Comparison of Crisp and Fuzzy Character Networks in Handwritten Word Recognition

Paul Gader, Magdi Mohamed, and Jung-Hsien Chiang

Department of Electrical and Computer Engineering
University of Missouri-Columbia
Columbia, MO 65211

This research was supported by the Environmental Research Institute of Michigan and the United States Postal Service Office of Advanced Technology

ABSTRACT

Experiments involving handwritten word recognition on words taken from images of handwritten address blocks from the United States Postal Service mailstream are described. The word recognition algorithm relies on the use of neural networks at the character level. The neural networks were trained using crisp and fuzzy desired outputs. 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.

INTRODUCTION

Handwritten word recognition by computer is a very difficult task. Although considerable research has been performed in character recognition, not much has been done in word recognition. Interest has picked up lately, as can be seen by viewing the contents of the proceedings of recent conferences in these areas [1,2,3,4]. Even in the machine-printed case, word recognition consists of more than just reading the individual characters in the word [5,6,7]. 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 difference characters in different words. Furthermore, multiple characters can look like characters. For example, the "tl" in the image of the word "Portland" in Figure 8 could be an "H".

The implication of this is that high recognition rates may not be the ultimate goal of an alphabetic character classifier that is to be used in word reading. Accurate representation of ambiguity is more important. Thus if a certain character in the training set is called a "u" but could be either a "u" or "v", then the desired output of a classifier for that sample should reflect the ambiguity. That is, the notion of fuzzy set membership of characters is very natural and important in the development of character classifiers to be used in word recognition.

In this paper, we discuss a handwritten word recognition algorithm that uses neural network classifiers on the character level to attempt to read a word. The algorithm is designed to read words that are amenable to segmentation-based approaches; handprinted and well-formed cursive words. We discuss experiments involving the using of assigning desired outputs in the training of the neural networks using a fuzzy k-nearest neighbor algorithm. We compare the use of such networks with crisply trained networks at the character level and at the word level. Our experimental results indicate that the fuzzy output networks do not perform as well on the character level but perform better at the word level.

CHARACTER TRAINING

FEATURES

Currently, a character image is size and skew normalized to size 24 x 16. In the first stage of processing a normalized image is input and a set of eight feature images are generated as output. Each feature image corresponds to one of four 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. An example of the background and foreground feature images corresponding to the east-west directions for an upper-case "B" is shown in Figure 1.

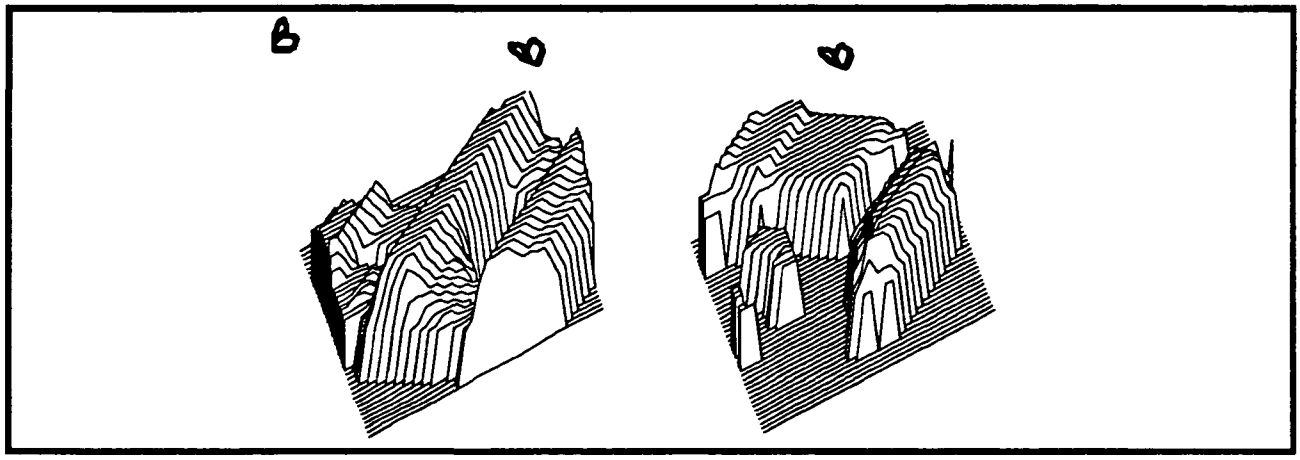

Figure 1. An upper case "B" and the foreground and background bar-feature images corresponding to the east/west directions.

