Portland State University

PDXScholar

Dissertations and Theses

Dissertations and Theses

1991

Neural network character recognition with a 2-D Fourier transform preprocessor

Daqiao Du

Portland State University

Follow this and additional works at: https://pdxscholar.library.pdx.edu/open_access_etds

Part of the [Electrical and Computer Engineering Commons](#)

Let us know how access to this document benefits you.

Recommended Citation

Du, Daqiao, "Neural network character recognition with a 2-D Fourier transform preprocessor" (1991). *Dissertations and Theses*. Paper 4200.

<https://doi.org/10.15760/etd.6084>

This Thesis is brought to you for free and open access. It has been accepted for inclusion in Dissertations and Theses by an authorized administrator of PDXScholar. Please contact us if we can make this document more accessible: pdxscholar@pdx.edu.

AN ABSTRACT OF THE THESIS OF Daqiao Du for the Master of Science in
Electrical and Computer Engineering presented August 7, 1991.

Title: Neural Network Character Recognition with a 2-D Fourier Transform
Preprocessor

APPROVED BY MEMBERS OF THE THESIS COMMITTEE

[Redacted Signature]

George G. Lendaris, Chairman

[Redacted Signature]

Fu Li

[Redacted Signature]

Hormoz Zarefar

In pattern recognition applications, it is usually important that the same identification be given for a pattern, independent of a variety of positions, rotations and /or distortions of the pattern within the recognition device's field of view. This research relates to development of a preprocessor for a neural network character recognition system, where the role of the preprocessor is to assist in minimizing the difficulties related to variations of position and rotations of a character within the field of view. The preprocessor explored here was suggested in 1970' (Lendaris & Stanly, 1970), and

is implemented here with more recent advances in neural network and discrete computation technologies.

The preprocessor consists of calculating the two-dimensional Fourier transform of the image (current hardware technology allows this to occur in less than 100 ms for a 256x256 pixels image , on a PC based machine with accelerator card), and then taking certain measurements on the transformed image. These measurements are given to the neural network, which processes the data to provide the character identification. Introduction of the preprocessor is shown to yield a great reduction in sensitivity to image translation and/or rotation.

NEURAL NETWORK CHARACTER RECOGNITION WITH
A 2-D FOURIER TRANSFORM PREPROCESSOR

by

DAQIAO DU

A thesis submitted in the partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE
in
ELECTRICAL AND COMPUTER ENGINEERING

Portland State University
1991

TO THE OFFICE OF GRADUATE STUDIES:

The members of the committee approve the thesis of Daqiao Du presented August 7, 1991.

George G. Lendaris, Chairman

Fu Li

Hormoz Zarefar

APPROVED:

Rolf Schaumann, Chairman, Department of Electrical Engineering

C. William Savery, Vice Provost for Graduate Studies and Research

ACKNOWLEDGEMENTS

I would like to thank Dr. George G. Lendaris, my thesis research advisor, for providing the underlying ideas and many detail instructions on this research project, for donating so much of his valuable time during this whole period I have worked on my thesis.

I would like to thank Shirley Clark, administrative assistant of the Department of Electrical Engineering, for warmly and continuously giving her help starting from when I just arrived at P.S.U. as a foreign student.

I would like to thank Zibin Tang and Ihab A. Harb, who were graduate students in the Department of Electrical Engineering, for sharing their knowledge.

I would like to thank my family, my mother and sisters, for fully caring for me to the best of their ability; and my father, who is no longer with us, for giving me spiritual support at all times.

TABLE OF CONTENTS

	PAGE
ACKNOWLEDGEMENTS.....	iii
LIST OF TABLES.....	vi
LIST OF FIGURES.....	vii
CHAPTER	
I INTRODUCTION.....	1
I.1 Representation and Recognition of Patterns.....	1
I.2 Objective.....	4
II FOURIER TRANSFORM AND FEATURE EXTRACTION.....	8
II.1 Classical Mathematical Definition of Fourier Transform...	8
II.2 One Dimensional Discrete Fourier Transform (DFT).....	10
II.3 Fast Fourier Transform (FFT).....	14
II.4 Two-Dimensional Discrete Fourier Transform.....	15
II.5 Application of Two-dimensional DFT Properties.....	18
II.6 Measurements of the Fourier Transform Plane.....	18
III NEURAL NETWORKS	19
III.1 History And Development of the Neural Network	19
III.2 Basic Idea and Theories.....	19
III.3 ADALINE Algorithm and Delta Rule.....	22

III.4 Back-propagation Algorithm.....	25
III.5 Improved Delta Rule.....	28
IV EARLY EXPERIMENTS	30
IV.1 Preparation for Experiments with Preprocessor	30
IV.2 Development of Data Sets.....	31
IV.3 Generalization Experiments without Preprocessor.....	32
IV.4 Generalization Experiments with Preprocessor.....	35
IV.5 Conclusions from Early Experiments.....	39
V FURTHER EXPERIMENTS AND FUTURE WORK.....	41
V.1 Extended Experiments.....	41
V.2 Data Preparation.....	41
V.3 Conclusion from Latest Experiments.....	45
SELECTED BIBLIOGRAPHY.....	46
APPENDICES	
A 16X16 GRID ENGLISH ALPHABET FONTS.....	47
B EXAMPLE OF ROTATED AND SHIFTED 16X16 GRID ENGLISH ALPHABET FONTS.....	55
C 256X256 GRID ENGLISH ALPHABET FONTS.....	62
D TRAINING ACCURACY PROFILE FOR DATA SETS I-VII	70

LIST OF TABLES

TABLE	PAGE
I Training and Testing Results for Neural Network	
(16x16 Input) without Preprocessor.....	34
II Training and Testing Results for Neural Network	
(32x32 Input) with Preprocessor.....	38
III Training and Testing Results for Neural Network	
(256x256 Input) with Preprocessor.....	44

LIST OF FIGURES

FIGURE	PAGE
1. Pattern Recognition Process.....	1
2. Three Dimensional Pattern Space	4
3. Time And Frequency Domain Signal.....	5
4. Biological Neural Model.....	7
5. Periodic and Nonperiodic Signals.....	9
6. Time and Frequency Domain Relationship.....	11
7. Model of Processing Element.....	21
8. ADALINE Neural Network.....	22
9. Weight Change in the Delta Rule.....	25
10. Neural Layers and Connections.....	27
11. Neural Network# I.....	33
12. Example of Pixel Expansion	35
13. Approximate Annual-ring and Pie-wedge Windows for the 32x32 Data Array.....	36
14. Neural Network# II.....	36
16 . Neural Network#III.....	37
17. 256x256 Grid Approximate Annual-ring and Pie-wedge Windows (Applied on FFT Spectrum of Letter "A" from Font TIMES)..	42
18. Neural Network# IV.....	42
19. Neural Network# V.....	44

CHAPTER I

INTRODUCTION

Pattern recognition methods are becoming more and more useful in science, industry and everyday applications. A variety of theories and techniques have been developed in this area over the years. This thesis explores the application of an approach to pattern recognition that was suggested in the 70's (Lendaris & Stanly, 1970), as augmented here with more recent advances in neural network and discrete computation technology. The specific task of the pattern recognition to be considered here is to recognize upper case letters of the English alphabet.

I.1. REPRESENTATION AND RECOGNITION OF PATTERN

Basically, the pattern recognition process comprises two stages: the first stage accepts the input pattern and extracts certain "features" from the pattern; the second stage accepts the features provided by the first stage, and classifies the pattern into desired categories. A general diagram given in Figure 1.

Figure 1. Pattern recognition process.

In typical applications, inputs are sets of measurements, taken from physical objects or images, and coded appropriately for processing.

