Comparison of Crisp and Fuzzy Character Neural Networks in Handwritten Word Recognition

Paul Gader, Magdi Mohamed, and Jung-Hsien Chiang

Abstract— **Experiments comparing neural networks trained with crisp and fuzzy desired outputs are described. A handwritten word recognition algorithm using the neural networks for character level confidence assignment was tested on images of words taken from the United States Postal Service mailstream. The fuzzy outputs were defined using a fuzzy k-nearest neighbor algorithm. The crisp networks slightly outperformed the fuzzy networks at the character level but the fuzzy networks outperformed the crisp networks at the word level. This empirical result is interpreted as an example of the principle of least commitment.**

I. Introduction

TRAINING NEURAL network-based character recognizers using fuzzy class memberships rather than crisp class memberships as desired outputs can result in more useful character recognition modules for handwritten word recognition. In this paper, we describe experiments that support this statement. This result is not obvious; handwriting recognition researchers almost always train and evaluate character recognizers using crisp class memberships [1]–[3].

Handwritten word recognition is a difficult problem. It was not until 1992 that significant progress began to be reported [1], [3] although several earlier attempts were made, cf. [4]. Machine-printed word recognition is an easier task but still presents difficulty [5]–[8]. A glance at the current research (cf. [1]–[8]) and a little thought should convince a reader that handwritten and machine-printed word recognition by computer consists of more than just isolating and reading the individual characters in a word. Contextual information is not only useful, it is often necessary. People are able to read words with illegible and ambiguous characters. Many alphabetic characters are ambiguous when read out of context. In fact, the same pixel pattern can represent different characters in different words. Furthermore, multiple characters can look like characters. Some examples of ambiguity are shown in Fig. 1.

One implication of these considerations is that recognition rates may not be the proper performance measure of an character classifier that is to be used for word recognition. Accurate representation of ambiguity is more important. If a character is labeled as a "u" (because, for example, it was used as a "u" in a word) but could be a "u" or "v," then the desired output for that sample should reflect the ambiguity. Fuzzy set membership of character images in character classes is a natural approach to developing classifiers with this property.

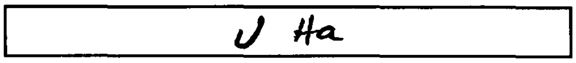


Fig. 1. Ambiguous patterns often occur in handwritten words. These patterns were taken from words in which they were used as "u" and "tta." They could be used equally as well as "J" and "Ha."

Although this approach seems natural, it is not commonly used. Researchers in handwriting recognition train and evaluate character recognition modules using crisp class memberships. The results in this paper are important in that they demonstrate, in a significant application on a complex, real problem, that fuzzy sets provide improved performance.

The purpose of this paper is to describe experiments comparing the use of neural networks trained with crisp and fuzzy desired outputs in handwritten word recognition. A segmentation-based handwritten word recognition algorithm that uses neural network classifiers on the character level to read a word is described. The features and neural network architectures used for character recognition are defined. The fuzzy k-nearest neighbor algorithm for assigning fuzzy desired outputs to training samples is given. Experimental results that indicate the fuzzy approach performs better at the word level but not at the character level are provided.

This result is consistent with the principal of least commitment which encourages delaying decision making until the maximum amount of information is available in the time allowed [9]. Providing more noncommittal confidence levels, and therefore multiple possibilities, at the character level delays the crisp decision making and allows the word classification decision to be more influenced by contextual information.

II. Neural Network-Based Character Recognition

The features used as inputs to the character recognition neural networks are 120-dimensional feature vectors that encode directional information. The neural networks are multilayer feedforward networks. Desired outputs are formed using ordinary crisp class encoding and a fuzzy k nearest neighbor algorithm. These character recognizers are now described in more detail followed by character recognition results.

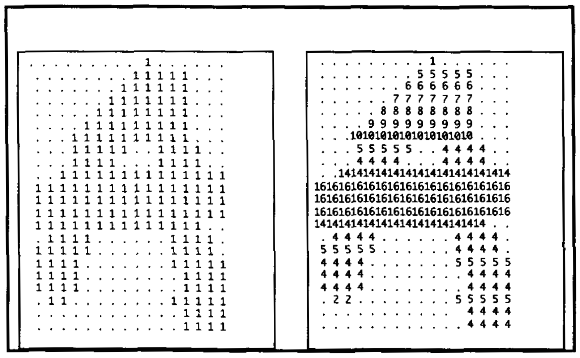
A. Features

The features are called bar-features [10]. The bar-features are calculated on binary images of isolated characters that need not be size normalized. Initially, eight feature images

Manuscript received July 9, 1993; revised February 23, 1995. This research was supported by the Environmental Research Institute of Michigan and the United States Postal Service.

