To the Graduate Council:

I am submitting herewith a thesis written by Martin Thao Do entitled A Pattern Recognition System Using Artificial Neural Networks and Wavelets for Taxonomic Identifications. I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Science.

Dr. Bruce Maclennan, Major Professor

We have read this thesis
and recommend its acceptance:

Accepted for the Council:

Associate Vice Chancellor
and Dean of the Graduate School

Pattern Recognition System Using Artificial Neural Networks and Wavelets for Taxonomic Identifications

A Thesis

Presented for the

Master of Science Degree

The University of Tennessee, Knoxville

Martin Thao Do

December 1996

Acknowledgments

I would like to thank my parents, who have always been there for me, for giving me support and encouragements throughout my life and for helping me through all the tough times, my siblings Marriane, Theresa, Mary, Ann, and Anthony for putting up with me through the years and my nephews Matthew and Sam for reminding me about what is really important in life and Steve and Jill Plichta, Mark Oehler, Karlyn Ammons and Vivie Babb for their friendship which has kept me sane throughout the masters program.

On the educational and professional side, thanks go to my advisor Dr. Maclennan whose guidance has been invaluable throughout the research, Dr. Gerry Bunick for his support, Dr. Straight and Dr. Riechert for agreeing to evaluate my research, Joel Harp who gave me the idea for this research and whose advices and work in obtaining the data for the research has been crucial, Kim Norris for her time in obtaining the unknown data set, Dr. Paris Lambdin for allowing me to use his equipments, and Dr. Charles Dondale for providing the samples.

Abstract

This research focuses on the development of a pattern classification system that is capable of identifying spiders to genus and species. This system attempts to emulate the eyes and expertise of arachnologists. It takes as input images of epygina, representative of a genus or species of spider, encodes it by means of a Daubechies 4 wavelet transform, and uses a fast and efficient cascade correlation artificial neural network to identify specimens of that genus or species. If the samples are chosen well then the system is able to perform accurate identification of members of the specified genus or species. In tests involving samples from the family Lycosidae, the system created by this research achieved an accuracy of 100% in the identification of genera and up to 88% accuracy in the identification of species.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Fast and accurate identification of an organism is a critical first step in a biological investigation. An incorrect or slow identification can lead to confusion and wasted resources in the best case and loss of life in the worst case. For example, incorrect or slow identification of pests can cause loss of valuable crops or prevent a physician from giving timely and proper treatment to a patient suffering from an allergic reaction or a poisonous insect bite.

Despite the importance of accurate identification of organisms, access to this capability is out of the reach for many people that need it. One of the reasons for the inaccessibility is the high cost of taxonomic identifications. In order to perform accurate identifications, taxonomists require access to extensive reference

collections and literatures which are expensive to maintain. Also, even when the funds are available for taxonomic identifications, the process is painstakingly slow. The process involves mailing the samples to the expert, so the response time is often too long for a physician who is trying to determine treatment for a poisonous insect bite. Furthermore, the number of experienced taxonomists is decreasing at an alarming rate, which makes the odds of obtaining an accurate identification low. In fact, Stephanie Pain ,in her article [9] for the New Scientist, she reported that "From its investigation of research in systematic biology, the House of Lords Select Committee on Science and Technology concludes that systematics biology is an endangered subject". This problem is not unique to England. John Rawlins, an entomologist at the Carnegie Museum of Natural History in Pittsburgh, predicted in an article [7] by Diana Nelson Jones, that entomologists "will soon blink out entirely."

This research is concerned with developing an expert system capable of "capturing" the expertise of experienced taxonomists. More specifically, it is concerned with capturing the expertise of experienced arachnologists. Spiders were chosen as subjects because of the availability of samples and their importance to fields such as medicine, agriculture, and biological research.

1.2 Components of the Pattern Classification System

1.2.1 Artificial Neural Network

The "heart" of this pattern classification system is an Artificial Neural Network (ANN). An ANN is a model of computation that is capable of "learning" and obtaining new knowledge. The reason the ANN was selected instead of symbolic Artificial Intelligence (AI) methods was because of the fine grain with which it represents knowledge. The fine grain knowledge representation makes it more flexible and less brittle than a symbolic AI system, which represents knowledge at a much coarser level of granularity which can be represented by words.