Assume, for example, that a set of measurements $N_1, N_2, N_3, \dots, N_n$ represents a given pattern. Any change of the individual data values will correspond to a deformation of the original image or an entirely new image. Define an n -dimensional space P to be the input pattern space. A point in this space is represented by an n -dimensional vector, such as

$$p = \begin{vmatrix} p_1 \\ p_2 \\ p_3 \\ \vdots \\ p_n \end{vmatrix}$$

Conceptually speaking, in the case when there is a finite number of patterns, say m of them, the pattern space P can be described by $m \times n$ matrix as follows:

$$P = \begin{vmatrix} p_1^T \\ p_2^T \\ p_3^T \\ \vdots \\ p_m^T \end{vmatrix}$$

$$= \begin{vmatrix} p_{11} & p_{12} & \dots & p_{1n} \\ p_{21} & p_{22} & \dots & p_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{m1} & p_{m2} & \dots & p_{mn} \end{vmatrix}$$

where the superscript T after each pattern vector denotes the transpose.

The goal of a pattern recognition system is to reduce the dimensions of pattern vectors in the pattern space being researched, usually quite large; and then to sort the reduced-dimensional vectors, called feature vectors, into various clusters. A feature vector has fewer dimensions than the corresponding pattern vectors; but must be designed to obtain sufficient information to perform the classification task.

A feature vector is designated as

$$f_i^T = \begin{vmatrix} f_{i1} \\ f_{i2} \\ f_{i3} \\ . \\ . \\ . \\ f_{ir} \end{vmatrix} \quad (1.1.1)$$

where $i = 1, 2, \dots, m, r \leq n$ and the feature space $\mathbf{F}$ corresponding to the pattern space $\mathbf{P}$ is given by

$$\mathbf{F} = \begin{vmatrix} f_{11} & f_{12} & . & . & f_{1r} \\ f_{21} & f_{22} & . & . & f_{2r} \\ . & . & . & . & . \\ . & . & . & . & . \\ f_{m1} & f_{m2} & . & . & f_{mr} \end{vmatrix}$$

After receiving the feature vectors as inputs, the task of the classifier is to operate on them in a way that positions each vector as a point in a new classification space, where those points corresponding to similar input patterns cluster together. Each mass of points in the classification output space corresponds to a distinct class. A simple three-dimensional classification space with three classes is shown in Figure 2.

Figure 2. Three dimensional pattern space.

An output pattern is designated as

$$c = \begin{vmatrix} c_1 \\ c_2 \\ c_3 \\ \cdot \\ \cdot \\ c_j \end{vmatrix}$$

where j is the dimension of the classification space.

I.2. OBJECTIVE

The objective of this research is to demonstrate that a neural network can be trained to distinguish upper case letters of the English alphabet, independent of certain rotations, displacements and font characteristics, based on the output of a Fourier transform preprocessor. An initial experiment comprised 8 fonts from those available on the McIntosh computer, and encoded into 12X12 pixels, being justified to the upper left corner (by a graduate student, Kris Roginski, Oregon Graduate Institute). A

number of difficulties arise due to artifacts of the coarse sampling (12x12) of the letters. A subsequent experiment comprised 4 fonts taken from a Postscript printer. A 256x256 pixels encoding was used for subsequent experiments with better results.

I.2.1. Fourier Transform Preprocessor

The Fourier transform is a linear transformation which provides a useful mathematical relationship between a waveform traditionally in the time domain and the phase-amplitude of the waveform's content in the frequency domain. For example, the sinusoidal waveform in Figure 3 (a) becomes two impulses with their phase components as shown in Figure 3 (b).

Figure 3. Time and frequency domain signal. (a)sinusoidal waveform; (b)frequency waveform.

The frequency spectrum, as shown in Figure 3 (b), is symmetrical about the origin. This is one of the properties of the Fourier transform. A consequence of this is that only half of the transformed plane can represent the whole original sinusoidal waveform. In fact, a small "window" that just looks at the small area containing one of the impulses acquires sufficient information to reconstruct the sinusoidal waveform.

These properties provide a basis for a more efficient representation of the original waveform.

An image on a plane is two dimensional. The term "two dimensional" means here that the dependent variable is a function of two coordinates, such as t_1 and t_2 , instead of a single coordinate t , as in the one dimensional situation. The Fourier transformation can be applied to the two dimensional case, with prospects for more efficient representation as suggested above for the one-dimensional case.

I.2.2. Neural Network

A neural network is a computational mechanism whose design is motivated by biological brain. This area has been researched for decades, and more recently, has been developed into real application status. In biological brain, a neuron is the fundamental cellular unit. It consists of a nucleus, which receives synaptic signals from other neurons, dendrites as input paths to the nucleus and an axon that carries the output signal. If the input signals are strong enough, an outgoing signal will be produced and sent through the output path. On each input path, there is a junction called synapse, which determines the strength of the input signal. It is estimated that the human brain has 10^{10} neurons with an average of 10^3 connections to each other.

An artificial neural network is created to have similar components and types of connections. The processing elements are approximations of a neuron. There are other functional parts that correspond to properties of a brain, such as activation function, input path, and weights. Following some selected algorithm, these processing elements convey the treated information to each other. When a coded pattern is presented at the input side of a neural network, an output is generated based on the interconnections and the current weight values of each processing element. If the output is not as desired,

the neural network will adjust its weights in an attempt to have its output approach the desired value. This process is called "learning". In real applications, the "learning" will consist of a huge number of iterations until a correct response appears. After a neural network has been trained for some inputs, it can usually provide useful outputs even for distorted or noisy inputs. This quasi real-time, disturbance-resistant approach has clear advantages compared to other software algorithms or hardware devices.

Figure 4. Biological neural model.

The remainder of thesis elaborates on some relevant theories and methods for feature extraction and classification, describes the experiments performed, gives some conclusions and topics for future research.

CHAPTER II

FOURIER TRANSFORM AND FEATURE EXTRACTION

II.1. CLASSICAL MATHEMATICAL DEFINITION OF FOURIER TRANSFORM

A periodic signal

$$g_T(t) = g(t + m T_0) \quad T_0 \neq 0$$

where $m = 1, 2, 3, \dots$ T_0 is the period of $g_T(t)$

can be representation by a Fourier series:

$$g_T(t) = a_0 + \sum_{n=1}^{\infty} (a_n \cos 2\pi n f_0 t + b_n \sin 2\pi n f_0 t) \quad (2.1.1)$$

$$a_0 = \frac{1}{T_0} \int_{t_0}^{t_0 + T_0} g_T(t) dt$$

$$a_m = \frac{1}{T_0} \int_{t_0}^{t_0 + T_0} g_T(t) \cos m \omega_0 t dt$$

$$b_m = \frac{1}{T_0} \int_{t_0}^{t_0 + T_0} g_T(t) \sin m \omega_0 t dt$$

where t_0 is arbitrary time, $T_0 = \frac{1}{f_0} = \frac{1}{2\pi\omega_0}$

an alternative expression is

$$g_T(t) = c_0 + \sum_{m=1}^{\infty} c_m \cos(m \omega_0 t + \Theta_m) \quad (2.1.2)$$

$$c_0 = a_0$$

$$c_m = \sqrt{a_m^2 + b_m^2}$$

$$\Theta_m = -\tan^{-1} \frac{b_m}{a_m}$$

a second alternative is the exponential form:

$$g_T(t) = \sum_{m=-\infty}^{\infty} G_m e^{jm\omega_0 t} \quad (2.1.3)$$

where

$$G_m = \frac{1}{T_0} \int_{t_0}^{t_0+T_0} g_T(t) e^{-jm\omega_0 t} dt \quad (2.1.4)$$

Figure 5. Periodic and nonperiodic signals.