The next stage of processing consists of generating feature vectors from the feature images using the technique of overlapping zones. Fifteen zones are being used; each zone is of size 8 x 8. The zones are maximally overlapping. Zone 1 has its upper left hand corner at position (1,1), zone 2 at position (1,5), ..., zone 4 at position (5,1), etc.

The values in each zone in each feature image are summed. The resulting sums are then normalized between 0 and 1 by dividing by the maximum possible sum in a zone. Thus, the resulting feature vector is of dimension 15 x 8 = 120 and has values between 0 and 1

NETWORK STRUCTURE

We trained separate networks for upper and lower case characters. The networks are four-layered, fully connected, back-propagation networks. Each has input, output, and two hidden layers. Each hidden and output unit has a bias. In this experiment we used 120 input units, 65 unitsfor the first hidden layer, 39 for the second hidden layer, and 26 outputs units. The SAIC neurocomputer [9] was used for training.

COMPUTATION OF DESIRED OUTPUTS

The desired outputs for the crisp networks were set by setting the desired output for the true class to 0.4 and the desired outputs for all other classes to -0.4. The desired outputs for the fuzzy networks were set using a fuzzy k-nearest neighbor algorithm described below.

The fuzzy K-nearest neighbor algorithm we used to assign desired outputs to the characters in the training sets was suggested by Keller et al [10]. The idea is to assign membership based on the percentage of characters in each class among the neighbors of a training sample. Each of our training samples has a true class associated with it, that is, what character the original writer intended to form when writing the character. We do not allow the desired output for the true class to be lower than the desired output for any other class.

We chose to use the twenty nearest neighbors of a training sample using Euclidean distance. The samples

were represented by their feature vectors, thus the distance is being measured in 120-dimensional space. Some examples of desired outputs computed using this Keller's algorithm are shown in Figure 2.

