

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

UMI[®]

Distance Measurements and their Combination in Handwritten Character Recognition

Sumeet S. Sawhney

A Major Report

in

The Department

of

Computer Science

**Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Computer Science at
Concordia University
Montreal, Quebec, Canada**

January 2001

© Sumeet S. Sawhney, 2001



**National Library
of Canada**

**Acquisitions and
Bibliographic Services**

**395 Wellington Street
Ottawa ON K1A 0N4
Canada**

**Bibliothèque nationale
du Canada**

**Acquisitions et
services bibliographiques**

**395, rue Wellington
Ottawa ON K1A 0N4
Canada**

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-59339-8

Canada

Abstract

Distance Measurements and their Combination in Handwritten Character Recognition

Sumeet S. Sawhney

Recognition of off-line isolated handwritten English block letters and numerals was done within the constrained domains separately. Distance measurements such as Nearest Neighbor (2), Similarity, Hamming, Linear Correlation, Cross Correlation, Entropy and Information Content were used for recognizing handwritten letters and numerals. 65 prototypes of English block letters and 31 prototypes of numerals were used. Experiments were conducted on over 180 images of letters and 120 images of numerals. In the case of letters it was observed that the results of Nearest Neighbor distance measurement were outstanding and Nearest Neighbor outperforms all other distance measurements. In the case of numerals no distance measurement appeared to be sufficient. In both cases (letters and numerals) Entropy and Information Content does not give any correct results. The thinning algorithm by Zhang & Suen was performed on isolated characters for further improvements. It was observed that thinning with distance measurements has no effect on improving the recognition results.

Since no distance measurement appeared to be sufficient in the case of numerals, a neural network of type Multi Layer Perceptron (MLP) was used to combine the results from 6 distance measurements (Nearest Neighbor (2), Similarity, Hamming, Linear Correlation, Cross Correlation) to obtain a single recognition result. It was observed that the combination enhanced the success rate. Further improvements to MLP results were obtained by using a structural verifier.

Acknowledgements

First and foremost I would like to express my sincere gratitude to my supervisor Dr. Ching Y. Suen for offering me such a wonderful environment and for his assistance at so many levels. I also thank him for suggesting this topic and for carefully reviewing this major report. It is his supervision and support that have made this work possible.

I would like to thank all the friends at CENPARMI Research Center for their help and friendship that enriched my study: Mary, Dong, Danny, Zhang, Yousef, Boulos, Zhou. I also shared nice memory with visitors to CENPARMI, in particular with Dr. Kim Jinho and Dr. Kim Kye Kyung.

Last but not the least, I would also like to express my sincere gratitude to my wife Preety and my parents, who were beside me all the time.

Contents

List of Figures.....	vi
List of Tables	vii
1. Introduction.....	1
1.1 OCR: Motivations	1
1.2 Project Plan	2
2. Implementation	4
2.1 Distance Measurement	5
2.1.1 Similarity Function	5
2.1.2 Hamming Distance	6
2.1.3 Linear Correlation.....	7
2.1.4 Cross Correlation.....	7
2.1.5 Nearest Neighbor.....	7
2.1.6 Information Content and Entropy	9
2.2 Size Normalization	10
2.3 Thinning	10
2.4 Experiments and Results	13
2.4.1 Results for letters.....	13
2.4.2 Results for numerals	17
3. Combining Distance Classifiers.....	20
3.1 Multi Layer Perceptron.....	20
3.2 Network Topology.....	21
3.3 Learning Algorithm.....	23
3.4 Training the Net	27
3.5 Experiments and Results	29
3.6 Further Improvements	31
4. Structural Verification.....	32
4.1 Algorithm	32
4.2 Experiments and Results	33
5. Conclusion	36
References.....	37
Appendices	
A. Prototypes of English Block Letters	38
B. Prototypes of Numerals	39
C. Distance Measurements for Numerals	40
D. Distance Measurements for Letters	41
E. Feature Vectors Used in Training	42

List of Figures

Figure 2.1: Optical Handwritten Character Recognition System	4
Figure 2.2: 8-connectedness topology	10
Figure 2.3: Results of thinning the character 'A'	12
Figure 2.4: Results of distance measurements in case of English block letters	13
Figure 2.5: Results of distance measurements in case of numerals	17
Figure 3.1: Activation functions of conventional perceptron	21
Figure 3.2: A two-layer feedforward neural network architecture	22
Figure 3.3: Performance of training the neural network	27

List of Tables

Table 2.1: Zhang & Suen thinning algorithm	11
Table 2.2: Block letter styles correctly recognized by “Nearest Neighbor-2”	16
Table 2.3: Block letter styles misclassified by “Nearest Neighbor-2”	16
Table 2.4: Numeral styles misclassified by “Nearest Neighbor-2”, correctly classified by others	18
Table 3.1: Notations used in error backpropagation training algorithm	23
Table 3.2: Batch Learning with error backpropagation	26
Table 3.3: Results obtained with combination	29
Table 3.4: Improvements obtained with combination	29
Table 3.5: Numeral styles correctly recognized by MLP	30
Table 3.6: Numeral styles misclassified by MLP	31
Table 4.1: Algorithm for structural verification of numerals ‘3’ & ‘8’	33
Table 4.2: Results obtained with structural verification of numerals ‘3’ & ‘8’	34
Table 4.3: Numerals ‘3’ & ‘8’ correctly verified	34
Table 4.4: Wrong results after the verification stage	34

Chapter 1

Introduction

1.1 OCR: Motivations

Character recognition techniques associate a symbolic identity with the image of a character. This problem of replication of human functions by machines involves the recognition of both machine printed and handprinted/cursive-written characters.

Character recognition is better known as optical character recognition (OCR) since it deals with the recognition of optically processed characters rather than magnetically processed ones. Though the origin of character recognition can be found as early as 1870, it first appeared as an aid to the visually handicapped and the first successful attempt was made by Russian scientist Tyurin in 1900 [3]. The modern version of OCR appeared in the middle of 1940s with the development of digital computers. Thenceforth it was realized as a data processing approach with application to the business world. OCR is, in a broad sense, a branch of artificial intelligence and it is also a branch of computer vision. OCR machines have been commercially available since the middle of the 1950s.

In most existing OCR systems, character recognition performs on individual characters. The objective of character recognition is to interpret input as a sequence of characters taken from a given set of characters.

The following are some of the applications [2, 3] for which OCR have been used:

- In Postal department – as a reader for printed/handwritten postal codes.
- In Motor vehicle bureau – as automatic number plate reader.
- For business applications – like cheque sorting.
- For direct processing of documents – as a multipurpose document reader for large-scale data processing.
- Signature verification.
- Writer identification.
- For use in customer billing as in telephone exchange billing system.
- In educational administrations – examination assessment and attendance record evaluation, and as a mark sheet reader.

On the basis of the nature of applications character recognition is grouped into two schemes, off-line and on-line character recognition [2]. In off-line systems the recognition is done at the time of preparing the documents, whereas in on-line the recognition is done as and when characters are drawn, and hence timing information of each stroke is also available with the character images.

1.2 Project Plan

The focus of our work was recognition of off-line isolated constrained handwritten English block letters and numerals separately. Constrained characters are characters that respect predefined prototypes.

Project Layout:

- Create prototypes of handwritten *English block letters*.

- Recognize the input letter using various distance measurements with prototypes.
- Check for improvements using thinning and distance measurements with thinned prototypes.
- Create prototypes of *handwritten numerals*.
- Recognize the input numeral using various distance measurements with prototypes.
- Check for improvements using thinning and distance measurements with thinned prototypes.
- Possible combination of distance measurements using Neural Network.
- Further improvements/Structural verification.