A non-periodic signal $g_{mp}(t)$ that has the same components as shown in figure 5. can be represented by a Fourier transform, by assuming T_0 to be infinity:

$$\begin{aligned} g_T(t) &= \lim_{T_0 \rightarrow \infty} g_{mp}(t) = \lim_{\Delta\omega \rightarrow 0} \frac{1}{2\pi} \sum_{m=-\infty}^{\infty} G(m\Delta\omega) e^{jm\Delta\omega t} \\ &= \frac{1}{2\pi} \int_{-\infty}^{\infty} G(\omega) e^{j\omega t} d\omega \\ G(\omega) &= \int_{-\infty}^{\infty} e^{-j\omega t} dt \end{aligned} \quad (2.1.5)$$

$g(t)$ and $G(w)$ respectively reflect the characteristic of the signal in continuous time domain and continuous transformed domain, usually called frequency domain. Such a transform pair may be designated by the notation:

$$g(t) \longleftrightarrow G(w)$$

There are several important properties of Fourier transform pairs. Three of them are mentioned here for future application in the thesis as follows:

1) Symmetry:

$$\begin{aligned} \text{if } g(t) &\longleftrightarrow G(w) \\ \text{then } |G(w)| &= |G(-w)| \end{aligned}$$

2) Time shifting:

$$\begin{aligned} \text{if } g(t) &\longleftrightarrow G(w) \\ \text{then } g(t-t_0) &\longleftrightarrow G(w)e^{-j\omega t_0} \end{aligned}$$

3) Scaling:

$$\begin{aligned} \text{if } g(t) &\longleftrightarrow G(w) \\ \text{for a real constant } a, \\ \text{then } g(at) &\longleftrightarrow \frac{1}{|a|} G\left(\frac{w}{a}\right) \end{aligned}$$

II.2. ONE DIMENSIONAL DISCRETE FOURIER TRANSFORM(DFT)

It is inconvenient to realize the computation for equation (2.1.4) on most available digital computers or other popular digital hardware because of the integration calculation. The discrete fourier transform technique offers easy access to solve such

problems by means of using summation and limited terms instead of the integration and infinite terms.

According to sampling theory, a continuous signal g_T that has limited bandwidth B_t and period T can be sampled into a discrete sequence $g_T^*(t)$ without losing any information if the sampling rate $f_p \geq 2 B_t$. In other words, as long as the sampling rate f_p is twice as large as the highest frequency contained in the original signal, $g_T(t)$ can be expressed by $g(n\Delta t)$, or for short, $g_T(n)$.

Here, $n = 0, 1, 2, \dots, N-1$ denotes the position of each sampling point; since the sampling points within one period T are $N = \frac{T}{\Delta t}$, $\Delta t = \frac{1}{f_p}$.

Figure 6. Time and frequency domain relationship.

To explain this sampling procedure, a $\delta(t)$ function is introduced to represent the sampled sequence:

$$\delta(t) = 0 \quad t \neq 0$$

$$\delta(t) = \infty \quad t = 0$$

$$\int_{-\infty}^{\infty} \delta(t) dt = 1$$

and it is easy to prove the $\delta(t)$ function has following properties

$$\int_{-\infty}^{\infty} g(t) \delta(t) dt = g(0) \quad (2.2.1)$$

$$\int_{-\infty}^{\infty} g(t) \delta(t - t_0) dt = g(t_0) \quad (2.2.2)$$

Therefore the sampling sequence of $g_T^*(t)$ can be written as

$$\begin{aligned} g_T^*(t) &= \sum_{-\infty}^{\infty} g_T(t) \delta(t - n\Delta t) \\ &= \sum_{-\infty}^{\infty} g_N^*(n) \delta(t - n\Delta t) \end{aligned}$$

where $g_N(n)$ is the sampling value of $g_T(t)$.

Using $g_T^*(t)$ replace the $g_T(t)$ in equation (2.1.3)

$$\begin{aligned} G_m &= \frac{1}{T} \int_T g_{T(t)} e^{-jm\omega t} dt \\ &= \frac{1}{T} \int_{T-\infty}^{\infty} \sum g_N \delta(t - n\Delta t) e^{-jm\omega t} dt \\ &= \frac{1}{T} \int_T \sum_{0}^{N-1} g_N \delta(t - n\Delta t) e^{-jm\omega t} dt \\ &= \frac{1}{T} \sum_{0}^{N-1} \Delta t g_N \int_T \delta(t - n\Delta t) e^{-jm\omega t} dt \end{aligned}$$

since $\omega = \frac{2\pi}{T}$ and $\Delta t = \frac{T}{N}$

$$\begin{aligned} G_m &= \frac{1}{N} \sum_{n=0}^{N-1} g_N(n) \delta(t - n \Delta t) e^{-j\omega t} \\ &= \frac{1}{N} \sum_{n=0}^{N-1} g_N(n) e^{-jmn\frac{2\pi}{N}} \end{aligned} \quad (2.2.4)$$

For the sake of simplicity in the notation, it is convenient to let

$$W_N = e^{-j\frac{2\pi}{N}} = \cos\frac{2\pi}{N} - j \sin\frac{2\pi}{N}$$

And

$$G_N(m) = \frac{1}{N} \sum_{n=0}^{N-1} g_N(n) W_N^{mn} \quad (2.2.5)$$

Then (2-2-4) becomes

$$G_m = \frac{1}{N} G_N(m) \quad m = \pm 1, \pm 2, \pm 3, \dots, \pm \infty \quad (2.2.6)$$

As observed from equation (2.2.5), W_N is periodic with period N , whereas $G_N(m)$ also is periodic with period N . So to obtain the G_m for $g_T(t)$, N points need to be computed. And a new variable $k = 0, 1, 2, 3, \dots, N-1$ is introduced for convince

$$G_N(k) = \sum_{n=0}^{N-1} g_N(n) W_N^{nk} \quad k, n = 0, 1, 2, \dots, N-1 \quad (2.2.7)$$

$$G_k = \frac{1}{N} \sum_{s=-\infty}^{\infty} G(k + sN) \quad k = 0, 1, 2, \dots, N-1 \quad (2.2.8)$$

equations (2-2-7) and (2-2-8) clearly show that only N points are needed to compute for the $G_n(n)$ and G_k could be extended to be periodic from the result of N point calculation.

II.3. FAST FOURIER TRANSFORM (FFT)

Consider the DFT (2.2.7)

$$G(k) = \sum_{n=0}^{N-1} g(n) e^{-j2\pi \frac{nk}{N}} \quad k, n = 0, 1, 2, \dots, N-1$$

This describes the computation of N equations. For example

if $N=4$, $W = e^{-j2\pi \frac{1}{N}}$

then (2-2-7) can be written as

$$\begin{bmatrix} G(0) \\ G(1) \\ G(2) \\ G(3) \end{bmatrix} = \begin{bmatrix} W^0 & W^0 & W^0 & W^0 \\ W^0 & W^1 & W^2 & W^3 \\ W^0 & W^2 & W^4 & W^6 \\ W^0 & W^3 & W^6 & W^9 \end{bmatrix} \begin{bmatrix} g(0) \\ g(1) \\ g(2) \\ g(3) \end{bmatrix} \quad (2.3.1)$$

or more compactly as

$$G(k) = W g(n) \quad (2.3.2)$$

Since W and possibly $g(n)$ are complex, the N complex multiplications and $N(N-1)$ complex additions are necessary to perform the required matrix computation. The fast Fourier transform algorithms may efficiently reduce the number of multiplications and additions required in computation of (2.3.1). When N is not a prime number, specifically $N=2^q$, $q = 1, 2, 3, \dots$, the optimized FFT algorithms could reduce the number of the complex multiplications and additions to $N(\log_2 N)$.

The various algorithms for the FFT have been discussed in the literatures such as "Numerical Recipes" (William H. Press ...et al, 1988) and "Digital Signal Processing" (Richard A. Robert and Clifford T. Mullis, 1987). No further details will given here.