1.2.2 Wavelet Transforms

Since spiders can be identified at the genus and species levels by the shape of their epigynum, the logical input to the ANN would be images of the epigynum of the samples. However, even the smallest recognizable image of the epigynum would overwhelm most available computers simulating the computing intensive ANN. One solution to this problem is to reduce the dimension of the images by eliminating unnecessary details. This can be done by encoding the image by means of a wavelet transform, which essentially breaks it down into its component resolutions. The higher resolution components, which are often not needed for identification, can be eliminated, thus the number of inputs to the ANN is reduced.

1.3 Summary

Although fast and accurate identification of organisms is critical for a biological investigation, it is often unavailable to many biologists because of cost and the difficulty in quickly reaching a dwindling number of taxonomists. A good pattern classification system could allow non-specialists to perform affordable, fast and accurate identifications. The focus of this research is in developing a pattern recognition for identifying spiders. Spiders were chosen because of the availability of samples and their importance to fields such as medicine, agriculture, and biological research. However, it can be adapted to identifying other organisms as well.

Chapter 2

Wavelets

This chapter discusses the relevant concepts of wavelet theory that will be used in the work presented in later chapters of this thesis.

2.1 Introduction

According to Graps [6] "Wavelets are functions that satisfy certain mathematical requirements and are used in representing data or other functions." This idea is similar to Joseph Fourier's discovery in 1807 that sine and cosine functions could be superimposed to represent other functions. However, there exist a number of major differences between Fourier's discovery and wavelets. In wavelet analysis the algorithm used processes the data at varying scales or resolutions. The resolution seen is directly related to the size of the window used to view the data so one can determine the resolution of the data to look at by selecting a viewing window of

proper size. Another difference is that Fourier analysis is restricted to using sine and cosine functions which are by definition non-local and therefore are poorly equipped to approximate sharp spikes. Wavelet analysis on the other hand can use functions neatly contained in finite domains to approximate data with sharp discontinuities.

2.2 History

Wavelet analysis began with Joseph Fourier's theories of frequency analysis in 1807. Fourier suggested that any periodic function $f(x)$ over the period of 2π can be represented by the sum of its Fourier series shown below.

$$a_0 + \sum_{k=1}^{\infty}(a_k \cos(kx) + b_k \sin(kx)) \tag{2.1}$$

Where a_0, a_k, b_k are calculated as follows.

$$a_0 = \frac{1}{2\pi}\int_0^{2\pi} f(x)dx \tag{2.2}$$

$$a_k = \frac{1}{\pi}\int_0^{2\pi} f(x)\cos(kx)dx \tag{2.3}$$

$$b_k = \frac{1}{\pi}\int_0^{2\pi} f(x)\sin(kx)dx \tag{2.4}$$

Fourier's study of frequency analysis eventually inspired the discovery of scale analysis. Scale analysis is the analysis of a given function $f(x)$ by creating mathematical structures that vary in scale.

It was not until 1909 that wavelets were first mentioned in the thesis of A. Haar. The Haar wavelets have the desirable property of having compact support, which means that they vanish outside of a finite interval. Over the years a number of other wavelet families were developed such as the Daubechies, biorthogonal, and Coifflets.

2.3 Differences between Fourier and wavelet transform

The Fourier transform is a valuable tool for analyzing the frequency content of a signal in the time domain. It works by "transforming" a function in the time domain to the frequency domain. The resulting coefficients of the transformed function provide the contribution of each sine and cosine function at each frequency.

If $f(t)$ is a periodic signal then the Fourier transform can give an accurate analysis of the frequency content. However, if $f(t)$ is nonperiodic then the summation of the periodic sine and cosine functions no longer gives an accurate representation of the signal. One solution to this problem is to use a windowed Fourier transform. A windowed Fourier transform breaks $f(t)$ into sections and analyzes

each section separately. The effect is to localize the signal in time.

Although the window allows for the analysis of nonperiodic signals, it is not a perfect solution. Figure 2.1 shows a Fourier transform in which the sine or cosine function is truncated to fit in a window of a specific width. Since a single window, or set of basis functions, is used for all frequencies in the windowed Fourier transform, the analysis will yield the same resolution everywhere in the time-frequency plane.