Chapter 2

Implementation

The design used in our off-line handwritten recognition system is outlined by a block diagram in Figure 2.1

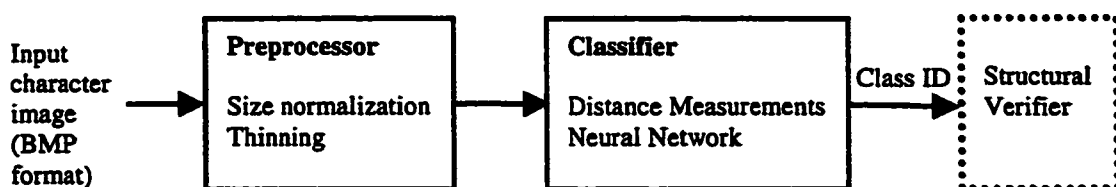


Figure 2.1: Optical Handwritten Character Recognition System

Input images in BMP¹ file format [10] were considered. The character image is fed to the preprocessor.

The “Preprocessor” is used to prepare the raw input character image for recognition purposes. In our work we have used two preprocessing techniques: size normalization and thinning, well explained in Section 2.2 and 2.3 respectively.

“Classifier” is used to assign a specific class to the input character. Classifiers used in our project are of type binary matching, using different distance measurements, and neural network. Binary matching as discussed in Section 2.1 is a procedure of matching the input character to standard set of prototypes. 65 prototypes of English block letters (*Appendix A*) and 31 prototypes of numerals (*Appendix B*) were used.

¹ BMP images are stored upside down

Another type of classifier “Neural Network” discussed in Chapter 3 is used to combine various distance measurements to improve recognition results.

Structural Verifier discussed in Chapter 4 is used to further improve the recognition results of neural network.

2.1 Distance Measurement

The procedure [4, 6, 7] consists of defining measures of similarity/dissimilarity between the unlabeled target and a class of specified or labeled prototypes (*Appendix A and B*). The unlabeled target is then labeled according to whether or not the measure satisfies a specified criterion.

The objective is to assign an unknown target or object Y into one of C classes or populations that are represented by the specified templates (prototypes) $\{X_L\}$ for $L = 1, 2, \dots, C$.

In the below subsections m and n represents the number of rows and columns of the character respectively.

2.1.1 Similarity Function

The similarity function $S(Y,X)$ was used to measure the number of matrix cells “occupied” by both models Y and X . It is given by the following formula:

$$S(Y,X) = \sum_{i=1}^m \sum_{j=1}^n Y_{ij} \cdot \text{AND} \cdot X_{ij}$$

where

$$Y_{ij} \cdot \text{AND} \cdot X_{ij} = \begin{cases} 1, & \text{if the } ij\text{th cell is "occupied" by} \\ & \text{both models Y and X,} \\ 0, & \text{otherwise.} \end{cases}$$

The smaller the value of $S(Y,X)$ is, the less the “common area” is shared by models Y and X, and therefore the less the degree of “similarity” between the two models. Thus largest value over entire population is preferred.

2.1.2 Hamming Distance

The hamming distance $H(Y, X)$ was used to measure the number of different cells occupied by the two models Y and X. It is given by the following formula:

$$H(Y, X) = \sum_{i=1}^m \sum_{j=1}^n Y_{ij} \cdot \text{XOR} \cdot X_{ij}$$

where

$$Y_{ij} \cdot \text{XOR} \cdot X_{ij} = \begin{cases} 1, & \text{if the } ij\text{th cell is "occupied" by one model and not by} \\ & \text{the other of the two models Y and X,} \\ 0, & \text{otherwise.} \end{cases}$$

The larger the value of $H(Y, X)$ is, the greater the “difference” between the models Y and X. Thus smallest value over entire population is preferred.

2.1.3 Linear Correlation

Taking into considerations the various degrees of misalignment and stroke width variations of models, the similarity function $S(Y, X)$ was modified to obtain linear correlation measurement. It is given by the following formula:

$$LC(Y, X) = 2 * [S(Y, X) / (N_Y + N_X)]$$

Where N_Y and N_X are numbers of cells occupied by models Y and X, respectively. A smaller value of $LC(Y, X)$ indicates that the normalized common area shared by model Y and X is smaller. Thus largest value over entire population is preferred.

2.1.4 Cross Correlation

Taking into considerations the various degrees of misalignment and stroke width variations of models, the similarity function $S(Y, X)$ was modified to obtain cross correlation measurement. It is given by the following formula:

$$CC(Y, X) = [S(Y, X)]^2 / (N_Y * N_X)$$

Where N_Y and N_X are numbers of cells occupied by models Y and X, respectively. A smaller value of $CC(Y, X)$ indicates that the normalized common area shared by model Y and X is smaller. Thus largest value over entire population is preferred.

2.1.5 Nearest Neighbor

The “nearest cell distance” $d(Y_{ij}, X)$ was used to measure the “distance” between ij th cell of model Y and the nearest cell occupied by model X. It is given by the following formula:

$$d(Y_{ij}, X) = \begin{cases} 0, & \text{if } Y_{ij} = 0, \\ \min_{\substack{1 \leq m \leq \max_cols \\ 1 \leq n \leq \max_rows}} [(m-i)^2 + (n-j)^2 \mid X_{mn} \neq 0], & \text{if } Y_{ij} \neq 0. \end{cases}$$

A larger value of $d(Y_{ij}, X)$ indicates a larger “distance” between the cell Y_{ij} and the “nearest cell” occupied by the model X .

For any pair of model Y and X , two measurements were used to indicate the difference between the pair. They are given by the equations:

Nearest Neighbor-1:

$$ND1(Y, X) = \frac{1}{N_Y} \sum_{i=1}^m \sum_{j=1}^n [d(Y_{ij}, X)]^{1/2} + \frac{1}{N_X} \sum_{i=1}^m \sum_{j=1}^n [d(X_{ij}, Y)]^{1/2}$$

and

Nearest Neighbor-2:

$$ND2(Y, X) = \left(\sum_{i=1}^m \sum_{j=1}^n d(Y_{ij}, X) / N_Y + \sum_{i=1}^m \sum_{j=1}^n d(X_{ij}, Y) / N_X \right)^{1/2}$$

Where N_Y and N_X are numbers of cells occupied by models Y and X , respectively. A larger value of “nearest-neighbor distance-1” $ND1(Y, X)$ or the “nearest-neighbor distance-2” $ND2(Y, X)$, indicates that the “cell difference” between the models Y and X is greater. Thus smallest value over entire population is preferred.

2.1.6 Information Content and Entropy

The information content and entropy measurements are given by the following equations:

$$\text{INF}(Y) = \sum_{i=1}^m \sum_{j=1}^n (I_{ij} \cdot Y_{ij})$$

and

$$\text{ENT}(Y) = \sum_{i=1}^m \sum_{j=1}^n (P_{ij} \cdot I_{ij} \cdot Y_{ij})$$

where P_{ij} is the probability of the ij th cell being occupied by all models (*see Appendices A and B*) in the set.

$$I_{ij} = -\log_2 P_{ij}$$

$$Y_{ij} = \begin{cases} 1, & \text{if the } ij\text{th cell is "occupied" by model } Y, \\ 0, & \text{otherwise.} \end{cases}$$

The information content $\text{INF}(Y)$ is a measure of the information carried by cells of model Y based on the distribution of the entire set of models and the entropy $\text{ENT}(Y)$ is a measure of the average information content carried by model Y . Over entire population the one with smallest value of INF or ENT is preferred.