II.4. TWO-DIMENSIONAL DISCRETE FOURIER TRANSFORM

In real world applications, the discrete Fourier transform representation of two-dimensional sequences is of considerable computational importance in the digital processing of two-dimensional signals such as photography, and image array data including alphabet fonts for pattern recognition. Many of the transform properties for one-dimensional signals are extendable to two-dimensional signals. A similar generalization from discrete Fourier series applies to the two-dimensional discrete Fourier series as

$$g(m, n) = g(m + qM, n + rN)$$

where q and r are arbitrary positive or negative integer.

Such sequences have a Fourier series representation as a finite summation of complex exponentials in the form

$$g(n, m) = \frac{1}{MN} \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} G(k, l) W_M^{-km} W_N^{-ln} \quad (2.4.1)$$

where $G(k, l)$ can be shown to be

$$G(k, l) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} g(m, n) W_M^{km} W_N^{ln} \quad (2.4.2)$$

$$\text{and } W_M = e^{-j2\frac{\pi}{M}}$$

$$W_N = e^{-j2\frac{\pi}{N}}$$

A finite-duration sequence is handled in a manner similar to the one-dimensional situation described before, such sequence can be as a single period of a periodic sequence to applied the Fourier series theorem.

To develop a two-dimensional DFT, a finite area sequence $g(m, n)$ is defined as zero except for the interval $0 \leq m \leq M-1, 0 \leq n \leq N-1$. It is of area (M, N) , and construct the periodic sequence

$$g(m, n) = g(m, n)R(m, n)$$

$$\text{where } R(m, n) = 1, \quad 0 \leq m \leq M-1, 0 \leq n \leq N-1$$

$$R(m, n) = 0 \quad \text{otherwise}$$

The discrete Fourier transform of $g(m, n)$ correspond to the Fourier series of $g(m, n)$.

$$g(m, n) = \frac{1}{MN} \left[\sum_{k=0}^{M-1} \sum_{l=0}^{N-1} G(k, l) W_M^{-km} W_N^{-ln} \right] R_{MN}(m, n) \quad (2.4.3)$$

$$G(k, l) = \left[\sum_{m=0}^{M-1} \sum_{n=0}^{N-1} g(m, n) W_M^{-km} W_N^{-ln} \right] R_{MN}(m, n) \quad (2.4.4)$$

Since k and l are independent each other, the two-dimensional DFT is used in terms of one-dimensional DFT. That is because of the independence of k and l . we can write $R_{MN}(k, l) = R_M(k) \cdot R_N(l)$, then equation (2.4.4) becomes

$$G(k, l) = \left[\sum_{n=0}^{N-1} H(k, n) W_N^{-ln} \right] R_N(n) \quad (2.4.5)$$

where

$$H(k, l) = \left[\sum_{m=0}^{M-1} g(m, n) W_M^{-km} \right] R_M(k)$$

The function $H(k, l)$ corresponds to an M point one-dimensional DFT for each value of n . In other words, a M by N two-dimensional sequence can be computed by doing an M point one dimensional DFT N times. Therefore the calculation of two-dimensional DFT is simplified to the solution of the one-dimensional problem.

Also the properties of a two-dimensional DFT are similar to those of a one dimensional DFT, even one-dimensional continuous Fourier transform. Some of those properties can be proved easily by using the two-dimensional DFT transform pair (2.4.1) and (2.4.2). Three of them are listed as

1) Conjugate Symmetry:

$$\text{if } g(m, n) \leftrightarrow G(k, l)$$

$$\text{then } G(n, m) = G^*(-m, -n)$$

$$\text{also } |G(m, n)| = |G(-m, -n)|$$

2) Spatial shifting:

$$\text{if } g(m, n) \leftrightarrow G(k, l)$$

$$\text{then } \leftrightarrow G(k, l) e^{-j2\pi(kn_0 + lm_0)/N}$$

$$\text{or } |g(n-n_0, m-m_0)| \leftrightarrow |G(k, l)|$$

3) Rotation Invariance:

when polar coordinates (ρ, θ) and (γ, ϕ) are substituted for the rectangular coordinates then $g(m, n)$ and $G(k, l)$ becomes $g(\rho, \theta)$ and $G(\gamma, \phi)$ respectively. If using $m = \rho$, $n = \theta$, $l = \phi$, $m_0 = 0$, $n_0 = \Delta\theta$

$$\text{then } g(\rho, \theta + \Delta\theta) \leftrightarrow G(k, l) e^{j2\pi l \Delta\theta} = G(k, l + \Delta\theta)$$

II.5. APPLICATION OF TWO-DIMENSIONAL DFT PROPERTIES

The properties mentioned above imply some very important relationships between the spatial and transform domain:

1) Conjugate Symmetry implies that only half of the transform domain is enough to represent the whole spatial domain.

2) A shifting of $g(m,n)$ does not affect the magnitude of its transformed domain at all.

3) A rotation of $g(m, n)$ causes a same angle rotation of $G(k, l)$.

II.6. MEASUREMENTS OF THE FOURIER TRANSFORM PLANE

A method was developed by Lendaris(Lendaris & Stanly, 1970) in the 1970's for classifying images based on measurements taken from the two-dimensional Fourier transform on those images. Such a measurement consists of adding up the amplitude squared of all the points in a specified region of the Fourier Transform plane. Each area is called a sampling window. Several sampling window geometries were suggested to measure the amount of light energy of a continuous Fourier Transform optically generated. A set of annular-ring sampling windows can be used for detecting the regularity of a spatial image by adding up the values of every pixel within each ring. Another set of wedge shaped sampling widows gives a direction profile of the content.

In the special case of digitized fonts of English alphabet represented by 32x32 pixels, the 15 annular-ring summations plus 16 summations for the pie-wedge windows comprises the output of the preprocessor for the basic experiments.

CHAPTER III

NEURAL NETWORKS

III.1. HISTORY AND DEVELOPMENT OF THE NEURAL NETWORK

The earliest prototype neural network was created by McCulloch and Pitts (1943). It simulates a brain cell. And in such simulating process, the output depends on the input signal and the action threshold at input side.

In 1950's, Frank Rosenblatt(1957) developed the McCulloch-Pitts model and built a "trainable" network that can "learn" some desired task. Such network was named "Perceptron" because of its eye retina-like structure. Later, Rosenblatt proved theoretically that his Perceptron algorithm can surely converge to a final solution if the solution exists.

Rosenblatt's convergence theory apply only to single-layer, feed-forward flow networks and these turn out not being powerful enough for many of real world problems. During this past decade, significant improvement have been made. Especially John Hopfield(1984) and David Rummelhart(1984), respectively, developed algorithms and proved in the convergence for a fully connection and multiple-layer networks. These main stream and many parallel researches have laid a foundation for today's upsurge in this area.

III.2. THE BASIC IDEA AND THEORIES

An elemental cell of neural network, called processing element, for short, element, has a similar function with a biological neuron as described in chapter I. This

kind of element has a group of input paths and an output path except for those located in the open terminal end of Neural Network's incoming and outgoing signals. A signal coming in through the input path is multiplied by a weight variable; therefore, it can be said that the weighted signal finally reaches the nucleus of the element. Being stimulated by its inputs, an element reacts first sums the weighted input signals; next, it puts the summation through a transfer function to yield an output signal. Both the summation and output signal could be expressed as following:

$$net_{pj} = \sum_i w_{ji} o_{pi} \quad (3.2.1)$$

$$o_{pj} = f_j (net_{pj}) \quad (3.2.2)$$

where o_{pj} is the output state of the j^{th} element for an input patten p, w_{ij} is the weight of the connection joining the i^{th} element to j^{th} element, net_{ij} , is the summation of input to the j^{th} element, $f()$ is the transfer (activation) function.