The best analysis is one that will allow the window size to change to accomodate the differing portions of a signal. For example, if a signal contains discontinuities, then short, high frequency basis functions are needed to isolate it. However, if a detailed frequency analysis is desired, then long, low frequency basis functions are needed. Wavelet transforms have an infinite set of possible basis functions which are capable of accessing information invisible to other analysis techniques such as Fourier analysis. Figure 2.2 shows how a wavelet transform decompose a signal.

2.4 How wavelets work

Since the Haar basis is the simplest wavelet basis, it is a good starting point for understanding how a function can be decomposed using wavelets. The following example, taken from [3], will be used to explain how wavelets work. Let the

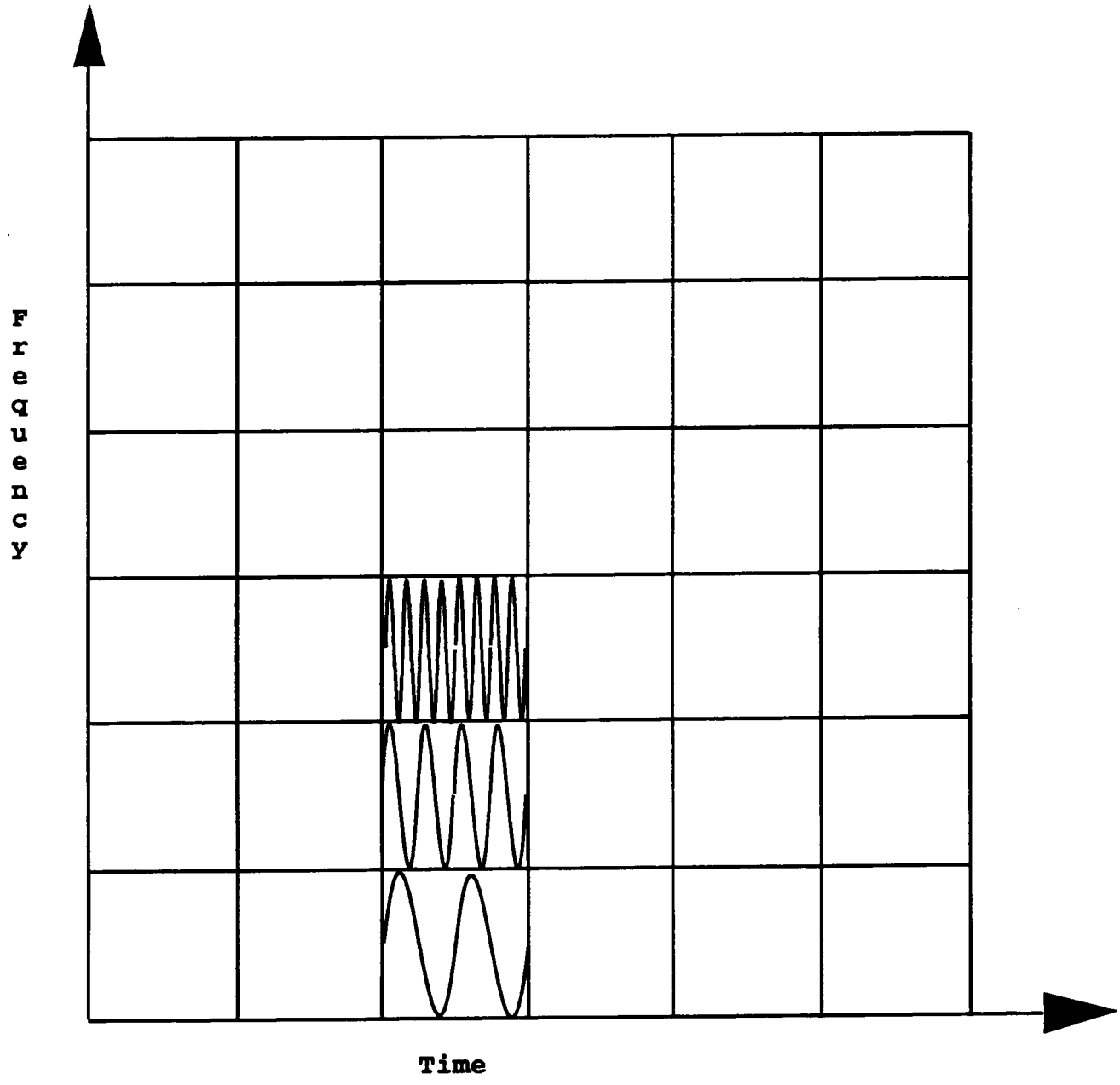

Figure 2.1: Fourier basis functions, time-frequency tiles, and coverage of the time-frequency plane (figure adapted from [3])