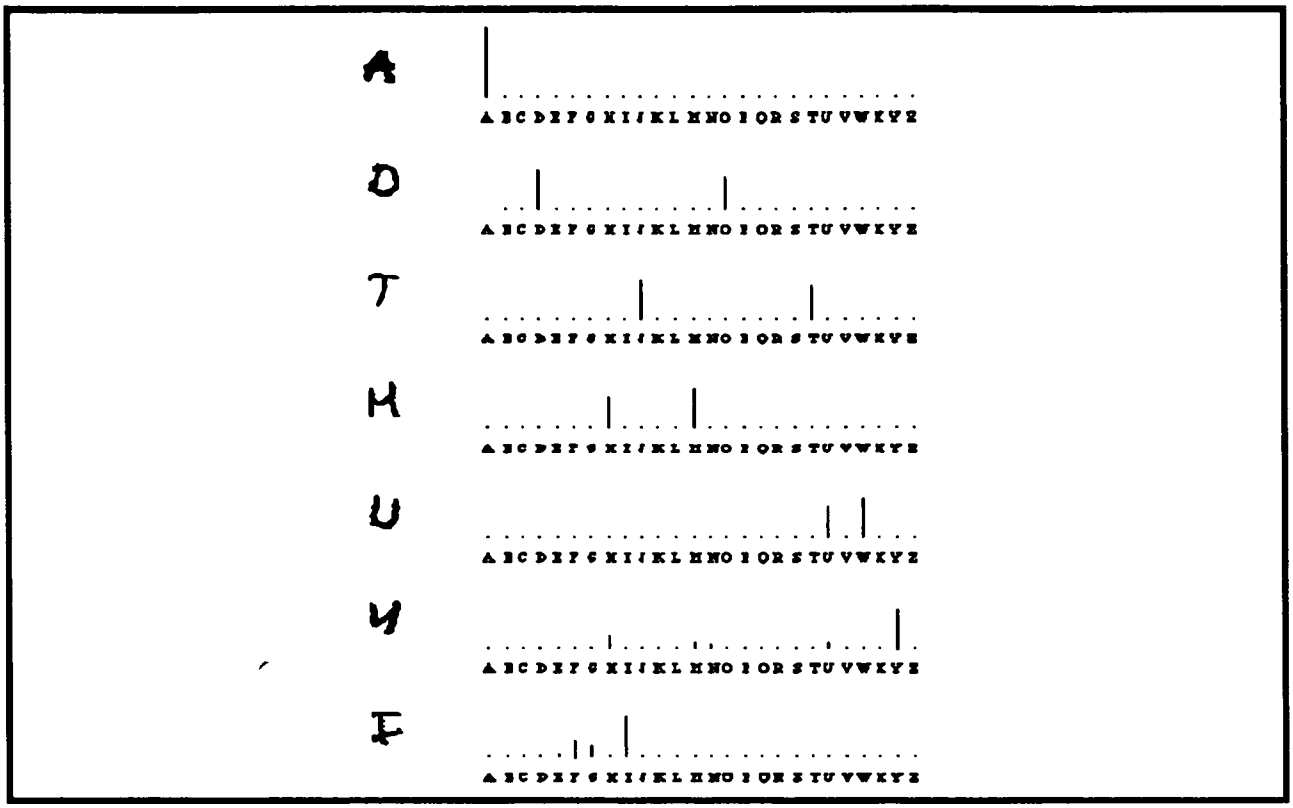

Figure 2. Some uppercase characters and their fuzzy set memberships as determined by the fuzzy k-nearest neighbor algorithm.

TRAINING DATA

Two sources of data were used to construct our training and test sets: characters from the NIST data set and characters from images of handwritten address blocks obtained from the USPS, which we refer to as HWAB data. Both sets of characters were extracted from images using automatic and manual extraction techniques[11].

We are interested in reading words in address blocks and therefore the HWAB data is more important to us. Some classes of characters are not well represented in the HWAB data. For example, we were only able to find seven lower case "j"s in a set of 3000 address blocks. We used the NIST data to fill in the "gaps" in the HWAB data.

Specifically, the training and testing sets for the normalized neural networks consisted of 250 characters from each class. We used as many characters as possible from each class using HWAB data. Thus, if only 300 characters were available from a given class, then we would use 150 in the training set and 150 in the test set. The difference between the number of characters available from the HWAB data and 250 was made up using NIST data. The results are shown in Table 1.

Table 1 HWAB character data correct classification rates.

UPPER CASE	Training Set	Test Set	RMS error	Learning Cycles
Crisp	92.98%	85.91%	0.063959	2158
Fuzzy	88.12%	84.03%	0.050164	1778
LOWER CASE				
Crisp	88.43%	82.15%	0.071576	3255
Fuzzy	86.91%	80.72%	0.050958	2014

At first glance, it may seem that the networks trained with fuzzy outputs are not doing as well. However, several interesting points can be made concerning the above results.

The networks trained with fuzzy outputs converged to lower RMS errors than their crisp counterparts in far fewer learning cycles. Furthermore, the drop in performance was somewhat less for the fuzzy output networks than for the crisp output networks, indicating that the fuzzy output networks may be more robust.

Another view of the results at the character level is given by the sequence of graphs in Figures 3 - 6. We have translated the output values linearly between 0 and 10 and quantized them to integer values. We then constructed histograms of the number correct and incorrect in each bin for the crisp and fuzzy case (Figures 3 and 4) and for the percentage of answers correct and incorrect in each bin (Figures 5 and 6). The number of answers in the high value bins is smaller for the fuzzy case. However, the percentage of answers that are correct in the higher value bins is higher for the fuzzy case. This indicates one can trust answers with higher values more in the fuzzy case.