Usually, the activation function for a given element is fixed at the time when the neural network is established; and so the only changeable factor that affects an output signal is the weights belonging to the elements when a particular pattern represents at the input path. After defining the basic element of neural network, how to structure such a trainable and efficient network becomes an important topic. The choice of neural algorithm mainly depends on these three conditions:

- 1) the way to build a connection among the processing elements.
- 2) the way to modify the weights of the processing elements.
- 3) the choosing of activation function.

Figure 7. Model of processing element.

A researcher may combine different choices of the three conditions in an attempt to meet the requirements of the environment and input patterns,. Generally, neural networks can be divided into two categories: supervised network and unsupervised network.

In the supervised situation, a training input pattern is presented at the neural network's input terminal, and the correct output for that input pattern is used by the teacher to direct the neural network to achieve that correct answer eventually. In each training iteration, if an input pattern has a response along the right direction, it means that the neural network's output gets closer to the desired (target) output . Then the neural network does nothing on the current processing input. Otherwise, the neural network will take an appropriate action by applying selected learning algorithm to adjust appropriate weights according to the needs. In brief, a supervised neural network knows whether its learned representation are right or not and how the representation can be made right.

Figure 8. ADALINE neural network.

In an unsupervised case, there is no desired output available for a "teacher" to use. This method is not used for the present research, and therefore will not be discussed. Both networks mentioned above are important for different purposes of applications. This thesis will discuss the supervised case only, which is feasible for the research subject of character recognition.

III.3. ADALINE ALGORITHM AND DELTA RULE

At the early stages of developing "learning machines", an "Adapted Linear Element" (ADALINE) was developed by Bernard Widrow (1962). The corresponding learning rule is called Delta rule. The ADALINE neural network has a single layer structure show as following:

In ADALINE network, each processing element has same structure as figure 3-3-1 in addition to a feedback loop and a desired output. To make a ADALINE network element start its learning procedure, a learning rule that regulates the method of weight change depends on whether a group of specially coded input patterns has been trained correctly or not. For pattern p , the weight change at each training iteration is given by:

$$W_{p_{new}} = W_{p_{old}} + D(t_{pj}, o_{pi}, o_{pj}) \quad (3.3.1)$$

or:

$$W_{p_{new}} - W_{p_{old}} = D(t_{pj}, o_{pi}, o_{pj}) \quad (3.3.2)$$

or:

$$\Delta W_{ij} = D(t_{pj}, o_{pi}, o_{pj}) \quad (3.3.3)$$

where $D()$ is an increment or decrement from the old weight $W_{p_{old}}$ to new weight $W_{p_{new}}$. It is a function of variables t_{pi} , o_{pi} , and o_{pj} .

In more detail, Hebb(1949) crafted the function $D()$ as:

$$D(t_{pj}, o_{pi}, o_{pj}) = \eta(t_{pi} - o_{pj}) = \eta \delta_{pj} o_{pi} \quad (3.3.4)$$

where η is a learning index that will be discuss later

Bringing the equation (3.3.3) to (3.3.4):

$$\Delta W_{ij} = \eta \delta_{pj} o_{pi} \quad (3.3.5)$$

This is the original Delta rule. It was mathematically proved to be a steepest descent procedure by Widrow later (1962). The Delta rule was verified to yield a statistical solution for minimizing the square error over the training samples. The actual error function should be:

$$E_p = \frac{1}{2} \sum_i (t_{pi} - o_{pi})^2 \quad (3.3.6)$$

To obtain the relationship between E_p and ΔW_{ji} in the error space, a gradient descent method is used to find the changed direction and amplitude for the weight vector around an error surface (those direction and amplitude lead weight vector gradually to reach the final idea point on the error surface). To show this mathematically, we take partial derivative on E_p :

$$\begin{aligned}\frac{\partial E_p}{\partial o_{pj}} &= -(t_{pj} - o_{pj}) \\ &= \delta_{pj}\end{aligned}\tag{3.3.7}$$

$$\text{since } o_{pj} = \sum W_{ji} o_{pi} \tag{3.3.8}$$

$$\frac{\partial o_{pj}}{\partial W_{ji}} = o_{pi} \tag{3.3.9}$$

$$\text{so: } \frac{\partial E_p}{\partial W_{ji}} = \frac{\partial E_p}{\partial o_{pj}} \frac{\partial o_{pj}}{\partial W_{ji}} = \delta_{pj} o_{pi} \tag{3.3.10}$$

A simple instance explains the equation graphically based on two input paths and weight components existing for a processing element. Therefore, there is a three-dimensional error surface as shown as Figure 9..

The bottom of the bowl-shape surface is the point where the weight vectors have perfect direction and amplitude since the error term in the equation(3.3.10) is minimum (zero if the bowl touches the x, y plane) where the weight vector no longer needs to change. What the Delta rule has to do is simply to move the weight vector from its current position on the bowl-shape surface to a new position where is closer to the bottom point(minimum error point) by falling down the negative gradient of the bowl. That is why the Delta rule is also called gradient-descent learning rule. An ADALINE neural network's weights initially starts at an arbitrary point around the bowl-shape surface. Step by step, the weight vector approaches the minimum error point where the change of the weight vector terminates. And the whole training procedure is accomplished.

Figure 9. Weight change in the delta rule.

Two key points are to pointed out regarding the Delta rule:

First, during a training procedure, as described before in equation(3.3.1) and (3.3.4), the learning index η determines the amount of change for each weight at each step. A larger delta not only cause faster movement to minimum error point but also has more risk to cause oscillation around the point. In general, for a neural network that just starts from an initial status, a larger delta should be chosen; and then it should gradually be reduced along with the accumulation of training time. At the final training status, the learning index has its minimum value which makes the change ("Delta") of the weight a small amount. Therefore, both problems on the training speed and possible oscillation could be taken care of.

III.4 BACK-PROPAGATION ALGORITHM

An ADALINE that has a simple single layer possesses the ability to "learn". However, under the principle of ADALINE, its processing elements are separately trained to respond exactly what their own particular inputs are. It means that certain input pattern can be exclusively lead to certain straight output. In actuality, a learning machine should have the ability to synthesize and analyze obscured or overlapped input

pattern which has never been presented and categorized, that is, a creative thinking ability other than mechanical comparing.

In order to learn the similarity of those patterns, a more complete neural network is needed. Minsky and Papert (1969) have provided a detailed analysis of the condition for such networks in which at least three layers, one input layer, at least one middle layer and an output layer are required. The input information comes to the elements in the input layer and is recoded into an internal representation, then is evenly distributed to each of the middle-layer elements. The outputs of the neural network are generated by the internal representation rather than by the original input pattern.

A Back-propagation algorithm is a typical application of the multiple-layer network. The input layer (first layer) simply works as a buffer or distributor to receive input signals and assign them to the elements of the middle layer. Each output path on each input-layer element connects only to the input paths of the next layer. There is no connection among elements within a layer itself. This structure of connection continues to as many layers as needed in the neural network except for the last layer (output layer) in which each output path contacts directly with the outside world.

In each "training" iteration (one input pattern presented once), two flows, a forward activation flow and a backward error flow appear in succession. The former presents an input pattern to the input layer of the network and propagates the input through all layers to obtain a general network output, as well as to set all summed input o_{pi} and o_{pj} output for all processing elements in the network. From the equation (3.1.2), the output vector in output layer is:

$$o_{pj} = f_j (net_{pj}) \quad (3.4.1)$$

Figure 10. Neural layers and connections.

Starting from the output layer, the backward error flow effects the network by computing local error and adjusting weights for all elements in this layer according to "generalized" Delta rule equation (3.3.5):

$$\Delta W_{ji} = \eta \delta_{pj} o_{pj}$$

and equation

$$\delta_{pj} = (t_{pj} - o_{pj}) f'_j(\text{net}_{pj}) \quad (3.4.2)$$

where $f'(\text{net}_{pj})$ is the derivative of activation function in the element.