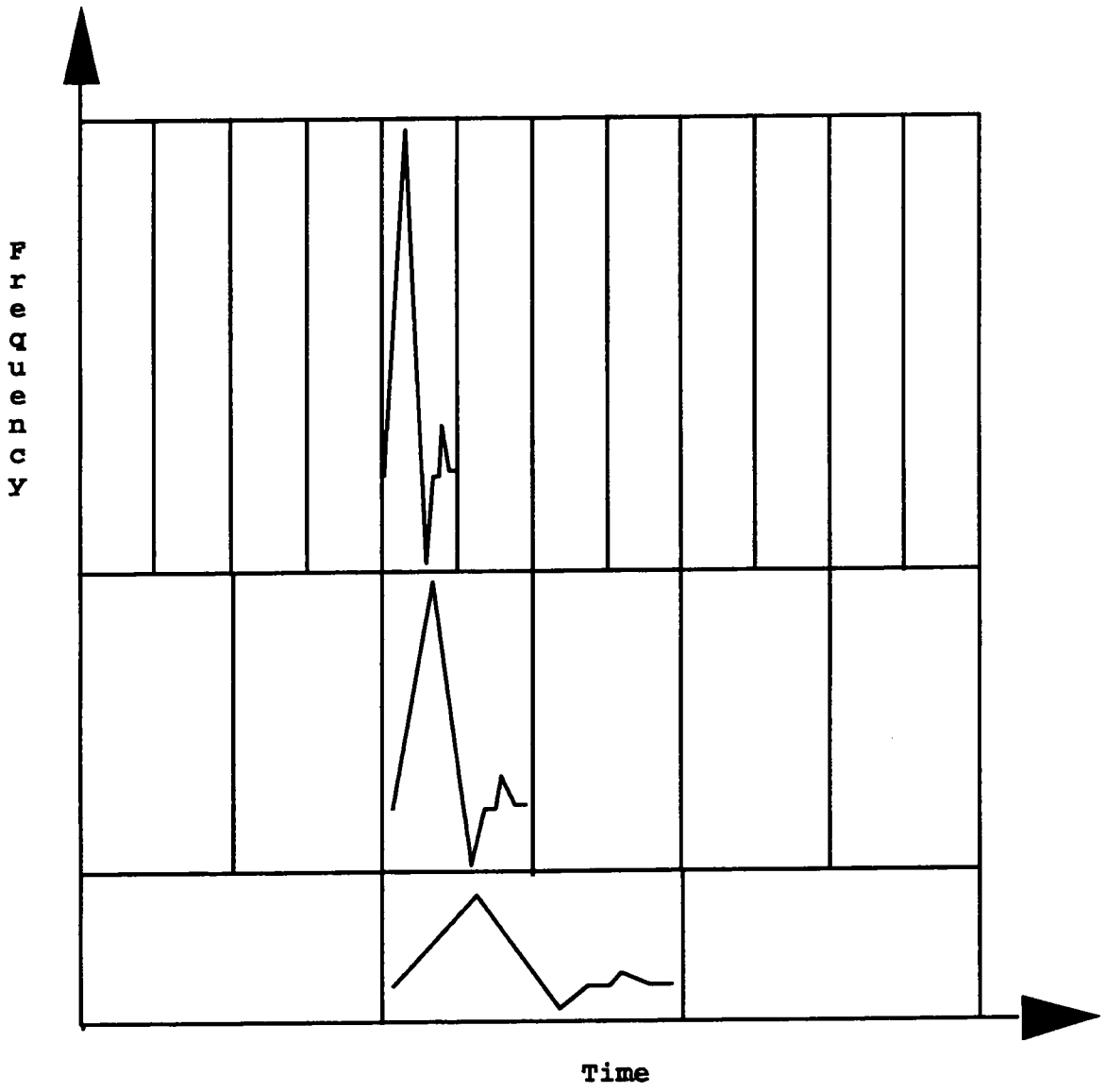

Figure 2.2: Daubechies wavelet basis functions, time-frequency tiles, and coverage of the time-frequency plane (figure adapted from [3])

following vector represent a one dimensional image with a resolution of 4 pixels.