2.2 Size Normalization

The purpose of size normalization is to make the size of the input character image equal to the size of prototype image, in order to facilitate distance measurements.

For size normalization from original image of width w and height h to new image of width w' and height h' , the following transformations on each black pixel (x, y) is used:

$$x' = (w' / w) * x$$

$$y' = (h' / h) * y$$

2.3 Thinning

The input pattern is thinned down to a “skeleton” of unitary thickness.

Zhang & Suen [5] is a fast parallel thinning algorithm which tries to bring the skeleton to the center of the image by peeling off the external and internal pixels.

For each pixel, its 8 neighbors are taken into account.

$P_8(i-1, j-1)$	$P_1(i-1, j)$	$P_2(i-1, j+1)$
$P_7(i, j-1)$	$P_0(i, j)$	$P_3(i, j+1)$
$P_6(i+1, j-1)$	$P_5(i+1, j)$	$P_4(i+1, j+1)$

Figure 2.2: 8-connectedness topology

P_0 is the pixel to be considered for deletion. The Zhang & Suen thinning algorithm is well explained in Table 2.1. An example of thinning implementation is depicted in Figure 2.3

Each iteration consists of two subiterations:

Subiteration 1: It deletes south-east boundary points and the north-west corner points.

- Get pixels to be removed if it satisfies the following conditions:

1. $2 \leq B(P_0) \leq 6$

2. $A(P_0) = 1$

3. $P_1 * P_3 * P_5 = 0$

4. $P_3 * P_5 * P_7 = 0$

$B(P_0)$ is the nonzero neighbors of P_0 , that is

$$B(P_0) = P_1 + P_2 + P_3 + P_4 + P_5 + P_6 + P_7 + P_8$$

$A(P_0)$ is the number of 0 to 1 transitions in the ordered set

$P_1, P_2, \dots, P_8, P_1$

- Make changes to the original image.

Subiteration 2: It deletes north-west boundary points and south-east corner points.

- Get pixels to be removed, if the following conditions are satisfied:

Change conditions 3 & 4 of the above subiteration to:

3' $P_1 * P_3 * P_7 = 0$

4' $P_1 * P_5 * P_7 = 0$

- Make changes to the original image.

Continue the above iterations (Subiteration 1 and 2) until there is no change.

Table 2.1: Zhang & Suen thinning algorithm

(a) Before Thinning

(b) After Thinning

2.4 Experiments and Results

Experiments were conducted on over 180 isolated images of English block letters and 120 isolated images of numerals generated by the mouse. The images were normalized to (30 x 25) for recognition purposes. Distance measurements were done with 65 English block letter prototypes (*Appendix A*) and 31 numeric prototypes (*Appendix B*). Images in the test data were 'closer' to and 'farther' from the prototypes in proportionate order.

2.4.1 Results for letters

The results obtained by different distance measurements are shown in Figure 2.4

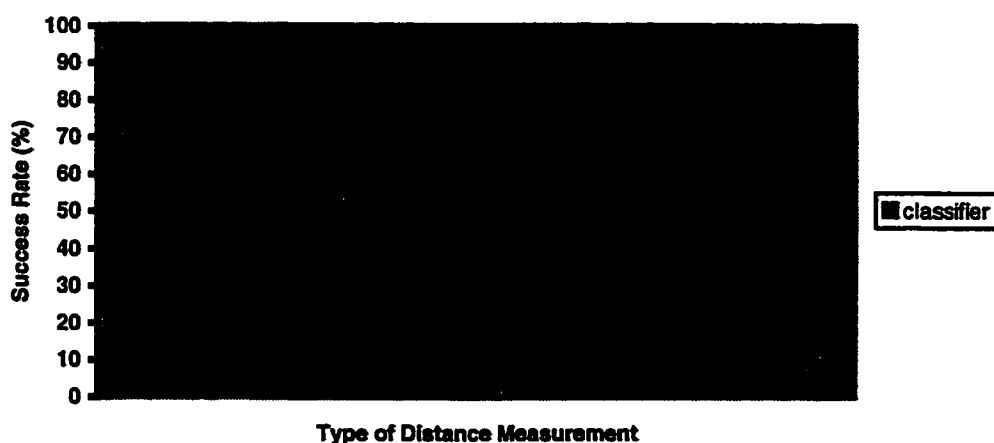


Figure 2.4: Results of distance measurements in case of English block letters

In order to improve the results, recognition was done using *thinned* input images and prototypes. No improvements were observed, rather it gave wrong results to some of the inputs correctly recognized without thinning. This is because thinning is very sensitive to noise.

It was observed that in case of English block letters “Nearest Neighbor-2” distance measurement (labeled as ‘b’) is most powerful amongst all other distance measurements with success rate of 95.58 %. The letters misclassified by “Nearest Neighbor-2” were also misclassified by all other distance measurements. Therefore, “Nearest Neighbor-2” is the only best choice amongst others.

Since “Nearest Neighbor-2” distance measurement has excellent results and outperforms all others, we have shown its performance over English block letters in Tables 2.2 and 2.3

Handwriting Practice: Uppercase Letters A-Z							
A	A	A	A	A	A	A	A
B	B	B	B	B	B	B	B
C	C	C	C	C	C	C	C
D	D	D	D	D	D	D	D
E	E	E	E	E			
F	F	F	F				
G	G	G	G	G	G	G	G
H	H	H	H	H	H	H	
I	I	I	I	I			
J	J	J	J	J	J	J	J
K	K	K	K	K	K	K	
L	L						
M	M	M	M	M	M	M	M
N	N	N	N	N	N	N	N
O	O	O	O	O			
P	P	P	P				

STYLES CORRECTLY RECOGNIZED BY "NEAREST NEIGHBOR-2"							
Q	Q	Q	Q				
R	R	R					
S	S	S	S	S	S		
T	T						
U	U	U	U	U			
V	V	V	V	V	V	V	
W	W	W	W	W	W	W	W
X	X	X	X	X	X	X	
Y	Y	Y	Y	Y			
Z	Z	Z	Z	Z	Z	Z	

Table 2.2: Block letter styles correctly recognized by "Nearest Neighbor-2"

STYLES MISCLASSIFIED BY "NEAREST NEIGHBOR-2"		
Input Image	Written As	Classified As
D	O	D
K	K	H
V	V	Y

Table 2.3: Block letter styles misclassified by "Nearest Neighbor-2"

From Table 2.3 it can be observed that “Nearest Neighbor-2” does not give good results for very confusing structures (in the given images, the structure of ‘O’ is confused with ‘D’, ‘K’ is confused with ‘H’ and ‘V’ is confused with ‘Y’).

The best matched measurement values between some inputted English block letter styles and the prototypes are shown in *Appendix D*.