The authors are with the Department of Electrical and Computer Engineering, University of Missouri-Columbia, Columbia, MO 65211 USA.

IEEE Log Number 9412968.



```
. . . . . . . . . 1 . . . . . .
. . . . . . . . 1 1 1 1 1 . . .
. . . . . . . 1 1 1 1 1 1 . . .
. . . . . . 1 1 1 1 1 1 1 . . .
. . . . . 1 1 1 1 1 1 1 1 . . .
. . . . 1 1 1 1 1 1 1 1 1 . . .
. . . 1 1 1 1 1 1 1 1 1 1 . . .
. . . 1 1 1 1 1 . . 1 1 1 1 . .
. . . 1 1 1 1 . . . 1 1 1 1 . .
. . 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 . .
. 1 1 1 1 . . . . . . 1 1 1 1 .
1 1 1 1 1 . . . . . . 1 1 1 1 .
1 1 1 1 . . . . . . . 1 1 1 1 1
1 1 1 1 . . . . . . . . 1 1 1 1
1 1 1 1 . . . . . . . . 1 1 1 1
. 1 1 . . . . . . . . 1 1 1 1 1
. . . . . . . . . . . . 1 1 1 1
. . . . . . . . . . . . 1 1 1 1
```

```
. . . . . . . . . 1 . . . . . .
. . . . . . . . 5 5 5 5 5 . . .
. . . . . . . 6 6 6 6 6 6 . . .
. . . . . . 7 7 7 7 7 7 7 . . .
. . . . . 8 8 8 8 8 8 8 8 . . .
. . . . 9 9 9 9 9 9 9 9 9 . . .
. . .10101010101010101010 . . .
. . . 5 5 5 5 5 . . 4 4 4 4 . .
. . . 4 4 4 4 . . . 4 4 4 4 . .
. .1414141414141414141414141414
16161616161616161616161616161616
16161616161616161616161616161616
16161616161616161616161616161616
1414141414141414141414141414 . .
. 4 4 4 4 . . . . . . 4 4 4 4 .
5 5 5 5 5 . . . . . . 4 4 4 4 .
4 4 4 4 . . . . . . . 5 5 5 5 5
4 4 4 4 . . . . . . . . 4 4 4 4
4 4 4 4 . . . . . . . . 4 4 4 4
. 2 2 . . . . . . . . 5 5 5 5 5
. . . . . . . . . . . . 4 4 4 4
. . . . . . . . . . . . 4 4 4 4
```

Fig. 2. A character image and the feature image for the foreground horizontal direction.

are generated. Each feature image corresponds to one of the directions: east, northeast, north, and northwest, in either the foreground or the background. Each feature image has an integer value at each location that represents the length of the longest bar that fits at that point in that direction.

A recursive, two-pass algorithm is used to compute the feature images. In the forward pass the image is scanned left-to-right and top-to-bottom. At each point, either the foreground or background feature images are updated as shown below. On the backward, pass, the maximum is propagated back up from bottom-to-top, right-to-left. Pseudo-code for the computing the bar-features on the foreground is as follows:

```
/* FORWARD PASS */
  FOR i = 1,2,···, nrows DO
   FOR j = 1,2,···, ncols DO
      e(i,j)  = e(i,j − 1) + 1
      ne(i,j) = ne(i − 1,j + 1) + 1
      n(i,j)  = n(i − 1,j) + 1
      nw(i,j) = nw(i − 1,j − 1) + 1

/* BACKWARD PASS */
  FOR i = nrows,nrows − 1,···,1 DO
   FOR j = ncols,ncols − 1,···,1 DO
      e(i,j)  = max(e(i,j),e(i,j + 1))
      ne(i,j) = max(ne(i,j),ne(i + 1,j − 1))
      n(i,j)  = max(n(i,j),n(i + 1,j))
      nw(i,j) = max(nw(i,j),nw(i + 1,j + 1)).
```

An example of an original image and the bar-feature image $e(i,j)$ for the foreground is shown in Fig. 2. Each point is given a value equal to the longest horizontal bar that fits in the horizontal direction. In this way, the bar-feature images measure the amount of "horizontalness" at a point.