$$\begin{bmatrix} 8 & 4 & 1 & 3 \end{bmatrix} \tag{2.5}$$

This one dimensional image can be viewed as constant functions over the open intervals $[0,\frac{1}{4})$, $[\frac{1}{4},\frac{1}{2})$, $[\frac{1}{2},\frac{3}{4})$, and $[\frac{3}{4},1)$. The vector space containing these intervals is called V^2. More generally, one dimensional images with 2^j pixels can be viewed as piecewise-constant functions on the interval $[0, 1)$ contained in the vector space V^j. The interval is divided equally into 2^j different pieces or vectors. Since the vectors in V^j are all functions defined on the unit interval, every vector in V^j is also in V^{j+1}.

$$V^0 \subset V^1 \subset V^2 \subset \cdots \tag{2.6}$$

The basis functions for the spaces V^j, or scaling functions, denoted by 'ϕ', are shown below [3].

$$\phi_i^j(x) = \phi(2^j x - i) \quad i = 0, \cdots, 2^j - 1 \tag{2.7}$$

$$\phi(x) = \begin{cases} 1 & \text{for } 0 \leq x < 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.8}$$

For V^2 the Haar basis functions can be represented by the graphs in Figure 2.3

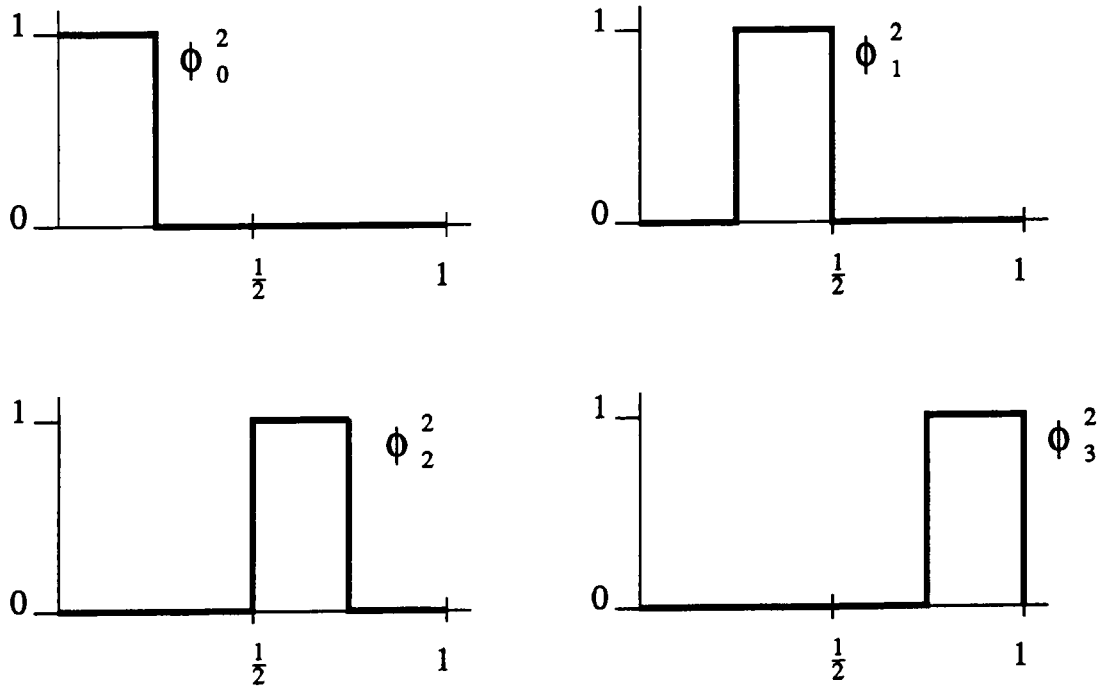

Figure 2.3: Haar basis functions for V^2

It is now necessary to define another vector space W^j that will be the orthogonal complement of V^j in V^{j+1}. So the vector space W^j is the space of all functions in V^{j+1} that are orthogonal to all functions in V^j for the following inner product [3].

$$< f, g >= \int_0^1 f(x)g(x)dx \qquad f, g \in V^j \tag{2.9}$$

The functions spanning W^j are called wavelets and are represented by the symbol

$\psi_i^j(x)$. The Haar wavelets can be defined by [3].

$$\psi_i^j(x) = \psi(2^j x - i) \quad i = 0, \cdots, 2^j - 1 \tag{2.10}$$

$$\psi(x) = \begin{cases} 1 & 0 \leq x < \frac{1}{2} \\ -1 & \frac{1}{2} \leq x < 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.11}$$

Figure 2.4 show the wavelets spanning W^1.