2.4.2 Results for numerals

The results obtained by different distance measurements are shown in Figure 2.5

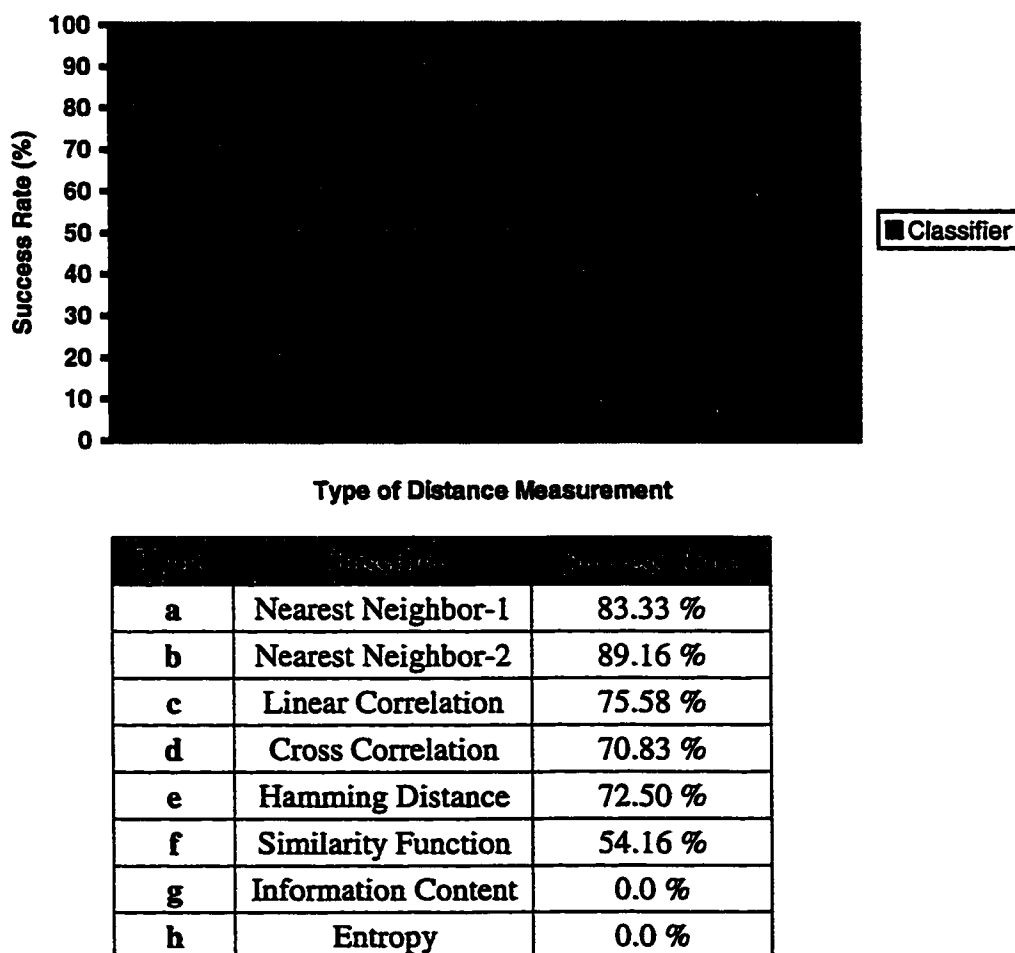


Figure 2.5: Results of distance measurements in case of numerals

It was observed that in case of numerals no single distance measurement appeared to be sufficient. “Linear Correlation”, “Hamming” and “Similarity” correctly recognized some of the numerals misclassified by “Nearest Neighbor-2” as shown in Table 2.4.

Input Image	Written As	Output	Correctly Classified By
1	1	9	Hamming
6	6	8	(Linear, Cross) Correlation and Hamming
3	3	8	Hamming
8	8	3	Similarity

Table 2.4: Numeral styles misclassified by “Nearest Neighbor-2”, correctly classified by others

In order to improve the results, recognition was done using *thinned* input images and prototypes. No improvements were observed, rather it gave wrong results to some of the inputs correctly recognized without thinning. This is because thinning is very sensitive to noise.

The best matched measurement values between some inputted numeral styles and the prototypes are shown in *Appendix C*.

Since some numerals misclassified by “Nearest Neighbor-2” were correctly classified by other distance measurements, we decided to combine 6 distance classifiers (Nearest Neighbor-1, Nearest Neighbor-2, Linear Correlation, Cross Correlation, Hamming Distance, and Similarity function) using neural network to get single recognition result with improved success rate.

Chapter 3

Combining Distance Classifiers

To enhance the recognition of patterns, many researchers have suggested that straightforward single methods are inadequate for complex problem such as handwritten numeral recognition, while combined methods can offer a better recognition performance [1].

Neural Networks are good combiner, hence we used it in case of numerals to combine 6 distance classifiers (Nearest Neighbor-1, Nearest Neighbor-2, Linear Correlation, Cross Correlation, Hamming Distance, and Similarity function). Misclassifications from distance measurements can be trained to get correct results, thus increasing the success rate.

3.1 Multi Layer Perceptron

Work on artificial neural networks began more than 50 years ago with the efforts of McCulloch and Pitts, Hebb, Rosenblatt. More recent work by Hopfield, Rumelhart and McClelland, Feldman and others has led new interests to the field. Good reviews of various artificial neural networks can be found in [8, 9].

Among various neural net models, Multilayer Perceptron (MLP) is the most widely used, especially in the problem of pattern classification.

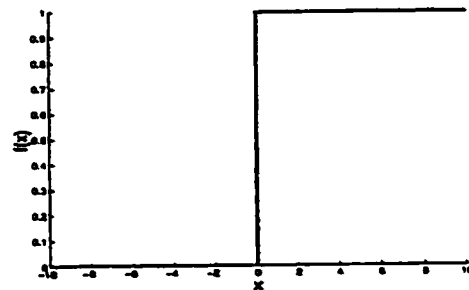
3.2 Network Topology

The perceptron [8, 9] was conceived by Rosenblatt in 1959. The perceptron, as the building block of MLP network, forms a weighted sum of n components of the input vector and adds a bias value, θ . The result is then passed through a nonlinearity.

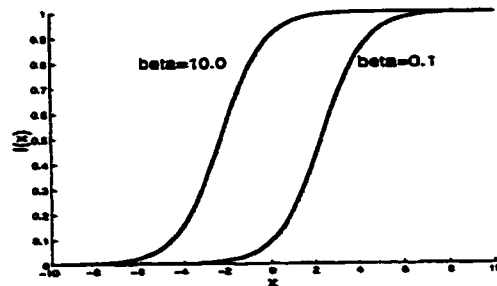
Rosenblatt's original model used the hard-limiting nonlinearity Figure 3.1 (a). When perceptrons are cascaded together in layers, it is more common to use the sigmoid nonlinearity:

$$f(x) = (1 + e^{-\beta x})^{-1} \quad (3.1)$$

where slope factor β determines the steepness of the transition region Figure 3.1 (b). One of the advantages of the sigmoid compared with hard-limiting is that it is differentiable, which makes it possible to derive a gradient search learning algorithm for the multilayer network.



(a) Hard-limiting



(b) Sigmoid function

Figure 3.1: Activation functions of conventional perceptron

The power of single neuron can be greatly amplified by using multiple neurons in a network of layered connectionist architecture, as displayed in Figure 3.2.