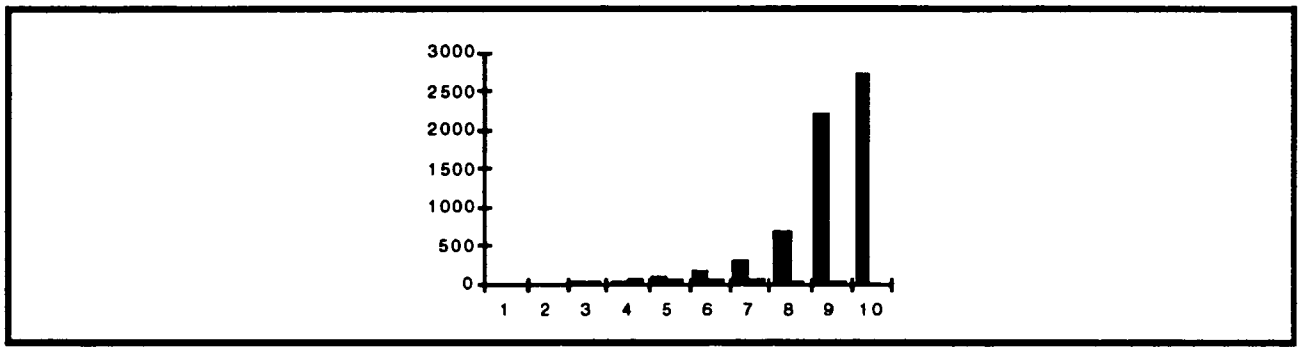

Figure 3. Number Correct/Incorrect by confidence bin - crisp case

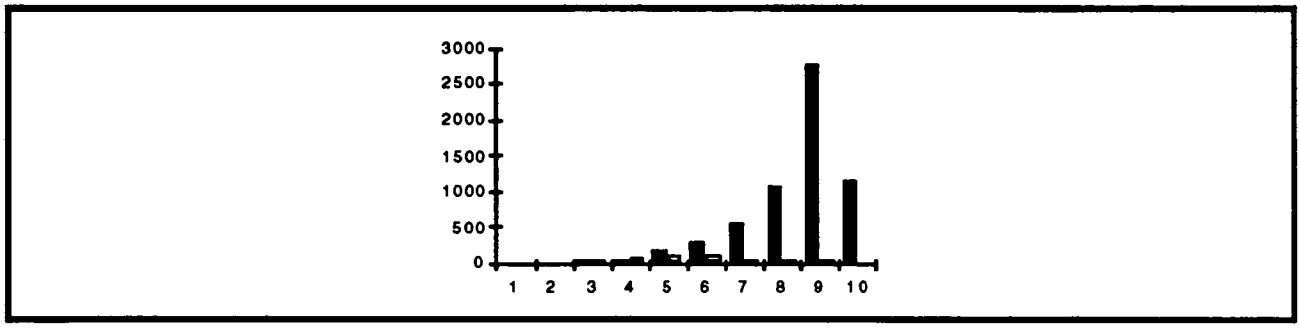

Figure 4. Number Correct/Incorrect by confidence bin - fuzzy case

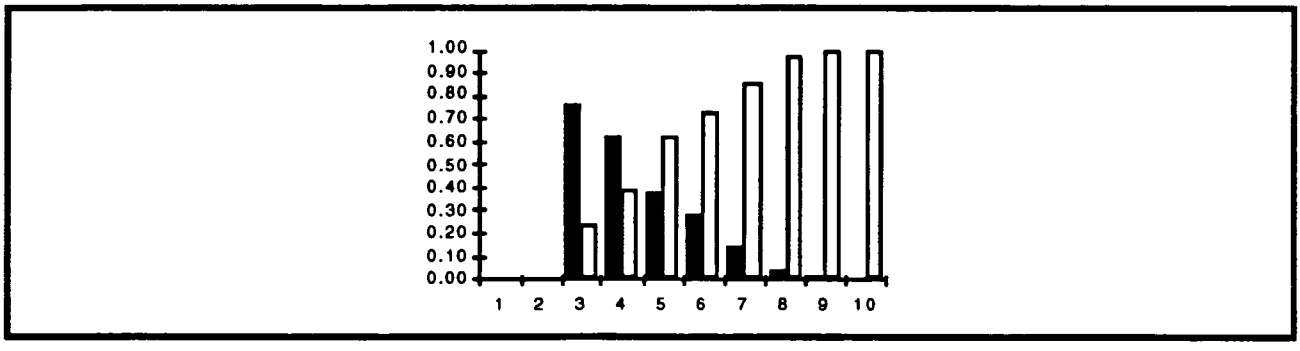

Figure 5. Percent Correct/Incorrect by confidence bin - crisp case

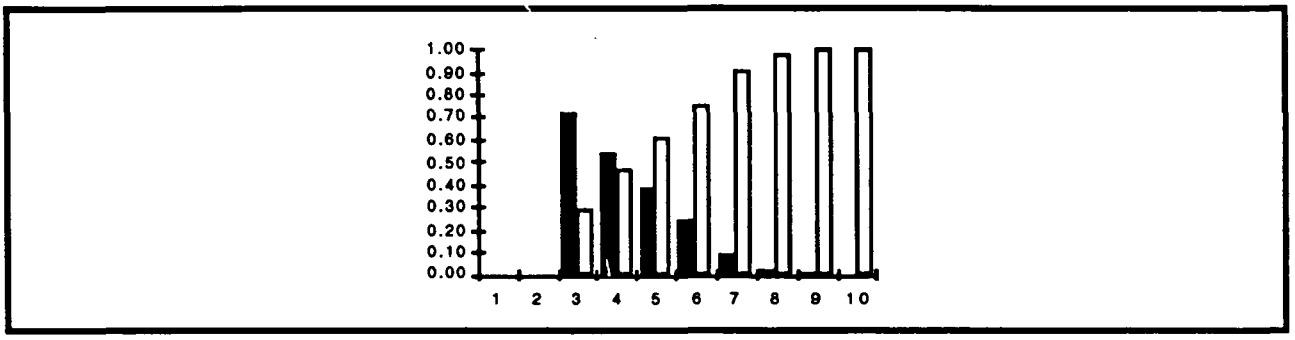

Figure 6. Percent Correct/Incorrect by confidence bin - fuzzy case

WORD RECOGNITION ALGORITHM

OVERVIEW

The word recognition algorithm used is based on image segmentation and dynamic programming matching. The inputs to the algorithm are a binary image of a word and a lexicon. The lexicon is a list of strings representing all possible candidate words for the image.

The approach is based on segmenting the word image into character images, matching the character images against the characters in the word strings in the lexicon, and assigning a confidence to each string in the lexicon based on an aggregation of the confidence of each of the character segments. Unfortunately, it seems impossible to correctly segment a word image into characters without the use of recognition because of the ambiguity of characters and multiple characters mentioned in the introduction above. We therefore need to generate multiple segmentation hypotheses.