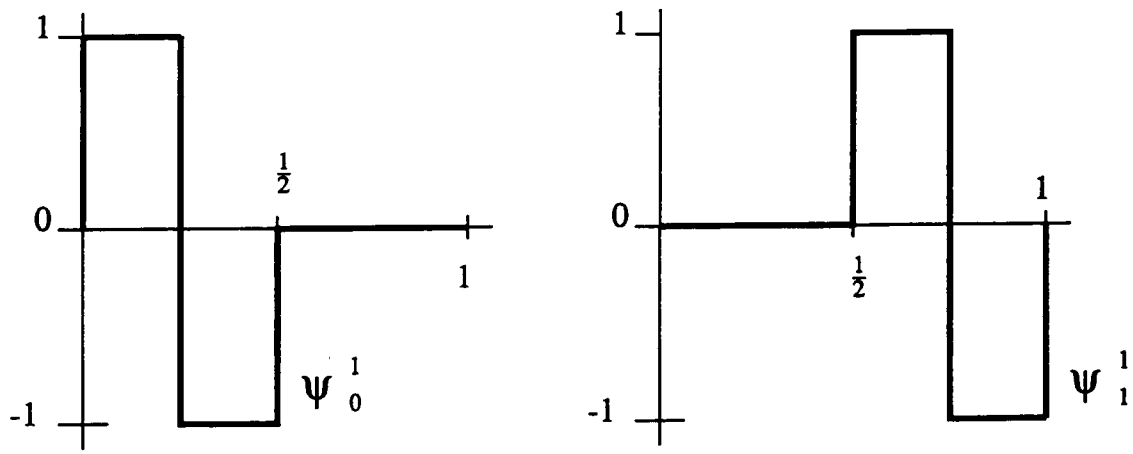

Figure 2.4: Haar wavelets spanning W^1

Using the tools previously defined, it is now possible to perform wavelet transformation on the one dimensional image described earlier. The first step of the transformation is to express the image as a combination of the basis functions in the V^2 vector space.

$$image = c_0^2\phi_0^2(x) + c_1^2\phi_1^2(x) + c_2^2\phi_2^2(x) + c_3^2\phi_3^2(x) \tag{2.12}$$

In equation 2.12 c_0^2, c_1^2, c_2^2, *and* c_3^2 correspond to the pixel values in the original image. Another way to express this expression is shown in figure 2.5.

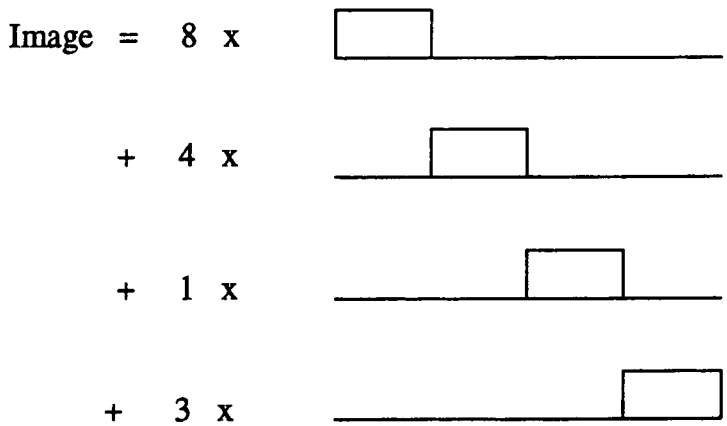

Figure 2.5: Original image expressed as linear combination of box basis functions in V^2. (Figure from [3])

The image can also be written in terms of the basis functions in V^1 and W^1.

$$image = c_0^1\phi_0^1(x) + c_1^1\phi_1^1(x) + d_0^1\psi_0^1(x) + d_1^1\psi_1^1(x) \tag{2.13}$$

Equation 2.13 expresses the image as a lower resolution image along with detail information to reconstruct the original image. The lower resolution image is represented by $(c_0^1\phi_0^1(x) + c_1^1\phi_1^1(x))$, which is in terms of V^1 basis functions. The detail information is represented by $(d_0^1\psi_0^1(x) + d_1^1\psi_1^1(x))$, which is in terms of W^1 basis functions.