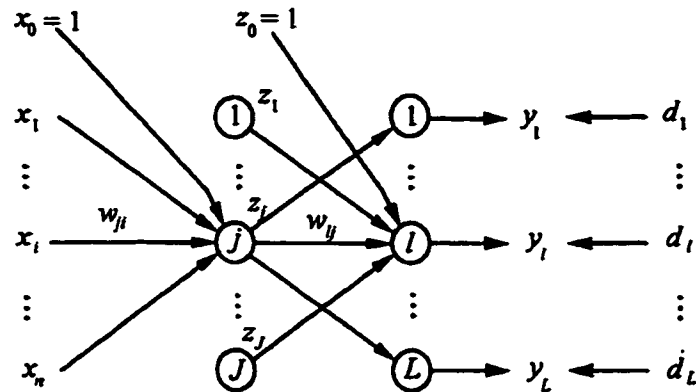


Figure 3.2: A two-layer feedforward neural network architecture

Such a *Multiple Layered Perceptron* (MLP) is also called a *Feedforward Artificial Neural Network* and abbreviated to FANN. The modifier “feedforward” distinguishes it from feedback (recursive) networks. On the left is the layer of inputs, or *branching*, nodes, which are not artificial neurons. The *hidden layer* (the middle layer here) contains neural nodes, as does the *output layer* on the right. x_0 and z_0 are the added bias. This is the *architecture* of two-layered NN (so called because there are two layers of neural units).

3.3 Learning Algorithm

Error backpropagation (or *backprop*) [8, 9], is one of the most frequently used learning rules in many applications of artificial neural networks. Backprop provides a computationally efficient method for changing the weights in a feedforward network, with differentiable activation function units, to learn from a training set of input-output examples. Backpropagation is a gradient-descent search algorithm.

With the notations as in Table 3.1, as well as the one hidden layer topology of Figure 3.2 as the illustrative example, we can derive the learning algorithm:

w_{lj}	Connecting weight between l th node in output layer and j th node in hidden layer
w_{ji}	Connecting weight between j th node in hidden layer and i th node in input layer
d_l	Desired output of the l th output node
y_l	Real output of l th output node
z_j	Output of j th hidden layer node
x_i	i th component for input sample x
n	Number of input layer nodes
J	Number of hidden layer nodes
L	Number of output layer nodes
m	Number of exemplar pairs

Table 3.1: Notations used in error backpropagation training algorithm

Activation function f_h of the hidden nodes and f_o of the output nodes is assumed to be a differentiable nonlinear function, see Equation (3.1)

Next, consider a set of m input/output pairs $\{\mathbf{x}^k, \mathbf{d}^k\}$, where \mathbf{d}^k is an L -dimensional vector representing the desired network output upon presentation of \mathbf{x}^k . The objective here is to adaptively adjust the $J(n + 1) + L(J + 1)$ weights of this network such that the underlying function/mapping represented by the training set is approximated or learned. Since the learning here is supervised (i.e., target outputs are available), an error function may be defined to measure the degree of approximation for any given setting of the network's weight. A commonly used error function is the SSE measure.

If a differentiable criterion function is used, gradient descent on such a function will naturally lead to a learning rule. This idea was invented independently by Amari (1967, 1968), Bryson and Ho (1969), Werbos (1974), and Parker (1985). Next, this idea is illustrated by deriving a supervised learning rule for adjusting the weights, w_{ji} and w_{lj} such that the following error function is minimized (in a local sense) over the training set:

$$E(\mathbf{w}) = \frac{1}{2} \sum_{l=1}^L (d_l - y_l)^2 \quad (3.2)$$

Here \mathbf{w} represents the set of all weights in the network.

Since the targets for the outputs units are explicitly specified, one can use the delta rule directly, for updating the w_{lj} weights. That is,

$$\Delta w_{lj}(t) = -\rho_o \frac{\partial E}{\partial w_{lj}(t)} + \alpha \Delta w_{lj}(t-1) = \rho_o (d_l - y_l) f_o'(\text{net}_l) z_j + \alpha \Delta w_{lj}(t-1) \quad (3.3)$$

with $l = 1, 2, \dots, L$ and $j = 0, 1, \dots, J$. Here $\text{net}_l = \sum_{j=0}^J w_{lj} z_j$ is the weighted sum for

the l th output unit, f_o' is the derivative of f_o with respect to net , ρ_o is the learning parameter. The z_j values are computed by propagating the input vector x through the hidden layer according to

$$z_j = f_h \left(\sum_{i=0}^n w_{ji} x_i \right) = f_h(net_j) \quad j = 1, 2, \dots, J$$

$\alpha \Delta w_{ji}(t-1)$ is the momentum to gradient search. Momentum accelerates the convergence where, t is iteration step, and α is a momentum rate normally chosen between 0 and 1. The learning rule for the hidden-layer weights, w_{ji} is not as obvious as that for the output layer because we do not have available as set of target values (desired outputs) for hidden units. However, one may derive a learning rule for hidden units by attempting to minimize the output-layer. This amounts to propagating the output errors $(d_l - y_l)$ back through the output layer toward the hidden units in an attempt to estimate “dynamic” targets for these units. Gradient descent is performed on the criterion function in Equation (3.2), but this time gradient is calculated with respect to the hidden weights:

$$\Delta w_{ji}(t) = -\rho_h \frac{\partial E}{\partial w_{ji}(t)} + \alpha \Delta w_{ji}(t-1) \quad j=1, 2, \dots, J; \quad i=0, 1, 2, \dots, n$$

using chain rule for differentiation we get:

$$\Delta w_{ji}(t) = \rho_h \left(\sum_{l=1}^L (d_l - y_l) f_o'(net_l) w_{lj} \right) f_h'(net_j) x_i + \alpha \Delta w_{ji}(t-1) \quad (3.4)$$

$\alpha \Delta w_{ji}(t-1)$ is the momentum to gradient search, ρ_h is a learning rate parameter. The complete procedure for updating the weights in a feedforward neural net utilizing these rules is summarized in Table 3.2

Initialize all weights and refer to them as “current” weights w_{ij}^c, w_{ji}^c .

- Set the learning rates ρ_o and ρ_h to small positive values.

Repeat

For $q = 1, \dots, m$

- Select q th input pattern \mathbf{x}^k from the training set and propagate it through the network, thus generating hidden and output unit activities based on the current weight settings.
- Use the desired target \mathbf{d}^k associated with \mathbf{x}^k , and employ Equation (3.3) to compute the output layer weight changes $\Delta w_{ij}(t)$.
- Employ Equation (3.4) to compute the hidden-layer weight changes $\Delta w_{ji}(t)$.
- Update all weights according to $w_{ij}^c = w_{ij}^c + \Delta w_{ij}(t)$
and $w_{ji}^c = w_{ji}^c + \Delta w_{ji}(t)$

End

/* Test for convergence */

Calculate Root-Mean-Square (RMS) error given by $\sqrt{\frac{2E}{mL}}$ and using weights w_{ij}^c and w_{ji}^c

$$\text{where } E = \frac{1}{2} \sum_{k=1}^m \sum_{l=1}^L (d_l - y_l)^2$$

Until the termination condition is reached.

Table 3.2: Batch learning with error backpropagation

3.4 Training the Net

It was observed that it's not possible to train the net using the actual measurement values. The measurement values obtained from 6 distance measurements (Nearest Neighbor-1, Nearest Neighbor-2, Linear Correlation, Cross Correlation, Hamming Distance and Similarity function) for the various numeral styles overlapped and lie nearly in the same domain (*Appendix C*), this made it difficult to train the net. So we decided to train the net using the recognition results (0,...,9) obtained from 6 distance classifiers and normalizing them to (0,...,1). The advantage of supervised training is to map the wrong results of distance classifiers to correct ones.