Feature vectors are computed from the feature images using overlapping zones. Fifteen rectangular zones arranged in five rows with three zones each are used; each zone is of approximate size $h/3 \times w/2$ where h and w are the height and width of the image, respectively. The upper left hand corners of the zones are approximately at positions $\{(r,c)|r = 0, h/6, 2h/6, 3h/6, 4h/6$ and $c = 0, w/4, 2w/4\}$. The approximates are due to the fact that not all image sizes are exactly divisible by four or six. The values in each zone in each feature image are summed. The sums are normalized between zero and one by dividing by the maximum possible sum in a zone. The feature vector is of dimension $15 \times 8 = 120$. More precisely, let $f_1, f_2, \cdots, f_8$ be feature images associated with an input pattern and let z_i denote a rectangular zone of size $h/3 \times w/2$ with upper left hand corner at (r_0, c_0). The kth feature value associated with zone z_i is

$$z_{ik} = (\frac{1}{N}) \sum_{r=r_0}^{r_0+(W/2)} \sum_{c=c_0}^{c_0+(h/3)} f_k(r,c)$$

where

$$\begin{aligned} N &= w \quad \text{if } f_k = e, ne, \text{ or } nw \\ &= h \quad \text{if } f_k = n. \end{aligned}$$

The vector of 120 values $(z_{ik}: i = 1, 2, \cdots, 15$ and $k = 1, 2, \cdots, 8)$ are the bar-features which are used as inputs to the character recognition neural networks.

B. Network Structure

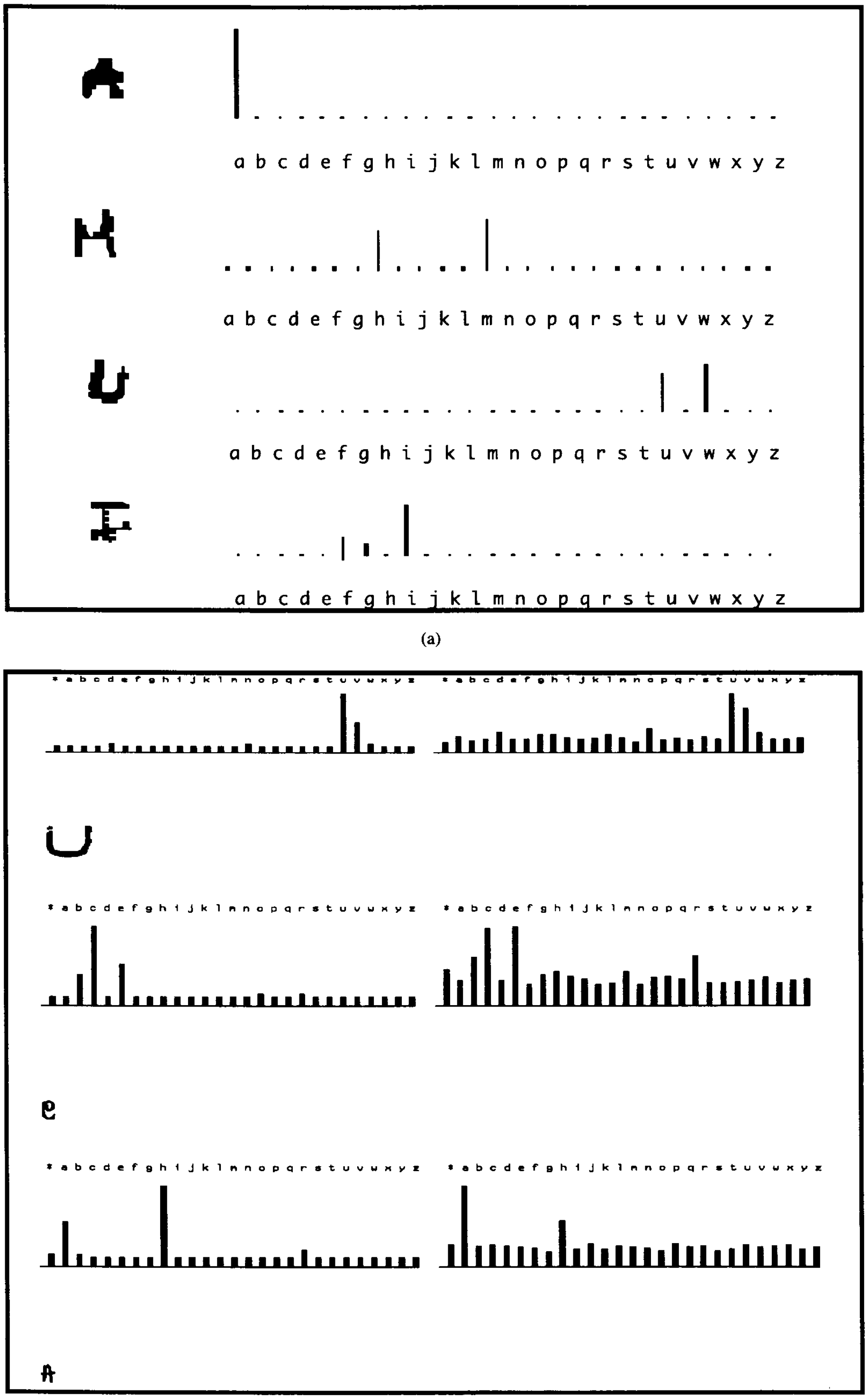
Separate networks were used for upper and lower case characters. The networks were trained using backpropagation. Each has input, output, and two hidden layers. Each hidden and output unit has a bias. Each network has 120 input units, 65 units in the first hidden layer, 39 in the second hidden layer, and 27 outputs units, one for each class and one for images of noncharacters (e.g., pieces of or multiple characters). A sigmoid with values between −0.5 and 0.5 was used.