In order to obtain the coefficients for the lower resolution image, start by averaging the pixels, pairwise to get:

$$\begin{bmatrix} 6 & 2 \end{bmatrix} \tag{2.14}$$

where $c_0^1 = 6$ and $c_1^1 = 2$. Then to ensure that no information from the original image is lost, additional "detail" coefficients, which capture information missing from the lower resolution image, are created. In this example, the first "detail" coefficient, d_0^1, is a 2 since in the lower resolution image that was computed, the average is 2 less that 8 and 2 more than 4. The second "detail" coefficient, d_1^1, is a -1 since $2 + (-1) = 1$ and $2 - (-1) = 3$. These "detail" coefficients allow the recovery of the original image. The image now becomes :

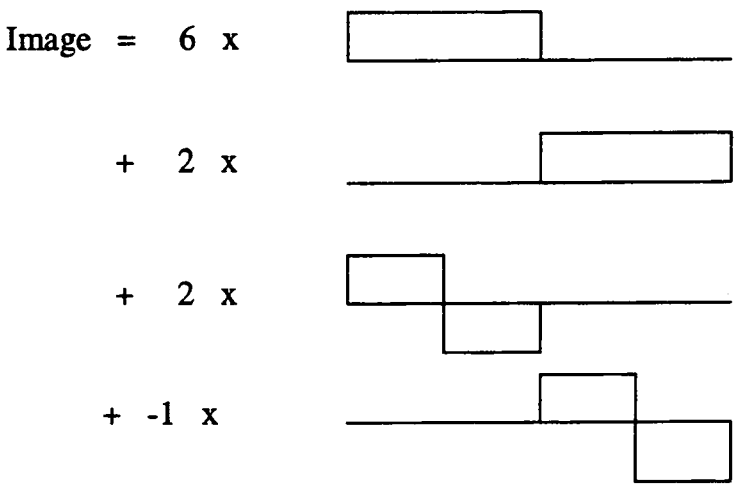

Figure 2.6: image expressed as linear combination of box basis functions in V^1 and W^1. (Figure from [3])

Note that the first two functions in figure 2.6 have become wider to reflect the

fact that they are associated with a lower resolution image. Likewise the last two functions in figure 2.6 have changed since they represent detail information.

Finally, the image can be expressed in terms of the basis functions in V^0, W^0, and W^1.

$$image = c_0^0\phi_0^0(x) + d_0^0\psi_0^0(x) + d_0^1\psi_0^1(x) + d_1^1\psi_1^1(x) \tag{2.15}$$

Equation 2.15 can be obtained by further reducing the low resolution image in equation 2.13, $(c_0^1\phi_0^1(x) + c_1^1\phi_1^1(x))$, to an even lower resolution image, $(c_0^0\phi_0^0(x))$, which is in terms of the V^0 basis function, and detail information, $(d_0^0\psi_0^0(x))$, which is in terms of the W^0 basis function. The coefficient c_0^0 can be determined by performing pairwise averaging on c_0^1 and c_1^1 to get $c_0^0 = 4$. Then d_0^0 can be found by noticing that c_0^0 is 2 less than $c_0^1 = 6$ and $c_1^1 = 2$ so $d_0^0 = 2$. Equation 2.15 can be represented graphically in figure 2.7.

Once again, the first function in figure 2.7 has grown in size due to the fact that it was obtained by averaging the pixels c_0^1 and c_1^1 in the previous image and is therefore lower in resolution. Similarly, the second function has also changed since it is associated with detail information. The four coefficients c_0^0, d_0^0, d_0^1, and d_1^1 are the Haar wavelet transform of the original image and the corresponding four functions make up the Haar basis for V^2.