After thoroughly studying various numeral styles and their recognition results from 6 distance classifiers, we came up with 126 exemplar feature vectors, (*Appendix E*). The performance of training the net using exemplar feature vectors and test data set of 120 numerals is shown in Figure 3.3

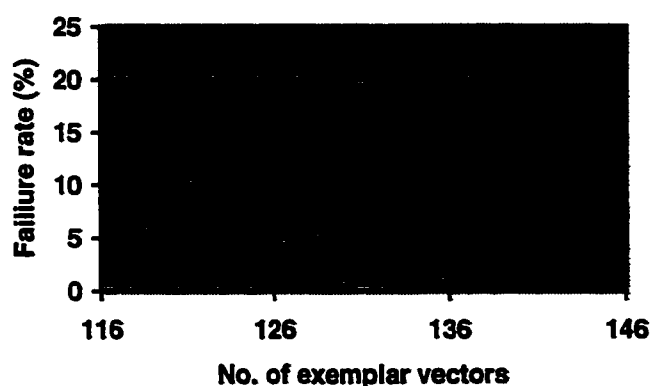


Figure 3.3: Performance of training the neural network

Using 126 exemplar feature vectors showed minimum failure rate. Increase in the number of exemplar vectors, increases the failure rate, this is because feature vectors

appeared to be very near and thus error backpropagation learning rule finds it very difficult to approximate them to the desired targets.

With 126 exemplar feature vectors MLP was tuned to 7-20-10, with 7 input nodes (including bias and 6 distance classifiers), 20 hidden nodes (including bias) and 10 output nodes (0 – 9). The representation of the input nodes and the output nodes is shown in *Appendix E*.

Parameters used in training the net:

Learning parameters ρ_o and ρ_h set to 1.0 & 0.5 respectively.

Momentum $\alpha = 0.3$

Activation function f_h of the hidden nodes and f_o of the output nodes set to Equation (3.1), with slope factor $\beta = 1$

Root Mean Square error (RMS) = 0.05

3.5 Experiments and Results

Experiments were conducted on test data set of 120 isolated images of numerals generated by the mouse. The images were normalized to (30 x 25) for recognition purposes. The recognition results (0,...,9) obtained from of 6 distance classifiers (Nearest Neighbor-1, Nearest Neighbor-2, Linear Correlation, Cross Correlation, Similarity, and Hamming) were normalized to (0,...,1) and were fed to the neural network (MLP) to obtain the final recognition result.

The results obtained by using Neural Network are shown in Table 3.3

Classifier	Recognition Rate
Neural Network	95 %

Table 3.3: Results obtained with combination

Combination of distance measurements resulted in increased performance over their individual counterpart. Improvements obtained with combination are shown in Table 3.4

Input Image	Written As	Classified As
1	1	1
6	6	6
8	8	8

Table 3.4: Improvements obtained with combination

Numeral styles correctly recognized by MLP are shown in Table 3.5

1	1	1	1	1	1	
2	2	2	2	2	2	2
3	3	3	3	3	3	3
4	4	4	4	4	4	4
5	5	5	5	5	5	5
6	6	6	6	6	6	6
7	7	7	7	7	7	7
8	8	8	8	8	8	8
9	9	9	9	9	9	9
0	0	0	0	0	0	0

Table 3.5: Numeral styles correctly recognized by MLP

Numeral styles misclassified by MLP are shown in Table 3.6

Input Image	Written As	Classified As
3	3	8
3	3	8
3	3	8
1	7	1
8	8	3

Table 3.6: Numeral styles misclassified by MLP

3.6 Further Improvements

From Table 3.6 it was observed that a style of '7' is misclassified as '1'. This style of '7' is confused with '1' by human mind too. It was also observed that numerals ('3' and '8') were mostly misclassified to one another. Structurally they differ only by a long tail. It is difficult for neural network (MLP) to approximate them to the correct target because the feature vectors were very near.

The misclassification of ('3' and '8') by MLP can be improved by doing *structural verification* at the last stage.

Chapter 4

Structural Verification

It was observed that most of the misclassifications by MLP were for numerals '3', '8'. It was also observed that mostly '3' and '8' were misclassified one another. Researchers have shown that postprocessing such as verification improves the overall performance of the recognition systems [1]. So, we decided to reduce the neural network misclassifications by doing structural verification at the last stage.

4.1 Algorithm

We used a very simple technique, if the result of MLP is '3' or '8', do structural verification. Our algorithm is presented in Table 4.1 which checks for crossing counts on each row.

-
- Set $\text{cross_count_a} = 0$, $\text{cross_count_b} = 0$
- For** $\text{row} = 1, \dots, \text{max_row}$
- Obtain crossing counts, i.e transitions from '0' to '1' or '1' to '0'
 - If crossing counts ≤ 2 , then $\text{cross_count_a} = \text{cross_count_a} + 1$
Else $\text{cross_count_b} = \text{cross_count_b} + 1$
- End**
- If $\text{cross_count_b} > \text{cross_count_a}$
Output result '8'
 - Else
Output result '3'
-

Table 4.1: Algorithm for structural verification of numerals '3' & '8'

4.2 Experiments and Results

Experiments were conducted on the test data set of 120 images of numerals using MLP and structural verifier for numerals '3' and '8'. The images were normalized to (30 x 25) for recognition purposes. It was observed that the misclassifications of '3' to '8' and '8' to '3' were improved thus increasing the overall success rate by 1.6%. Overall results obtained with 120 isolated images of numerals using the neural net (MLP) and structural verification are shown in Table 4.2

MLP with Structural Verification	96.6 %
----------------------------------	--------

Table 4.2: Results obtained with structural verification of numerals '3' & '8'

Numerals '3' & '8' correctly verified are shown in Table 4.3




Input Image	Written As	Classified As
	3	3
	3	3
	8	8

Table 4.3: Numerals '3' & '8' correctly verified

Wrong results after the verification stage are shown in Table 4.4




Input Image	Written As	Classified As
	3	8
	8	3
	7	1

Table 4.4: Wrong results after the verification stage

From Table 4.4 it can be observed that numeral '3' with extreme large tail at bottom is wrongly verified as '8' and numeral '8' with one side extremely small is wrongly verified as '3'. Numerals '1' and '7' were not subject to verification and hence resulted in wrong output.

Chapter 5

Conclusion

Recognition of handwritten English block letters and numerals was done and various experiments were performed. Following observations were made:

- *Information Content and Entropy* distance measurements does not prove to be of any use, as they depend on the probability of cell occupancy.
- Distance measurements with thinned images did not prove to be beneficial. The reason being that *thinning* is sensitive to noise.
- *In case of letters* “Nearest Neighbor-2” has excellent results and outperforms all other distance measurements. It can be said “Nearest neighbor-2” is the best choice amongst others.
- *In case of numerals* various distance measurements tend to complement each other. No single distance measurement is said to be sufficient. In order to obtain correct single recognition result, distance classifiers were combined using neural network of type MLP. Recognition by using MLP increased the success rate. It can be said “Combination has a vital role in increasing the recognition performance”.
- *Further improvements* to MLP results were obtained by adding a structural verifier for numerals ‘3’ and ‘8’. It can be said “Structural verifier plays an important role in increasing the recognition performance”.