The image of the word is segmented into primitive segments. Each primitive segment is generated from a subimage of the original word and ideally consists of either a character or a part of a character. The correct segmentation can be thought of as a path through the space consisting of all primitive segments and their legal unions. Dynamic programming is used to find the best cost path. The cost of a path is currently defined to be the sum of the character confidence of each segment along the path. A more detailed description is given in the following sections. An overview of the system is shown in Figure 7.

As noted in the introduction, this system is being designed to read words that are mainly handprinted; segmentation-based techniques do not seem appropriate for cursive words. Thus, our system contains a module to filter out words that are look too much like cursive words.. The filter is set loosely so that we do process a significant number of cursive words.

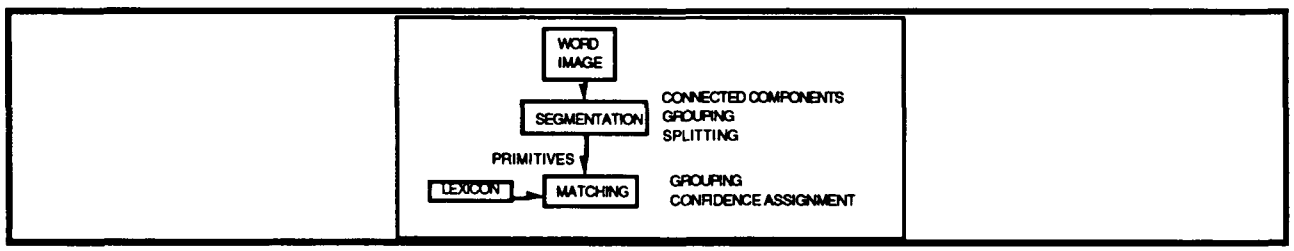

Figure 7. Overview of word recognition system.