For each previous layers(backward flow begins at the output layer) such as hidden layer II in figure 10., the desired perfect output is unknown. The local errors on the element in this layer are based on their portion of contribute to the next layer. The amount is computed as:

$$\delta_{pj} = f'_j(\text{net}_{pj}) \sum_k \delta_{pk} W_{kj} \quad (3.4.3)$$

In analogy to this, the back error flow passes through all layers except for the input layer and updates corresponding to the weights in those layers. When an input

pattern appears again and again, both flows recycle until the network possesses the optimum value of weight vector for the input pattern.

Now here is a literal summation of the operation of a neural network during training: A neural network receives an input pattern and generates an activation flow from the input layer through the output layer. After comparing desired output and the actual output at the output layer, the network, beginning at the output layer, adjusts the weights layer by layer; propagates error of each layer back to the previous layer and computes the amount of the change for each weight. The weight change happens consequently in reversed direction. Each pair of forward and backward flows form a complete iteration of training.

III.5. IMPROVED DELTA RULE

As explained before, the Delta rule plays a very important role in a training process of a neural network. The learning index parameter, η , is available for adjustment by the researcher. Its value has to be set properly; otherwise, the network may either be trained quite slowly(η too small) or be oscillately around the minimum point (η too large). Furthermore, the Delta rule applies the gradient-descent principle to look for a minimum point, but does not guarantee whether the point is local or global minimum.

An additional term, called Momentum, is added to assist in solving this problem. The Improved Delta rule is more useful in general neural research and applications

$$\Delta W_{ij}(n) = \eta \delta_{pj} o_{pj} + \alpha \Delta W_{ji}(n-1) \quad (3.5.1)$$

where α is a value-changeable index, usually range (0.0 , 1.0) . $\Delta W_{ji}(n-1)$ is the amount of weight change for the training before the current one.

The extra term functions just as a sled on a hill covered with snow to overcome local minimum and make the network approach a global minimum more surely.

CHAPTER IV

EARLY EXPERIMENTS

IV.1. PREPARATION FOR EXPERIMENTS WITH PREPROCESSOR

These experiments were designed to fit the limitations of the maximum number of elements allowed in the current neural network simulation software(NeuralWorks Rev.2.00 NEURALWARE, INC.) and limitations of the available equipment. For these basic experiments, each original letter from the 12x12 upper-left corner justified fonts, as mentioned before, was centered and expanded to 16x16 and 32x32 pixel formats.

In a previous chapter, it was noted that a feed-forward, back-propagation of error algorithm, known as back-propagation, was selected for this supervised learning case. The input to such a neural network is the feature vector, and in the present case, the "features" are the pixels themselves. The output side has 26 wires, which respectively represent 26 letters of the English alphabet. Each output wire has a value from 0.00 to 1.00. The higher value an output wire has, the more affirmative its "vote" for the corresponding letter is. A threshold value T , set between 0.00 and 1.00, is used to make the "Yes" or "No" judgement on every wire for its corresponding letter.

The training algorithm, number of layers, and the number of elements in each layer are selected for a particular experiment. For each element, the indices α and β (as described earlier) are set. In addition, the simulator has an index G , which adjusts the slope of the sigmoid transfer function. A bigger G value means a steeper slope of

the sigmoid transfer function, and a steeper slope forces the value on the output wires to be clearly close to 0.00 or close to 1.00. But too big a G value will cause a neural network to misrecongize certain input fonts:

- 1) None of the wire values exceeds the threshold, so no letter is affirmed.
- 2) More than one wire value exceeds the threshold, so more than one output is affirmed into the same input letter.

To avoid the above situations and to get exactly one wire value to exceed the threshold, some pre-training test neural networks were created for determining a network structure and the values of G, α and β .

IV.2. DEVELOPMENT OF DATA SETS

The fonts named TIMES, PALATINO, HELVETICA, GENEVA, AVANT GARDE, CARTOON, MONACO and ASCHAM were used and arranged to suit the study purposes of this thesis, as follows:

- 1) Input data set I for generalization on rotations:

The following formula was used to rotate each letter font array mathematically

$$\begin{aligned} y_{new} &= x_{old}\cos\theta + y_{old}\sin\theta \\ x_{new} &= -x_{old}\sin\theta + y_{old}\cos\theta \end{aligned} \quad (4.2.1)$$

where x_{old} and y_{old} are the position in the original font array, the x_{new} and y_{new} are the position in the rotated font array, the θ is the rotated angle.

Each letter of the eight original fonts were individually rotated counter-clockwise to 0° , $+22.5^\circ$, $+45^\circ$, $+67.5^\circ$, $+90^\circ$ and used as input training set for the neural network. The network was tested in the following rotated versions of the same fonts:

rotated counter-clockwise $+11.50^\circ$, $+33.75^\circ$, $+56.25^\circ$, $+78.75^\circ$, rotated clockwise -22.5° , -45° , -67.5° , -90° and -11.25° , -33.75° , -56.25° , -78.75° .

2)Input data set II for generalization on shifts:

Four of the original fonts were used as training fonts; The same fonts were shifted by (right 3 pixels, up 3 pixels) and (left 3 pixels, down 3 pixels) for testing the neural network's capability for generalizing on shifting.

3)Generalization on fonts:

a)Input data set III for training and testing:

Four of the original fonts were used as training fonts: four of the remaining fonts were used as test fonts.

b)Input data set IV for training and testing:

Four of the fonts with all rotations were used as training fonts: four of the remaining fonts with all rotations were used as test fonts.

IV.3. GENERALIZATION EXPERIMENTS WITHOUT PREPROCESSOR

To obtain a basis for comparison, the original fonts were directly trained without a preprocessor. Since each neural network in the NeuralWorks simulator can have a maximum of 500 elements, it was possible to use the $16 \times 16 = 256$ pixel representation for each letter as the input feature vector d .

$$d = \begin{vmatrix} d_1 \\ d_2 \\ \vdots \\ d_{256} \end{vmatrix} \quad (4.3.2)$$

In pre-experiments, several different neural networks were tested with the input d. The neural network configuration shown in Figure 11. was selected , and designated Neural Network#I.

Figure 11. Neural Network# I.

The input layer consists of 256 elements, each receiving one of the pixels in a 16x16 font; 32 elements are in the single hidden layer; and 26 elements are in the output layer, corresponding to the 26 letters of the alphabet.

1)For generalization on rotations:

After 150,000 training iterations, the network achieved an accuracy of 96.1% on the training set and yielded a 69.2% accuracy on the test set.

2)For generalization on shifts:

After 20,000 training iterations (the same network used as above), the network achieved an accuracy of 96.1% on the training set and yielded a 2% accuracy on the test set.

3)For generalization on fonts:

a)For input set III: After 150,000 training iterations, the network achieved an accuracy of 96.1% on the training set, and yielded a 71.1% accuracy on the testing set.

b)For input set IV: After 200,000 training iterations, the network achieved an accuracy of 93.4% on the training set, and yielded a 65.6% accuracy on the testing set.

The data for above experiments is collected together in the Table I. The training processes including snap-shots, being taken every 50,000 iterations, are shown in Appendix D.

TABLE I

TRAINING AND TESTING RESULTS FOR NEURAL NETWORK
(16X16 INPUT) WITHOUT PREPROCESSOR

	Training iterations	Data set (16x16)	Training accuracy / #training letters	Testing accuracy /#testing letters
Generalization on rotations	150,000	set I	96.1% / 1040	69.2% / 2496
Generalization on shifts	100,000	set II	96.1% / 104	2% / 208
Generalization on fonts	100,000	set III	96.1% / 104	71.1% / 104
Generalization on fonts	200,000	set IV	93.4% / 1768	65.6% / 1768

IV.4. GENERALIZATION EXPERIMENTS WITH PREPROCESSOR

The 16x16 pixel representation did not provide sufficient resolution for describing the preprocessor sampling window geometry. Therefore, every pixel in the 16x16 representation was expanded to 4 pixels so that a larger data array (32x32 pixels) could be used to represent better sampling windows (see Figure 12.).