C. Computation of Desired Outputs

Each character in the training set has a "true" class. Each character image was extracted from a word image; it was used as a specific character class in the word. That class was recorded as the true class. The desired outputs for the crisp networks were obtained by setting the desired output for the true class to 0.4 and for all other classes to −0.4. The desired outputs for the fuzzy networks were set using a fuzzy k-nearest neighbor algorithm suggested by Keller *et al.* [11]. The idea is to assign membership based on the percentage of characters in each class among the neighbors of a training sample.

Two variations of the algorithm were used, each with a different constraint on the desired output of the true class: 1) the desired output for the true class cannot be lower than the desired output for any other class, and 2) the desired output for the true class is high, 0.4. The first variation is Keller's formulation. The second variation can be thought of as interpreting the memberships as a possibility distribution; the possibility that the pattern can be used as the true class is high since we know that it was used at least once as a representative of the class. The algorithm is shown at the bottom of the next page.

We chose to use the 20 nearest neighbors of a training sample using Euclidean distance. Some examples of desired outputs computed using Keller's algorithm are shown in Fig. 3(a). Recall that the true class is always the highest output class. Some examples of desired outputs determined using the possibilistic constraint are shown in Fig. 3(b). Fig. 3(b) also



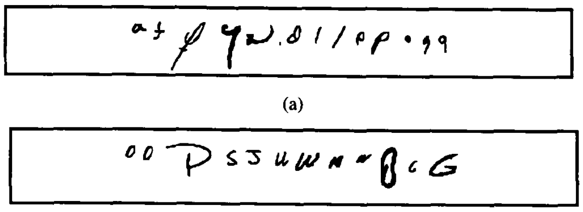
(a)

(b)

Fig. 3. (a) Some uppercase characters and their fuzzy set memberships as determined by the fuzzy k-nearest neighbor algorithm. The true classes of these characters are "A," "M," "W," and "I." (b) Some handwritten characters shown with their desired outputs as formed by the possibilistic k-nearest neighbor algorithm (left) and the actual outputs as computed by the neural networks after training (right). Class * is the noncharacter class. The true classes of these characters are "U," "c," and "H."

TABLE I
Training Results for Fuzzy and Crisp Networks

	FUZZY RECOGNITION RATE	FUZZY RMS ERROR	CRISP RECOGNITION RATE	CRISP RMS ERROR
UPPER CASE	86.4%	0.072	93.3%	0.071
LOWER CASE	81.2%	0.079	85.0%	0.090



(a)

(b)

Fig. 4. (a) Sample characters from the classes “a” “f” “f” “y” “z” “d” “i” “l” “e” “p” “i” “g” “q.” (b) Sample characters from the classes “D” “O” “P” “S” “J” “U” “W” “M” “W” “B” “C” “G.”

shows the outputs computed by the networks after training. As before, the true class is always a maximal output class.

D. Character Recognition Results

Character training sets were constructed by extracting characters from images of handwritten words. The positions of the characters in the words were identified manually, and the characters were extracted and sorted by class automatically. The word images and character positions were obtained from the Environmental Research Institute of Michigan (ERIM). The extracted characters were manually screened.

Several thousand characters were collected. Some classes were very poorly represented; there were over 3000 “A’s” but fewer than 10 “j’s.” Poorly represented classes were augmented with characters from the National Institute of Standards (NIST) data set [12], [13]. The result was a training set of 250 character images per class. Sample characters are shown in Fig. 4. Since the ultimate goal was word recognition, testing was performed at the word recognition level.