References

1. C. Y. Suen, J. Kim, K. Kim, Q. Xu, and L. Lam, "Handwriting Recognition – The Last Frontiers," *15th International Conference on Pattern Recognition*, Vol. 4, pp. 1-10, Barcelona, September 2000.
2. R. Plamondon, N. Srihari, "On-Line and Off-Line Handwritten Recognition: A Comprehensive Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22, No.1, pp. 63-84, January 2000.
3. V. K. Govindanu, "Character Recognition – A Review," *Pattern Recognition*, Vol. 23, No. 7, pp. 671-683, 1990.
4. J. D. Tubbs, "A Note On Binary Template Matching," *Pattern Recognition*, Vol. 22, No. 4, pp. 359-365, 1989.
5. F. Y. Zhang and C. Y. Suen, "A Fast Parallel Algorithm for Thinning Digital Patterns," *Communications of ACM*, Vol. 27, pp. 236-239, March 1984.
6. C. Y. Suen and C. Shiau, "An Iterative Technique of Selecting An Optimal 5 x 7 Matrix Character Set for Display In Computer Output Systems," *Proceedings of The Society for Information Display*, Vol. 21, No. 1, pp. 9-15, 1980.
7. C. Y. Suen, C. Shiau and R. Shinghal, "Reliable Recognition of Handprint Data," *1976 Joint Workshop on Pattern Recognition and Artificial Intelligence*, pp. 98-102, Massachusetts, June 1976.
8. Mohamad H. Hassoun. The fundamentals of Artificial Neural Networks. The MIT Press, 1995.
9. Carl G. Looney. Pattern Recognition using Neural Networks. Oxford University Press, 1997.
10. R. Simon, M. Gouker, and B. Barnes, Win32 Programming API Bible, Waite Group Press, 1996.

Appendix A

Prototypes of English Block Letters

A	F	M	T
A	G	M	U
A	G	M	U
B	G	N	U
B	G	N	V
B	G	O	V
C	H	O	W
C	I	P	W
D	I	P	W
D	J	Q	W
D	J	Q	X
E	J	R	X
E	K	R	Y
E	K K	S	Y
F	L	S	Z
F	M	S	Z

Appendix B

Prototypes of Numerals

1	2	4	6	8	
1	2	4	7	8	
1	3	4	7	9	
2	3	5	7	9	
2	3	5	8	0	
2	3	6	8	0	0

Appendix C

Distance Measurements for Numerals

The table below shows the best matched measurement values between some inputted numeral styles and the prototypes shown in *Appendix B*.

Inputted Numeral	Nearest Neighbor-1	Nearest Neighbor-2	Linear Correlation	Cross Correlation	Similarity	Hamming
3	1.49	1.95	0.56	0.34	171	180
5	1.70	1.99	0.56	0.33	164	215
8	1.63	1.96	0.66	0.36	172	218
2	1.63	1.91	0.52	0.27	116	207

Notations

NN1	Nearest Neighbor-1
NN2	Nearest Neighbor-2
LC	Linear Correlation
CC	Cross Correlation
S	Similarity
H	Hamming

Appendix D

Distance Measurements for Letters

The table below shows the best matched measurement values between some inputted English block letter styles and the prototypes shown in *Appendix A*.

A	1.59	1.83	0.53	0.28	132	228
B	1.57	1.83	0.61	0.38	185	221
C	0.83	1.09	0.67	0.45	141	120
D	1.29	1.83	0.66	0.44	172	176
G	0.59	1.06	0.78	0.61	185	103
K	1.39	1.74	0.58	0.34	131	189
L	0.56	0.86	0.75	0.57	105	70
O	1.10	1.57	0.65	0.44	158	164
Q	0.69	1.03	0.73	0.53	167	122
Z	0.75	1.11	0.70	0.49	154	129

Notations

NN1	Nearest Neighbor-1
NN2	Nearest Neighbor-2
LC	Linear Correlation
CC	Cross Correlation
S	Similarity
H	Hamming

Appendix E

Feature Vectors Used in Training

Input Vector representation used in training the MLP

Feature Vector	Feature Description
x_1	Nearest Neighbor-1
x_2	Nearest Neighbor-2
x_3	Linear Correlation
x_4	Cross Correlation
x_5	Similarity Function
x_6	Hamming Distance

Target Vector representation used in training the MLP

	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9	d_{10}
0	1	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0
2	0	0	1	0	0	0	0	0	0	0
3	0	0	0	1	0	0	0	0	0	0
4	0	0	0	0	1	0	0	0	0	0
5	0	0	0	0	0	1	0	0	0	0
6	0	0	0	0	0	0	1	0	0	0
7	0	0	0	0	0	0	0	1	0	0
8	0	0	0	0	0	0	0	0	1	0
9	0	0	0	0	0	0	0	0	0	1

The table below shows the input vectors and the corresponding target vectors used in training the MLP.

x_1	x_2	x_3	x_4	x_5	x_6		x_1	x_2	x_3	x_4	x_5	x_6	
0.0	0.0	0.0	0.0	0.0	0.0	0	0.4	0.4	0.4	0.4	0.4	0.4	4
0.0	0.0	0.0	0.0	0.0	0.8	0	0.4	0.4	0.4	0.4	0.6	0.4	4
0.0	0.0	0.0	0.0	0.8	0.0	0	0.4	0.4	0.4	0.6	0.4	0.4	4
0.0	0.0	0.0	0.8	0.0	0.0	0	0.4	0.4	0.6	0.4	0.4	0.4	4
0.0	0.0	0.8	0.0	0.0	0.0	0	0.4	0.4	0.4	0.6	0.6	0.4	4
0.0	0.0	0.8	0.8	0.0	0.8	0	0.4	0.4	0.6	0.6	0.4	0.4	4
0.0	0.0	0.0	0.8	0.8	0.0	0	0.4	0.4	0.6	0.4	0.6	0.4	4
0.0	0.0	0.8	0.8	0.8	0.0	0	0.4	0.4	0.6	0.6	0.6	0.4	4
0.0	0.0	0.8	0.8	0.8	0.8	0	0.5	0.5	0.5	0.5	0.5	0.5	5
0.1	0.1	0.1	0.1	0.1	0.1	1	0.5	0.5	0.5	0.5	0.0	0.5	5
0.9	0.9	0.8	0.8	0.8	0.1	1	0.5	0.5	0.5	0.0	0.5	0.5	5
0.1	0.1	0.9	0.1	0.1	0.1	1	0.5	0.5	0.0	0.5	0.5	0.5	5
0.1	0.1	0.1	0.9	0.1	0.1	1	0.5	0.5	0.0	0.0	0.5	0.5	5
0.1	0.1	0.1	0.1	0.9	0.1	1	0.5	0.5	0.5	0.0	0.0	0.5	5
0.1	0.1	0.1	0.1	0.1	0.9	1	0.5	0.5	0.0	0.5	0.0	0.5	5
0.1	0.1	0.9	0.9	0.1	0.1	1	0.5	0.5	0.0	0.0	0.0	0.5	5
0.1	0.1	0.1	0.9	0.9	0.1	1	0.5	0.5	0.8	0.8	0.8	0.5	5
0.1	0.1	0.9	0.1	0.9	0.1	1	0.6	0.6	0.6	0.6	0.6	0.6	6
0.1	0.1	0.9	0.9	0.9	0.1	1	0.8	0.6	0.8	0.8	0.8	0.8	6
0.1	0.1	0.9	0.9	0.9	0.9	1	0.6	0.8	0.8	0.8	0.8	0.8	6
0.1	0.1	0.9	0.9	0.9	0.7	1	0.6	0.6	0.6	0.6	0.5	0.6	6
0.1	0.1	0.8	0.8	0.8	0.7	1	0.6	0.6	0.6	0.5	0.6	0.6	6
0.2	0.2	0.2	0.2	0.2	0.2	2	0.6	0.6	0.5	0.6	0.6	0.6	6
0.2	0.2	0.8	0.2	0.2	0.2	2	0.6	0.6	0.6	0.5	0.5	0.6	6
0.2	0.2	0.2	0.8	0.2	0.2	2	0.6	0.6	0.5	0.5	0.6	0.6	6
0.2	0.2	0.2	0.2	0.8	0.2	2	0.6	0.6	0.5	0.6	0.5	0.6	6
0.2	0.2	0.8	0.8	0.2	0.2	2	0.6	0.6	0.5	0.5	0.5	0.6	6
0.2	0.2	0.2	0.8	0.8	0.2	2	0.6	0.6	0.5	0.5	0.5	0.3	6
0.2	0.2	0.8	0.2	0.8	0.2	2	0.6	0.6	0.6	0.6	0.8	0.6	6
0.2	0.2	0.8	0.8	0.8	0.2	2	0.6	0.6	0.6	0.8	0.6	0.6	6
0.8	0.2	0.8	0.8	0.8	0.2	2	0.6	0.6	0.8	0.6	0.6	0.6	6
0.3	0.3	0.3	0.3	0.3	0.3	3	0.6	0.6	0.6	0.8	0.8	0.6	6
0.3	0.3	0.3	0.3	0.8	0.3	3	0.6	0.6	0.8	0.8	0.6	0.6	6
0.3	0.3	0.3	0.8	0.3	0.3	3	0.6	0.6	0.8	0.6	0.8	0.6	6
0.3	0.3	0.8	0.3	0.3	0.3	3	0.6	0.6	0.8	0.8	0.8	0.6	6
0.3	0.3	0.3	0.8	0.8	0.3	3	0.8	0.8	0.6	0.6	0.6	0.6	6
0.3	0.3	0.8	0.8	0.3	0.3	3							
0.3	0.3	0.8	0.8	0.8	0.3	3							