Figure 12. Example of pixel expansion. (a) 16x16 grid letter "A"; (b) 32x32 grid letter "A". (CHICACO font shown)

After converting each letter, the results of 32x32 pixel data sets called expanded input set I, set II, set III, set IV and set V. A 32x32 grid, two-dimensional digital Fourier transform was calculated for each. Measurements were taken on these Fourier transform "images" via the sampling windows .

Using the expanded (32x32) representation, 15 approximate annular-ring and 16 pie-wedge windows were used to extract the feature vectors $\mathbf{r}$ for ring and $\mathbf{w}$ for wedge are expressed as:

$$\mathbf{r} = \begin{bmatrix} r_1 \\ r_2 \\ \cdot \\ \cdot \\ \cdot \\ r_{15} \end{bmatrix} \quad \mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \cdot \\ \cdot \\ \cdot \\ w_{16} \end{bmatrix}$$

Both r and w were used as inputs to the modified neural networks.

Figure 13. Approximate annual-ring and pie-wedge windows for the 32x32 data array.

Figure 14. Neural Network # II.

1) For generalization on rotation:

A three layer Neural Network # II was selected, which has 15 elements in the input layer for the $r_1, r_2, r_3, \dots, r_{14}, r_{15}$ input, 32 elements in the hidden layer, and 26 elements in the output layer.

After 150,000 training iterations, this network achieved an accuracy of 98% on the training set, and yielded a 89.7% accuracy on the test set.

2)For Generalization on shifting:

After 100,000 training iterations (same network as above), the network achieved an accuracy of 98% on the training set, and yielded a 98% accuracy on the test set.

Figure 15. Neural Network #III.

3)For generalization on fonts:

A four layer Neural Network # III was selected , with 31 elements in the input layer to accept the vectors r and w independently; the second layer is hidden layer^{# 1} comprising 31 elements; 15 out of the 31 elements are fully interconnected to the 15 elements of the input layer that receive the r inputs, and the remaining 16 elements are

fully interconnected to the 16 elements of the input layer that receives the w inputs. The next is a 30 element hidden layer^{# 2}; The last is the 26 element output layer.

a) For expanded input set III(32x32 pixels): After 100,000 training iterations, the network achieved an accuracy of 98% on the training set, and yielded a 82.7% accuracy on the testing set.

b) For expanded input set IV(32x32 pixels): After 150,000 training iterations, the network achieved an accuracy of 96.1% on the training set, and yielded a 80.8% accuracy on the testing set.

The data for above experiments is collected together in Table II. The training processes including snap-shots, being taken every 50,000 iterations, are shown in Appendix D.

TABLE II

TRAINING AND TESTING RESULTS FOR NEURAL NETWORK
(32X32 INPUT) WITH PREPROCESSOR

	Training iterations	Data Set (32x32)	Feature vector	Training accuracy / #training letters	Testing accuracy / #testing letters
Generalization on rotations	150,000	expanded set I	r	98% / 1040	89.7% / 2496
Generalization on shifts	100,000	expanded set II	r	98% / 104	98% / 208
Generalization on fonts	100,000	expanded set III	r & w	98% / 104	82.7% / 104
Generalization on fonts	150,000	expanded set IV	r & w	96.1% / 1768	80.8% / 1768

IV.5. CONCLUSIONS FROM EARLY EXPERIMENTS

IV.5.1. For Generalization on Rotation

a) Inclusion of the Fourier transform preprocessor resulted in generalization performance improving from 69.2% to 89.7%.

b) With the preprocessor, the results of training and testing on rotation in $+0^\circ$ to $+90^\circ$ and 0° to -90° are identical except for the letter "W" or "M" on some fonts because of their similarity when rotated 180° . Since the results are the same for the first and second quadrant in subsequent experiments, no training will be performed in the 0° to -90° quadrant space (the first quadrant).

c) There is potential for obtaining further improvement by increasing the resolution of font representation so that more precise window shapes can be realized and larger numbers of windows can be defined.

IV.5.2. For Generalization on Shifting

a) Inclusion of the Fourier transform preprocessor resulted in a generalization performance improving from 2% to 98%.

b) These results come close to real world application requirements. Translation invariance on shifting is an automatic by product of the Fourier transform and this experiment verified the results. No more experiments were performed to test generalizing ability of the neural network on pattern shifting.

IV.5.3. For Generalization on Fonts

a) Including the Fourier transform preprocessor and combining the annular-ring, pie-wedge data and adding an additional hidden layer resulted in generalization performance improving from 71.1% to 82.7%.

b) There is a potential for obtaining further improvement via increasing font representation resolution, as mentioned in IV.5.1., c).

CHAPTER V

FURTHER EXPERIMENTS AND FUTURE WORK

V.1. EXTENDED EXPERIMENTS

During the time period when the work in chapter IV was carried out, a research team at the Serology Laboratory of University of California at Davis completed a FFT and annular-ring and pie-wedge window program. They have shared their program with Professor Lendaris at Portland State University, and this was made available for my use in March, 1991. The new program, which reads in a 256x256 8bit image file, generates a two dimensional FFT and applies selected window geometries, making it possible to use significantly higher resolution images.

Image data for this program was created by using a video camera, whose signal was fed into a PC and 256x256 data arrays were generated for more experiments.

V.2. DATA PREPARATION

A new set of fonts, TIMES, PALATINO, HELVETICA and COURIER were printed out on a Postscript Printer, including a selected set of rotations for each letter. The video camera/PC system was used to generate a 256x256 data file for each letter.

The rotations selected for these experiments were counter-clockwise 0° , $+7^{\circ}$, $+15^{\circ}$, $+21^{\circ}$, $+30^{\circ}$. A set of annular-ring sampling windows containing 32 annular-rings of equal thickness was generated. Also a set of wedge sampling windows containing 120 wedges of equal angle (3°) was generated. The two-dimensional Fourier transform for each letter was calculated, and the sampling window measurements were

taken for each , and placed into an appropriate data file for use as input to the neural network.

Figure 17. 256x256 grid approximate annual-ring and pie-wedge widows (applied on FFT spectrum of letter "A" from font TIMES)

Figure 18. Neural Network #IV

The three layer neural network show in Figure 18 was selected for the test of the generalization on rotation. This network has 32 elements in the input layer for the $r_1, r_2, r_3, \dots, r_{31}, r_{32}$ input data, 32 elements in the hidden layer, fully interconnected to the input layer, and 26 elements in the output layer, fully interconnected to the hidden layer.

1) Input data set V for generalization on rotations:

After 150,000 training iterations, the training sets of 0° , 15° and 30° of the four fonts reached 100% accuracy. The network was tested with the $+7^\circ$, $+21^\circ$ rotations and yielded 94.9% correct generalization performance.

2) For generalization on fonts:

A four layer Neural Network #V, was selected for the test of generalization on fonts. This network has 152 elements in the layer, 32 elements receive the annular-ring data, and the remaining 120 elements receive the pie-wedge data; 152 elements in the first hidden layer, 32 elements are fully interconnected to the elements in the input layer receiving the annular-ring data and the remaining 120 elements fully interconnected to the elements in the input layer receiving the wedge window; 50 elements in the second hidden layer; and 26 element in the output layer. The TIMES and PALATINO fonts were used as the training sets. The HELVETICA and COURIER fonts were used as the testing set.

a) After 100,000 training iteration, the training sets of the two fonts with 0° rotations reached 100% accuracy and the remaining two fonts with 0° rotations yield 88.5% accuracy on the testing fonts (input data set VI).

b) After 150,000 training iteration, the training sets of the two fonts with all rotations reached 100% accuracy and the remaining two fonts with all rotations yield 88.5% accuracy on the testing fonts (input data set VII).