Training results are shown in Table I. The table gives the percentage of characters for which the highest output was associated with the true class (the recognition rate) and the root mean squared (RMS) error. The recognition rate is a standard measure of performance for character recognizers. The RMS error is a standard measure used in training neural networks. Standard wisdom is that high recognition rates and low RMS error rates are good. The crisp networks achieve a higher recognition rate than the fuzzy networks with similar or higher RMS error. One interpretation is that better representation of ambiguity increases the likelihood that the network will make a mistake when “forced” to decide at the character level.

III. Word Recognition

The word recognition algorithm uses image segmentation and dynamic programming. The inputs are a binary image of a word and a lexicon (a list of strings representing all candidate word identities). The algorithm assigns a confidence value to each string in the lexicon. The confidence value indicates the degree to which the word image matches the string. The confidence values are assigned to the strings by finding optimal matches between segmentations of the word image and the strings using dynamic programming. The string with the highest confidence can be considered to be the recognition result.

A word image is segmented into subimages called primitive segments, or primitives. Each primitive ideally consists of a character or a part of a character. A segment is either a primitive or a union of primitives. Ordering primitives from left to right yields a partial ordering on the segments. A segmentation into characters is a path through the partially ordered set of segments. Dynamic programming is used to find the best path, that is, the best sequence of segments to match a given string. The value of a path is the sum of the match scores of the segments in the path to the characters in the string as given by the output activations of the neural networks. An overview is shown in Fig. 5. The segmentation computation is performed once for every word image. For each word image, the dynamic programming matching is performed once for every string in the lexicon associated with the image. Fig. 5 shows only one match against the correct string. Many other matches must be performed as well.

A. Segmentation

The segmentation process is a refinement of that described in [13] and [14]. We describe it briefly. Connected components

```
Compute the bar-features for all character samples in the training set
For each character sample s in the training set
   Set the desired output for the true class to 1.0
   For each other class C
   Set the desired output to $a\left(\frac{N_c}{k}\right)$
   where:
      $N_c$ is the number of samples of class C among the k-nearest neighbors of s
      a is a parameter between 0 and 1
The desired outputs are mapped to the interval $[-0.4, 0.4]$ for training.
```

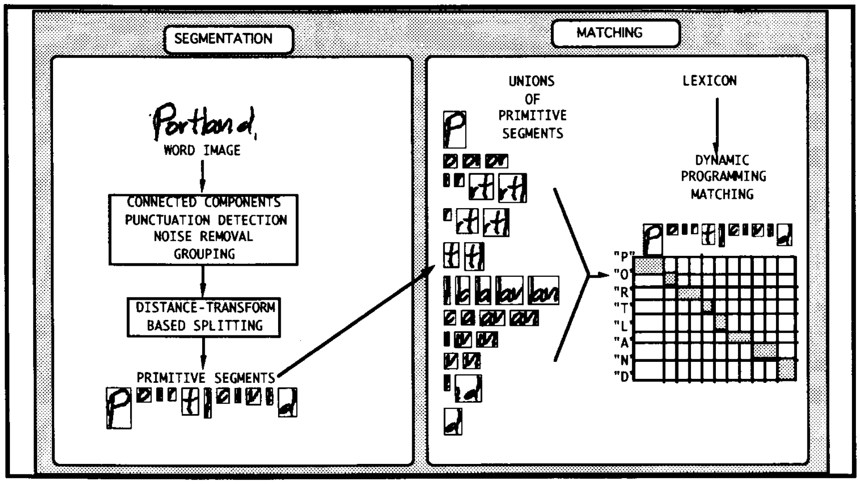


Fig. 5. Overview of word recognition algorithm.

are computed, punctuation is detected and removed, and bars, such as those in "D," "E," "F," etc., are grouped with larger connected components. The result is a set of initial segments that generally consist of images of one or more characters. Those consisting of more than one character need to be split.

Each segment in the initial segmentation is passed through a splitting module. The splitting module uses the distance transform to detect possible locations to split initial segments. The distance transform encodes each pixel in the background using the distance to the stroke. Split paths are formed that stay as far from the stroke as possible without turning too much. Heuristics are used to define starting points for the paths based on the shape of the image and to modify the distance function.

The sequence of images resulting from splitting and initial segmentation are postprocessed to correct for images that are very small or very complex. The postprocessed sequence is the set of primitive segments.