Table 1							Table 2							Table 3							Table 4						
x_1	x_2	x_3	x_4	x_5	x_6		x_1	x_2	x_3	x_4	x_5	x_6		x_1	x_2	x_3	x_4	x_5	x_6		x_1	x_2	x_3	x_4	x_5	x_6	
0.7	0.7	0.7	0.7	0.7	0.7	7	0.9	0.9	0.9	0.9	0.9	0.9	9	0.9	0.9	0.1	0.1	0.1	0.1	9	0.9	0.9	0.9	0.9	0.9	0.9	9
0.7	0.7	0.7	0.7	0.1	0.7	7	0.7	0.9	0.9	0.7	0.7	0.7	9	0.7	0.9	0.9	0.7	0.7	0.7	9	0.7	0.9	0.9	0.7	0.7	0.7	9
0.7	0.7	0.7	0.1	0.7	0.7	7	0.9	0.9	0.9	0.9	0.9	0.8	9	0.9	0.9	0.9	0.9	0.8	0.9	9	0.9	0.9	0.9	0.9	0.8	0.9	9
0.7	0.7	0.1	0.7	0.7	0.7	7	0.9	0.9	0.9	0.9	0.8	0.9	9	0.9	0.9	0.9	0.8	0.9	0.9	9	0.9	0.9	0.9	0.8	0.9	0.9	9
0.7	0.7	0.1	0.1	0.7	0.7	7	0.9	0.9	0.8	0.9	0.9	0.9	9	0.9	0.9	0.9	0.8	0.8	0.9	9	0.9	0.9	0.9	0.8	0.8	0.9	9
0.7	0.7	0.1	0.7	0.1	0.7	7	0.9	0.9	0.9	0.8	0.8	0.9	9	0.9	0.9	0.9	0.8	0.8	0.9	9	0.9	0.9	0.9	0.8	0.8	0.9	9
0.7	0.7	0.1	0.1	0.1	0.7	7	0.9	0.9	0.9	0.9	0.8	0.8	9	0.9	0.9	0.9	0.9	0.8	0.8	9	0.9	0.9	0.9	0.9	0.8	0.8	9
0.7	0.7	0.3	0.3	0.3	0.3	7	0.9	0.9	0.9	0.9	0.8	0.8	9	0.9	0.9	0.9	0.9	0.8	0.8	9	0.9	0.9	0.9	0.9	0.8	0.8	9
0.3	0.7	0.3	0.3	0.3	0.3	7	0.9	0.9	0.9	0.8	0.9	0.8	9	0.9	0.9	0.9	0.8	0.8	0.8	9	0.9	0.9	0.9	0.8	0.8	0.8	9
0.3	0.7	0.3	0.3	0.8	0.3	7	0.9	0.9	0.9	0.8	0.8	0.8	9	0.9	0.9	0.9	0.8	0.8	0.8	9	0.9	0.9	0.9	0.9	0.9	0.7	9
0.7	0.7	0.7	0.3	0.3	0.3	7	0.9	0.9	0.9	0.9	0.9	0.7	9	0.9	0.9	0.9	0.9	0.7	0.9	9	0.9	0.9	0.9	0.9	0.7	0.9	9
0.7	0.7	0.7	0.7	0.3	0.3	7	0.9	0.9	0.9	0.7	0.9	0.9	9	0.9	0.9	0.9	0.7	0.9	0.9	9	0.9	0.9	0.9	0.7	0.9	0.9	9
0.7	0.7	0.7	0.7	0.7	0.3	7	0.9	0.9	0.7	0.9	0.9	0.9	9	0.9	0.9	0.7	0.9	0.9	0.9	9	0.9	0.9	0.7	0.9	0.9	0.9	9
0.7	0.7	0.7	0.7	0.8	0.7	7	0.9	0.9	0.9	0.7	0.7	0.9	9	0.9	0.9	0.9	0.7	0.7	0.9	9	0.9	0.9	0.9	0.7	0.7	0.9	9
0.8	0.8	0.8	0.8	0.8	0.8	8	0.9	0.9	0.9	0.9	0.7	0.7	9	0.9	0.9	0.9	0.9	0.7	0.7	9	0.9	0.9	0.9	0.7	0.7	0.7	9
0.3	0.8	0.3	0.3	0.8	0.3	8	0.9	0.9	0.9	0.9	0.7	0.7	9	0.9	0.9	0.9	0.7	0.9	0.7	9	0.9	0.9	0.9	0.7	0.7	0.7	9
0.8	0.8	0.8	0.8	0.8	0.5	8	0.9	0.9	0.9	0.7	0.9	0.7	9	0.9	0.9	0.9	0.7	0.7	0.7	9	0.9	0.9	0.9	0.7	0.7	0.7	9
0.8	0.8	0.8	0.8	0.5	0.8	8	0.9	0.9	0.1	0.1	0.1	0.1	9	0.9	0.9	0.1	0.1	0.1	0.1	9	0.9	0.9	0.1	0.1	0.1	0.1	9
0.8	0.8	0.8	0.5	0.8	0.8	8																					
0.8	0.8	0.8	0.5	0.5	0.8	8																					
0.8	0.8	0.5	0.5	0.8	0.5	8																					
0.8	0.8	0.5	0.5	0.5	0.5	8																					
0.8	0.8	0.8	0.8	0.8	0.3	8																					
0.8	0.8	0.8	0.8	0.3	0.8	8																					
0.8	0.8	0.8	0.3	0.8	0.8	8																					
0.8	0.8	0.3	0.8	0.8	0.8	8																					
0.8	0.8	0.3	0.8	0.8	0.3	8																					
0.8	0.8	0.3	0.3	0.8	0.3	8																					
0.8	0.8	0.3	0.3	0.3	0.8	8																					
0.8	0.8	0.8	0.3	0.3	0.3	8																					
0.8	0.8	0.3	0.3	0.3	0.3	8																					