of Least Commitment. Fuzzy logic systems can represent high-level knowledge (such as that used in street number location) and can combine multiple sources of uncertain low-level information to arrive at accurate estimates of confidence in high-level hypotheses. Fuzzy measures are related to methods of robust statistics, namely linear combinations of order statistics, and these methods are useful for combining information that may contain outliers, a common situation in handwriting recognition.

Research in this area will continue. We have not yet considered the application of fuzzy and neural methods at levels higher than the word or numeric field levels. Commercial handwriting recognition systems are required to process entire documents, not just individual words. For example, reading entire handwritten addresses or complete phrases in handwritten forms (such as Census Bureau forms) are also difficult for computer systems, and all the principles and problems discussed in this article are applicable there as well. Fuzzy and neural methods can contribute to the solution of these automation problems. ❖


### Acknowledgments

Much of the research we described here was supported by the United States Postal Service through the Environmental Research Institute of Michigan (ERIM). We also acknowledge the significant contributions, through ideas and discussions, made by ERIM researchers Andrew Gillies, Daniel Hepp, Mike Whalen, and Margaret Ganzberger.


### References


1. R. Hecht-Nielson, *Neurocomputing*, Addison-Wesley, Reading, Mass., 1990.
2. P.D. Gader et al., "Handprinted Word Recognition on a NIST Data Set," *Machine Vision and Its Applications*, Vol. 8, 1995, pp. 31-40.
3. P. Gader, M. Mohamed, and J. Chiang, "Comparison of Crisp and Fuzzy Character Neural Networks in Handwritten Word Recognition," *IEEE Trans. Fuzzy Systems*, Aug. 1995, pp. 357-364.
4. J.-H. Chiang and P.D. Gader, "Hybrid Fuzzy-Neural Systems in Handwritten Word Recognition," *IEEE Trans. Fuzzy Systems*, accepted for publication.
5. D. Marr, *Vision*, W.H. Freeman, San Francisco, 1982.
6. P.D. Gader, M. Mohamed, and J.-H. Chiang, "Handwritten Word Recognition with Character and Inter-Character Neural Networks," *IEEE Trans. Systems and Man Cybernetics*, accepted for publication.
7. P.D. Gader, M.A. Mohamed, and J.M. Keller, "Dynamic Programming Based Handwritten Word Recognition Using the Choquet Fuzzy Integral as the Match Function," *J. Electronic Imaging*, Jan. 1996, pp. 15-24.
8. T. Murofushi and M. Sugeno, "A Theory of Fuzzy Measures: Representations, the Choquet Integral, and Null Sets," *J. Math Analysis and Applications*, Vol. 159, 1991, pp. 532-549.
9. M. Grabisch, H. Nguyen, and E. Walker, *Fundamentals of Uncertainty Calculi with Applications to Fuzzy Inference*, Kluwer Academic, Dordrecht, Germany, 1995.
10. J.-H. Chiang and P.D. Gader, "Improving Digit Recognition Reliability by a Hybrid Neural Model," *Proc. Int'l Conf. CFSA/IFIS/SOFT '95 on Fuzzy Theory and Applications*, IEEE CS Press, Los Alamitos, Calif., 1995, pp. 182-187.
11. R. Krishnapuram and J. Keller, "A Possibilistic Approach to Clustering," *IEEE Trans. Fuzzy Systems*, May 1993, pp. 98-111.
12. P. Gader, J. Keller, and J. Cai, "A Fuzzy Logic System for the Detection and Recognition of Street Number Fields on Handwritten Postal Addresses," *IEEE Trans. Fuzzy Systems*, Feb. 1995, pp. 83-96.



*Paul D. Gader is an assistant professor in the Department of Computer Engineering and Computer Science at the University of Missouri. He has worked at the Environmental Research Institute of Michigan and at Honeywell Systems and Research Center. He has been actively involved in handwriting recognition, automatic target recognition, image algebra and morphological image processing, obstacle detection from low-flying helicopters, medical image analysis, vehicle identification, scene analysis, and applied mathematics. Gader received a BS in mathematics from the University of Central Florida and an MS and a PhD in mathematics, both from the University of Florida.*

*James M. Keller is a professor in the Computer Engineering and Computer Science Department at the University of Missouri and is the R.L. Tatum Research Professor in the College of Engineering. His research interests include computer vision, pattern recognition, fuzzy set theory and fuzzy logic, fractal geometry, and neural networks. He is an associate editor of* IEEE Transactions on Fuzzy Systems *and the* International Journal of Approximate Reasoning *and serves on the editorial board of the* Journal of Intelligent and Fuzzy Systems*. Keller received a PhD in mathematics from the University of Missouri. He is a member of IEEE.*

*Raghu Krishnapuram is an associate professor in the Computer Engineering and Computer Science Department at the University of Missouri. He was a Humboldt Fellow at the European Laboratory for Intelligent Techniques Engineering. His research interests include applications of fuzzy set theory and neural networks to pattern recognition and computer vision. Krishnapuram received a BTech. in electrical engineering from the Indian Institute of Technology, Bombay, an MS in electrical engineering from Louisiana State University, and a PhD in electrical and computer engineering from Carnegie Mellon University.*

*Jung-Hsien Chiang is an associate professor in the Department of Information Management at the Chaoyang Institute of Technology, Taiwan. His research interests include neural networks, fuzzy systems, pattern recognition, and the modeling of decision making. Chiang received a BSc in electrical engineering from the National Taiwan Institute of Technology and an MS and a PhD in electrical and computer engineering from the University of Missouri, Columbia.*

*Magdi A. Mohamed is a senior staff engineer at Motorola Software Enterprises. He was a visiting professor in the Computer Engineering and Computer Science Department at the University of Missouri, Columbia. His research interests include handwriting systems, image processing, computer vision, fuzzy set theory, neural networks, pattern recognition, parallel and distributed computing, artificial intelligence, and fractals and chaos theory. Mohamed received a PhD in electrical engineering from the University of Missouri, Columbia. He is a member of IEEE.*

*Contact the authors at the University of Missouri, Dept. of Computer Engineering and Computer Science, Columbia, MO 65211; gader@ece.missouri.edu.*