B. Dynamic Programming Matching

The dynamic programming matcher takes a word image, a string, and the word image primitives and returns a value between zero and one indicating the degree to which the word image represents the string. The algorithm is implemented using a array approach. The rows of the array correspond to the characters in the string. The columns of the array correspond to primitives. The ij element is the best match between the first i characters in the string and the first j primitives. This value may be $-\infty$ if there is no legal match.

Let the primitives be $S_1, S_2, \cdots S_p$ and the characters in the string be $C_1, C_2, \cdots C_w$. Let $m(s, c) = \max\{$class c output activation of lower case network given input s, class c output activation of upper case network given input $s\}$. The value $m(s, c)$ is the confidence that s represents c. The ij element of the array, $v(i, j)$, is computed as follows (see the algorithm shown at the bottom of the page).

Let S_{kj} denote the union of primitive segments $S_k \cup S_{k+1} \cup \cdots \cup S_j$.

A union of two segments is considered legal if the two segments pass a sequence of tests. The tests measure closeness and complexity.

C. Word Recognition Results

The word recognition test was run on two sets with 250 word images in each set. The first set is referred to as words250 and the second as words500. The word images were binary images of city and state words that were extracted, binarized, and downsampled by a factor of two in each direction from 300 dots per inch, eight-bit gray scale images of addresses at ERIM. Thus, each word image represents the name of a city, state, or street.

One base lexicon of 100 strings was used with each word image. The base lexicons consisted of random collections of city, state, and street names. If the correct string representing the true word in the image was not in the lexicon, then the correct string was added to the true lexicon. Thus, the lexicon was either of size 100 or 101 for each word. Misspellings were not accounted for in the lexicon. Thus, "Buffalo" and

```
If $i = 1$, (matching against the first character) Then
    $v(1,j) = m(S_{1j}, C_1)$                        for all $j$ such that $S_{1j}$ is legal
            $= -\infty$                              otherwise
If $i > 1$ Then
    $v(i,j) = \max_k (v(i-1,k) + m(S_{kj}, C_i))$    for all $k, j$ such that $S_{kj}$ is legal
            $= -\infty$                              otherwise
```

TABLE II(A)
WORD RECOGNITION RESULTS USING FUZZY NETWORKS

RANK	WORDS250	WORDS500
1	74.4%	62.4%
2	81.2%	72.4%
3	84.0%	76.8%

TABLE II(B)
WORD RECOGNITION RESULTS USING CRISP NETWORKS

RANK	WORDS250	WORDS500
1	69.2%	58.8%
2	78.0%	67.6%
3	81.2%	73.2%

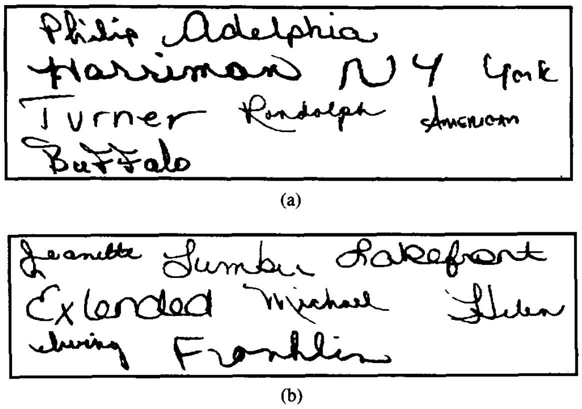


Fig. 6. (a) Some correctly recognized word images. (b) Some incorrectly recognized word images.