The data for above experiments is collected together in the Table III. And the training processes including snap-shots, being taken every 50,000 iterations, are shown in Appendix D.

Figure 19. Neural Network # V.

TABLE III

TRAINING AND TESTING RESULTS FOR NEURAL NETWORK
(256X256 INPUT) WITH PREPROCESSOR

	Training iterations	Data set (256x256)	Training accuracy /#training letters	Testing accuracy /# training letters
Generalization on rotations	150,000	set V	100% / 312	94.9% / 208
Generalization on fonts	100,000	set VI	100% / 52	88.5% / 52
Generalization on fonts	150,000	set VII	100% / 260	88.5% / 260

V.3. CONCLUSION FROM LATEST EXPERIMENTS

Achieving an 88.5% generalization on fonts is based on training with only two fonts. A more extensive training set could allow the neural network to perform better generalization. Unfortunately, the process of generating the font data proves a much time consuming work with the equipment available to the author, so further experiments are not possible at present.

A clear item for the future work is to generate more training and testing data sets, and to train the network more statistically to yield better generalization.

SELECTED BIBLIOGRAPHY

- Fu, K.S., "Applications of Pattern Recognition", CRC Press Inc. 1982.
- Hebb, D. O. 1949. "The Organization of Behavior" New York: Wiley.
- Hopfield, J. J. 1982. "Neural Networks and Physical Systems with Emergent Collective Computational Abilities". Proceedings of the National Academic of Sciences, USA, 79, 2554-2558.
- Lendaris, George G. & Stanley, G. L. 1970. "Diffraction-pattern Sampling For Automatic Pattern Recognition", IEEE Vol. 58, No2, pp. 198-216.
- McCulloch, W. S. & Pitts, W. 1943. "A Logical Calculus of the Ideas Immanent in Nervous Activity" Bulletin Of Mathematical Biophysics, 5, 115-33.
- Minsky, M., & Papert, S. 1969. "Perceptrons". Cambridge, MA: MIT Press
- Press, William H. [et al] 1989. "Numerical Recipes: the art of scientific computing " New York, N.Y. Cambridge Press.
- Rosenblatt, F. 1959. "Two Theorems of Statistical Separability in the Perceptron". In Mechanisation of Thought Processes: Proceedings of a Symposium Held at the National Physical Laboratory, November 1958. Vol 1 pp.421-456. London:HM Stationery Office.
- Rosenblatt, F.1962. "Principles of Neurodynamics". New York: Spartan.
- Rumelhart, D. E., McClelland & the Research Group 1986. "Parallel Distributed Processing: Explorations in the Microstructure of Condition", Vol. 1, Rumelhart, McClelland, et al, The MIT Press, Cambridge, MA.
- Widrow B., & Hoff, M. E. 1960. "Adaptive Switch Circuits". Institute of Radio Engineers, Western Electric Show and Convention, Convention Record, Part 4, 96-104.
- Widrow, B., 1962. "Self Organization System". Spartan Books, Washington D.C., pp436-461

APPENDIX A

16x16 GRID ENGLISH ALPHABET FONTS

16x16 GRID ENGLISH ALPHABET FONTS

The names of the eight 16x16 grid fonts are as follows*:

- i> TIMES
- ii> PALATINO
- iii> HELVETICA
- iv> GENEVA
- v> CARTOON
- vi> CHICAGO
- vii> MONACA
- vii> ASCHAM

* The following 16x16 data format was made by Ihab A. Hab.

16x16 grid fonts (left to right) TIMES, PALATINO, HELVETICA
and GENEVA.

16x16 grid fonts (left to right) TIMES, PALATINO, HELVETICA
and GENEVA (continued).

16x16 grid fonts (left to right) TIMES,PALATINO, HELVETICA
and GENEVA (continued).

16x16 grid fonts (left to right) CARTOON, CHICAGO,
MONACO and ASCHAM.

16x16 grid fonts (left to right) CARTOON, CHICAGO,
MONACO and ASCHAM (continued).

16x16 grid fonts (left to right) CARTOON, CHICAGO,
MONACO and ASCHAM (continued).

APPENDIX B

EXAMPLES OF ROTATED AND SHIFTED 16X16 GRID ENGLISH ALPHABET FONTS

EXAMPLES OF ROTATED AND SHIFTED 16X16
GRID ENGLISH ALPHABET FONTS*

- 1- The font GENEVA rotated counter-clockwise to $+45^{\circ}$, $+22.5^{\circ}$,
 $+11.25^{\circ}$ and rotated clockwise to -11.25° , -22.5° , -45° .
- 2- The font GENEVA shifted left 3 pixels and right 3 pixels.

* these rotations and shifts were generated using a C program

16x16 grid font GENEVA rotated counter-clockwise to $+45^{\circ}$, $+22.5^{\circ}$, $+11.25^{\circ}$ and clockwise to -11.25° , -22.5° , -45° .

16x16 grid font GENEVA rotated counter-clockwise to $+45^\circ$, $+22.5^\circ$, $+11.25^\circ$ and clockwise to -11.25° , -22.5° , -45° (continued).

16x16 grid font GENEVA rotated counter-clockwise to $+45^\circ$, $+22.5^\circ$, $+11.25^\circ$ and clockwise to -11.25° , -22.5° , -45° (continued).

16x16 grid font GENEVA shifted 3 pixels left and shifted 3 pixels right .

16x16 grid font GENEVA shifted 3 pixels left and shifted 3 pixels right (continued).

APPENDIX C

256x256 GRID ENGLISH ALPHABET FONTS

256x256 GRID ENGLISH ALPHABET FONTS

1- The 256x256 grid four fonts:

i> TIMES

ii> PALATINO

iii> HELVETICA

iv> COURIER

2- The rotated 256x256 grid font examples.

A	A	A	A
B	B	B	B
C	C	C	C
D	D	D	D
E	E	E	E
F	F	F	F

256x256 grid fonts (left to right) TIMES,PALATINO,
HELVETICA and COURIER.

G	G	G	G
H	H	H	H
I	I	I	I
J	J	J	J
K	K	K	K
L	L	L	L

256x256 grid fonts (left to right) TIMES, PALATINO,
HELVETICA and COURIER(continued).

M	M	M	M
N	N	N	N
O	O	O	O
P	P	P	P
Q	Q	Q	Q
R	R	R	R

256x256 grid fonts (left to right) TIMES, PALATINO,
HELVETICA and COURIER (continued).

S	S	S	S
T	T	T	T
U	U	U	U
V	V	V	V
W	W	W	W
X	X	X	X

Y	Y	Y	Y
Z	Z	Z	Z

E

E

E

E

The rotated 0° , 7° , 15° , and 30° letter "A" in font TIMES
(top to bottom).

APPENDIX D

TRAINING ACCURACY PROFILE FOR DATA SETS I-VII

Training accuracy profile for data set I (used in generalization on the 16x16 rotation experiments).

Training accuracy profile for data set III (used in generalization on the 16x16 shift experiments).

Training accuracy profile for data set III (used in generalization on the 16x16 across-fonts experiments).

Training accuracy profile for data set IV (used in generalization on 16x16 across-fonts experiments).

Training accuracy profile for expanded data set I (used in generalization on 32x32 font rotation experiments).

Training accuracy profile for expanded data set II (generalization on 32x32 font shift experiments).

Training accuracy profile for expanded data set III (used in generalization on 32x32 across-fonts experiments).

Training accuracy profile for expanded data set IV (used in generalization on 32x32 across-fonts experiments).

Training accuracy profile for data set V (used in generalization on 256x256 font rotation experiments).

Training accuracy profile for data set VI (used in generalization on 256x256 across-fonts experiments).

Training accuracy profile for data set VII (used in generalization on . 256x256 across-fonts experiments).