SEGMENTATION

The segmentation process is a refinement of that described in [12]. We describe it briefly here. The connected components in the image are computed. Punctuation is detected and removed. Some simple grouping of horizontal bars is performed. The result is an initial segmentation. The initial segments generally consist of images of one or more characters. Those that consist 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 into characters. The distance transform

encodes each pixel in the background using the distance from the stroke. Roughly speaking, split paths are formed that stay as far from the stroke as possible without turning too much. Thus, the distance transform can be thought of as a cost function and the process of splitting one of finding an optimal path. Heuristics are used to define starting points for the paths based on the shape of the image. Heuristics are also used to modify the distance function; for example, holes are encoded as uniformly high cost and "fat" strokes are encoded as low cost.

The result of splitting and initial segmentation is a sequence of subcharacter images which are postprocessed to correct for images that are very small or very complex. This yields the primitive segments as in Figure 8. Unions of primitives are not formed unless required to match strings in the lexicon.

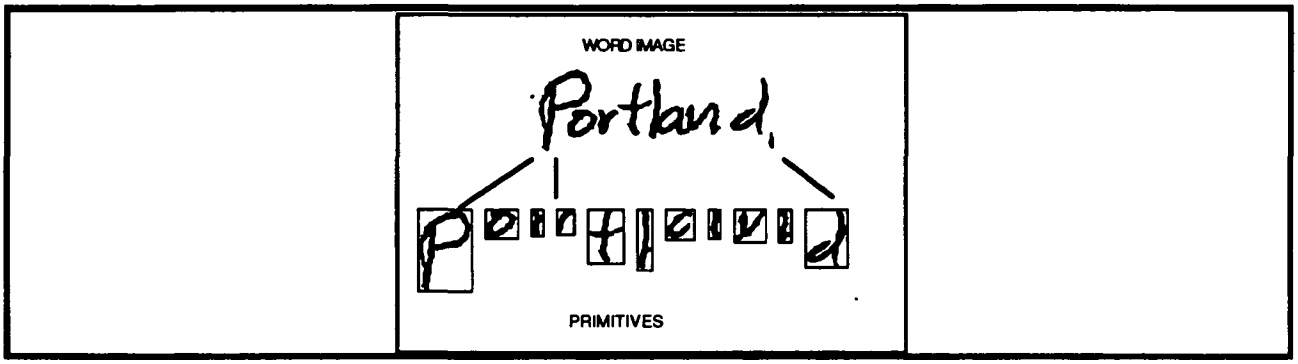

Figure 8. A word image and it's primitive segments.

DYNAMIC PROGRAMMING MATCHING

The core of the dynamic programming algorithm is a module that takes a word image, a string, and a list of the primitives from the word image and returns a confidence value between 0 and 1 that indicates the confidence that the word image represents the string. Dynamic programming is used to find the best path through the space of primitives and legal unions of primitives. The best path depends upon the method to evaluate each node in the path. The value of each node here is currently provided solely by the neural networks described above. The value of a path is computed by averaging the values of the nodes.

The algorithm is implemented using a matrix approach For each string in the lexicon, an array is formed. The rows of the array correspond to the characters in the string. The columns of the array correspond to primitive segments. The ij element in the array is the value of the best match between the first i characters in the string and the first j primitive segments. This value may be $-\infty$ if there is no legal match.

Let the primitive segments of the image be denoted by $S_1, S_2, \ldots, S_p$. Let the characters in the string be denoted by $C_1, C_2, \ldots, C_w$. Let m(c,s) be a function that takes a character c and a segment image s computes the confidence that s represents c. The ij element of the array is computed as follows:

If i = 1, (matching against the first character) Then

$$v(1,j) = m\left(\bigcup_{h=1}^{j} S_h, C_1\right) \qquad \forall\, j \text{ such that } \bigcup_{h=1}^{j} S_h \text{ is legal}$$

$$= -\infty \qquad \text{otherwise}$$

If i > 1 Then

$$v(i,j) = \max_{k}\left(v(i-1,k) + m\left(\bigcup_{h=k}^{j} S_h, C_i\right)\right) \qquad \forall\, k,j \text{ such that } \bigcup_{h=k}^{j} S_h \text{ is legal}$$

$$= -\infty \qquad \text{otherwise}$$

The match value is currently computed by running the upper and lower case neural networks on the segment and retrieving the output value associated with the character for each. Currently the maximum value is taken (except if the character is the first character in a word, in which case a capital letter is much more likely in our application). The neural network can be either the crisp output or the fuzzy output type.