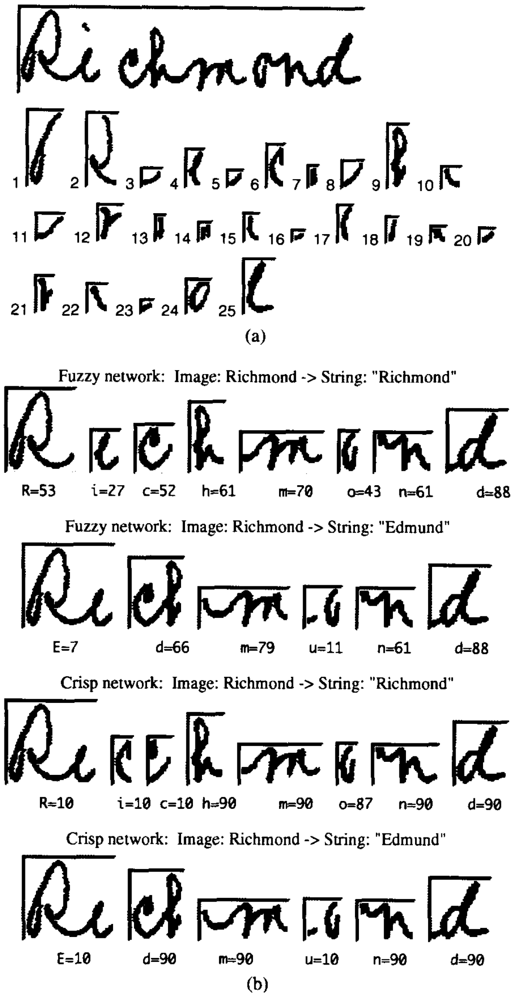

Richmond
1 2 3 4 5 6 7 8 9 10
11 12 13 14 15 16 17 18 19 20
21 22 23 24 25
(a)
Fuzzy network: Image: Richmond -> String: "Richmond"
R=53 i=27 c=52 h=61 m=70 o=43 n=61 d=88
Fuzzy network: Image: Richmond -> String: "Edmund"
E=7 d=66 m=79 u=11 n=61 d=88
Crisp network: Image: Richmond -> String: "Richmond"
R=10 i=10 c=10 h=90 m=90 o=87 n=90 d=90
Crisp network: Image: Richmond -> String: "Edmund"
E=10 d=90 m=90 u=10 n=90 d=90
(b)

Fig. 7. (a) A handwritten word image and the primitive segments generated by the segmentation algorithm. (b) Best matches of the handwritten word image in (a) to the strings "Richmond" and "Edmund" using fuzzy and crisp networks.

"Buffala" were considered to be distinct strings although they both represent the city of Buffalo, NY. Case was not assumed to be known either. Thus, "Buffalo" and "BUFFALO" were assumed to be the same strings.

The word recognition system was run on each set twice, once with the crisp networks and once with the fuzzy networks. The results are shown in Table II. Recall that the word recognition system assigns a confidence score to each string in the lexicon. Thus, a ranking of the lexicon is produced by the algorithm. The rows of each table indicate the percentage of word images for which the correct string was ranked at or above rank $i. i = 1, 2, 3$.

The rank 1 percentage can be considered to be the recognition rate. As can be seen, the fuzzy networks produced approximately 5 and 2.5% increases in recognition rates on words250 and words500, respectively. Fig. 6(a) shows some examples of correctly processed images and Fig. 6(b) shows some examples of incorrectly processed images.

Fig. 7 shows an example. In Fig. 7(a), a handwritten word image representing "Richmond" is shown together with the primitive segments generated by the segmentation algorithm. This word is processed correctly using fuzzy neural networks but not using crisp ones. The crisp neural networks chose the string "Edmund." In Fig. 7(b), four segmentations of the word image are shown. The segmentations shown are the those that provided the best matches to either the strings "Richmond" or "Edmund" by the dynamic programming matcher. The first two represent the segmentation found by the fuzzy system when matching the image to "Richmond" and "Edmund," respectively. The last two represent the segmentation found by the crisp system when matching the image to "Richmond" and "Edmund," respectively.

For example, in the fuzzy match to "Richmond," the dynamic programming matcher matched primitive segment unions $S_{1,3}$ to "R," $S_{4,5}$ to "i," $S_{6,8}$ to "c," $S_{9,10}$ to "h," $S_{11,16}$ to "m," $S_{17,18}$ to "o," $S_{19,22}$ to "n," and $S_{23,25}$ to "d."

The confidence values, or grades of membership, assigned by the neural networks are shown below each segment in Fig. 7(b). These values have been scaled from the range $[-0.4, 0.4]$ to the range $[10, 90]$. The score for each string can be obtained by summing the confidence values and dividing by the length of the string. Thus the fuzzy network matches the word image to the strings "Richmond" and "Edmund" with match scores 57 and 52, respectively, and the crisp network with match scores 48 and 63, respectively.

The crisp network makes hard decisions; the values are usually either (approximately) the lowest possible or the highest possible. It does not find the correct segmentation to match the string "Richmond" because the neural network confidence score for the segment which truly represents "R" is 10 (the lowest possible). The crisp networks assign the same score to the incorrect segment, resulting in a situation in which no real preference is given to either segment. Thus, the choice made by the dynamic programming matcher is essentially random. The second segment in the incorrect segmentations, consisting of the "c" and part of the "h," matching "Edmund" is also interesting. Both the crisp and fuzzy networks generate high confidence on this segment for "d." In fact, the segment does look somewhat like a "d." It is not, however, a very nice "d." The crisp network assigns the segment the highest possible confidence whereas the fuzzy network assigns a more reasonable value for the true degree of membership.