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

WORD TESTING RESULTS

The data used in these experiments were obtained from images of real mail pieces from the United States Postal Service mailstream. They consist of binary images of city and state names.

A test was run on 500 of the images as described above. The lexicon size for our results was 457. The results are shown in tables 2 and 3. The tables require some interpretation. Recall that , since this system is not designed to read all words, there is a check that rejects some words. The response percentage in the tables indicates the percentage of the 500 words that the system decided to process. The word recognition value returns a confidence value between 0 and 1 for each string in the lexicon. We can further decrease the number of responses by thresholding this confidence value.

For each value of a threshold, we compute the number of times the correct string was the top choice, the second highest choice, etc. The rows labeled 0 - 9 indicate these statistics. For example, In the column labeled Thresh = 0.25 of the fuzzy output table, 66.04% of the words for which there was a response were among the top three choices, etc.

Table 2. Results of word recognition with fuzzy output neural network

	Thresh = 0.0	Thresh = 0.25	Thresh = 0.5	Thresh = 0.75	
	response = 57%	response = 53%	response = 39%	response = 27%	
Rank	% at rank	% at rank	% at rank	% at rank	
0	54.74	55.85	69.19	75.37	
1	61.40	62.64	75.14	81.34	
2	64.56	66.04	77.30	83.58	
10	75.09	76.23	84.32	88.06	

Table 3. Results of word recognition with crisp output neural network

	Thresh = 0.0	Thresh = 0.25	Thresh = 0.5	Thresh = 0.75	
	response = 57%	response = 51%	response = 34%	response = 27%	
Rank	% at rank	% at rank	% at rank	% at rank	
0	52.98	57.25	69.01	73.68	
1	62.11	67.06	79.53	82.71	
2	64.91	69.02	79.53	82.71	
10	78.25	81.57	87.72	89.47	

There are several interesting points here. The fuzzy output network was usually higher in top choice and percentage of answers above the thresholds. Thus, the network with fuzzy output values got a higher percentage of answers correct at the top rank and answered on more words at each level than the crisp output network. It was expected that the fuzzy network would yield a higher percentage correct at the top choice. It was not expected that the network would answer more often at the higher confidence values. Also note that the crisp network had a consistently higher percentage among the top ten choices.

The percentage differences are not large between the two networks and the test set is too small to be conclusive. The experiment described here supports the use of the character network using fuzzy output values over that using crisp output values if the ultimate application is word recognition, but not if the application is isolated character recognition.

Examples of correctly and incorrectly read words are shown in figures 9 and 10.

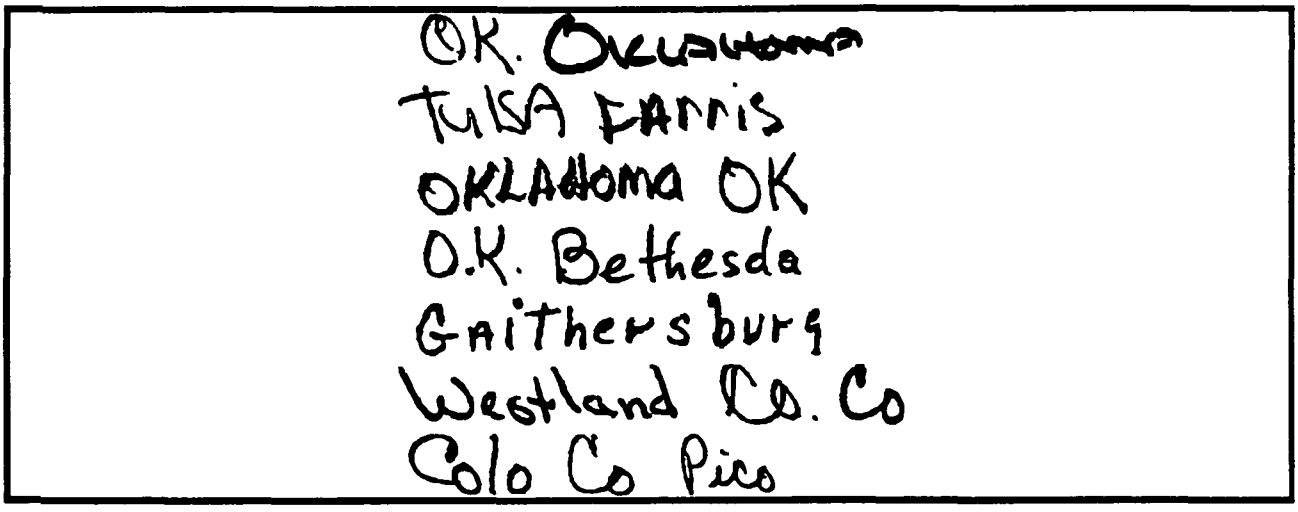

Figure 9. Some words correctly recognized by our system.

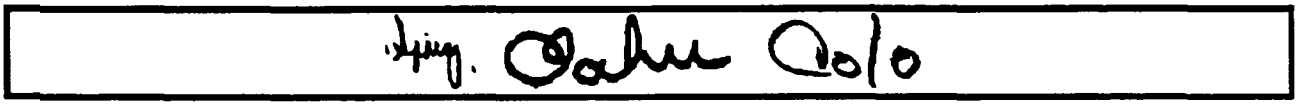

Figure 10. Some words incorrectly recognized by our system.

CONCLUSIONS

We have described an approach to word recognition that relies heavily on the use of neural networks at the character level. We described experiments involving networks trained with crisp output and with fuzzy outputs. The networks with crisp outputs performed better at the character recognition level. The networks with fuzzy outputs performed better at the word recognition level.

ACKNOWLEDGEMENTS

This research was supported by the Environmental Research Institute of Michigan (ERIM) and the United States Postal Service Office of Advanced Technology. Several researchers at ERIM had significant inputs to the word described here. We would like to acknowledge Andrew Gillies, Dan Hepp, Michael Whalen, and Margaret Ganzberger for their contributions to this work.

REFERENCES

[1] *Proceedings of the International Workshop on Frontiers in Handwriting Recognition*, Chateau de Bonas, France, Sept 1991.

[2] *Proceedings of the International Conference on Document Analysis and Recognition*, Saint-Malo France, Oct. 1991.

[3] *Proceedings of the United States Postal Service Advanced Technology Conference*, Washington DC, November 1990.

[4] R. M. Bozinovic and S. N. Srihari, "Off-line Cursive Script Word Recognition", IEEE Trans PAMI, Vol. 11, No. 1, Jan 1989.

[5] T. K. Ho, J. J. Hull, S. N. Srihari, "Word recognition with multi-level contextual knowledge", *Proceedings of the*

International Conference on Document Analysis and Recognition, Saint-Malo France, Oct. 1991, pp. 905 - 915.

[6] T. K. Ho, "A Word Shape Analysis Approach to Recognition of Degraded Word Images", *Proceedings of the United States Postal Service Advanced Technology Conference*, Washington DC, November 1990, pp. 207 - 233.

[7] A. M. Gillies, "Word Verification for the Contextual Analysis of Address Block Images", *Proceedings of the United States Postal Service Advanced Technology Conference*, Washington DC, November 1990, pp. 247 - 255.

[8] R. A. Duderstadt, "Isolated Word Recognition for Postal Address Processing", *Proceedings of the United States Postal Service Advanced Technology Conference*, Washington DC, November 1990, pp. 233 - 247.

[9] Manual for ANSim Version 2.30, Science Applications International Corporation, 1989.

[10] J. M. Keller, M. R. Gray, J. A. Givens, "A Fuzzy K-Nearest Neighbor Algorithms", IEEE Trans. Systems, Man, Cybernetics, Vol SMC-15, No. 4, July -August 1985.

[11] P. D. Gader and M. P. Whalen, "Advanced Research in Handwritten ZIP Code Recognition, Final Technical Report to USPS Office of Advanced Technology, Dec. 1990.

[12] M. P. Whalen, M. Ganzberger, P. D. Gader, "Handprinted Word Recognition II: Segmentation and Verification", submitted to Machine Vision and Applications.