IV. Conclusion

We have described an experiment that supports the use of fuzzy character class memberships in assigning character confidence for word recognition. Our results indicate that when evaluating character recognizers for use in word recognition, character recognition rates are not necessarily good indicators. Careful representation of ambiguity between characters may be more useful than high recognition rates.

Methods for measuring the success of representing the ambiguity between character classes are needed. This is a difficult problem and a special case of the problem of estimating membership functions. The success for representing ambiguity should be directly related to success in achieving the final goal, in this case, high word recognition rates. Neither recognition rates nor RMS error seem to capture this quality.

Assigning membership functions to character image samples by humans and measuring the degree to which an algorithm, such as the fuzzy k-nearest neighbor, agrees with the human assignment is appealing. The human assignment, however, is labor-intensive, subjective, and may vary significantly for different humans. Furthermore, it is not clear that the human assigned membership functions are the "best" possible for reaching the final goal.

Application specific measures may be most appropriate. In the word recognition example, the best measure of success is whether the word recognition algorithm achieves higher recognition rates.

Acknowledgment

The authors would like to acknowledge the support of the United States Postal Service and the Environmental Research Institute of Michigan (ERIM) in this research. ERIM provided financial support, data, and valuable technical interaction; A. M. Gillies, D. Hepp, M. Ganzberger, and M. Whalen all participated in various aspects of the development of the word recognition system described here. The authors would especially like to thank J. Tan of Arthur D. Little, Inc. and C. O'Connor of the United States Postal Service for their support of this work. The advice of J. Keller of the University of Missouri-Columbia in the area of membership function assignment is appreciated. Finally, the authors would like to thank the reviewers for their helpful comments.

References

[1] S. Srihari, Ed., in *Proc. 3rd Int. Workshop Frontiers Handwriting Recognition*, Buffalo, NY, May 1993.

[2] G. Lorette, Ed., in *Proc. Int. Conf. Document Anal. Recognition*, Saint-Malo, France, Oct. 1991.

[3] R. Weirich and W. Dowling, Eds., in *Proc. U.S. Postal Service Advanced Tech. Conf.*, Washington DC, Nov. 1992.

[4] R. M. Bozinovic and S. N. Srihari, "Off-line cursive script word recognition," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 11, no. 1, Jan. 1989.

[5] T. K. Ho, J. J. Hull, and S. N. Srihari, "Word recognition with multi-level contextual knowledge," in *Proc. Int. Conf. Document Anal. Recognition*, Saint-Malo, France, Oct. 1991, pp. 905–915.

[6] T. K. Ho, "A word shape analysis approach to recognition of degraded word images," in *Proc. U.S. Postal Service Advanced Tech. Conf.*, Washington DC, Nov. 1990, pp. 207–233.

[7] A. M. Gillies, "Word verification for the contextual analysis of address block images," in *Proc. U.S. Postal Service Advanced Tech. Conf.*, Washington DC, Nov. 1990, pp. 247–255.

[8] R. A. Duderstadt, "Isolated word recognition for postal address processing," in *Proc. U.S. Postal Service Advanced Tech. Conf.*, Washington DC, Nov. 1990, pp. 233–247.

[9] D. Marr, *Vision*, San Francisco CA: W. H. Freeman, 1982, p. 106.

[10] P. D. Gader, M. Mohamed, and J.-H. Chiang, "Fuzzy and crisp handwritten alphabetic character recognition using neural networks," in *Proc. Artificial Neural Networks Eng.*, St. Louis, MO, Nov. 1992, pp. 421–427.

[11] J. M. Keller, M. R. Gray, and J. A. Givens, "A fuzzy k-nearest neighbor algorithm," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-15, no. 4, pp. 580–581, July/August 1985.

[12] C. L. Wilson and M. D. Garris, "Handprinted character database," National Institute of Standards and Tech. Rep., April 1990.

[13] P. D. Gader and M. P. Whalen, "Advanced Research in Handwritten ZIP Code Recognition," final Tech. Rep. USPS Office of Advanced Technology, Dec. 1990.

[14] M. P. Whalen, M. Ganzberger, and P. D. Gader, "Handprinted word recognition II: Segmentation and verification," *Machine Vision Applicat.*, vol. 8, pp. 31–40, 1995.