The Haar wavelet transformation can be summarized by table 2.1

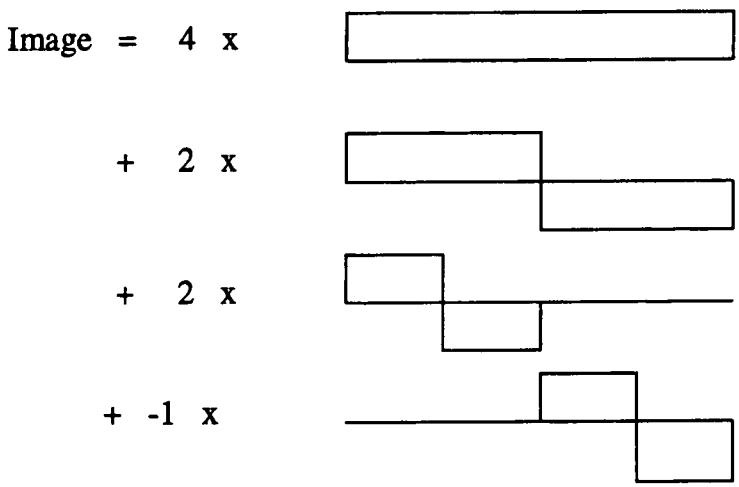

Figure 2.7: Image expressed as linear combination of box basis functions in V^0, W^0, and W^1. (Figure from [3])

Table 2.1: Summary of the Haar wavelet transform. (Adapted from [3])

resolution	image	detail coefficients
4	8 4 1 3	
2	6 2	2 −1
1	4	2

In each case, a higher resolution image can be represented by a lower resolution image and detail coefficients. The higher resolution image can be reconstructed by recursively adding and subtracting the detail coefficients from the lower resolution image.

In order to keep the explanation simple, the basis functions were not normalized. If they were L_2 normalized then they will change from

$$\phi_i^j(x) = \phi(2^j x - i) \tag{2.16}$$

$$\psi_i^j(x) = \psi(w^j x - i) \tag{2.17}$$

to

$$\phi_i^j(x) = 2^{\frac{j}{2}}\phi(2^j x - i) \tag{2.18}$$

$$\psi_i^j(x) = 2^{\frac{j}{2}}\psi(w^j x - i) \tag{2.19}$$

Normalization changes the wavelet transform. Instead of dividing the sum and difference of neighboring values by two it will be divided by $\sqrt{2}$.

The wavelet decomposition of the image can be summarized as follows.

$$image = c_0^0\phi(x) + \sum_{j=0}^{n-1}\sum_{i=0}^{2^j-1} d_i^j\psi_i^j(x) \tag{2.20}$$

It can also be written in matrix form for the current example as :

$$\begin{bmatrix} 8 \\ 4 \\ 1 \\ 3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & -1 & 0 \\ 1 & -1 & 0 & 1 \\ 1 & -1 & 0 & -1 \end{bmatrix} \begin{bmatrix} c_0^0 \\ d_0^0 \\ d_0^1 \\ d_1^1 \end{bmatrix} \tag{2.21}$$

$$\begin{bmatrix} c_0^0 \\ d_0^0 \\ d_0^1 \\ d_1^1 \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \\ 2 \\ -1 \end{bmatrix} \tag{2.22}$$

2.5 Multiresolution Analysis

It was indicated earlier that $V^0 \subset V^1 \subset V^2 \subset \cdots$ where V^j are vector spaces. As j increases, the resolution of the scaling functions, denoted by $\Phi^j(x)$, in V^j also increases. It was also indicated that W^j, the orthogonal complement of V^j in V^{j+1}, contains all functions in V^{j+1} that are orthogonal to all functions in V^j under a chosen inner product. The basis functions for W^j, denoted by $\Psi^j(x)$, are known as wavelets.

Nested vector spaces indicate that the scaling functions must be refinable. Therefore, there must exist a constant matrix P^j such that

$$\Phi^{j-1}(x) = \Phi^j(x)P^j \tag{2.23}$$

where

$$\Phi^j(x) = \left[\begin{array}{ccc} \Phi_0^j(x) & \cdots & \Phi_{M-1}^j(x) \end{array} \right] \quad M = \text{dimension of } V^j \tag{2.24}$$

These equations show that a scaling function at level $j-1$ is expressible as a linear combination of finer scaling functions at level j.

W^{j-1} is also by definition a subspace of V^j, so $\psi^{j-1}(x)$ can be written as follows:

$$\psi^{j-1}(x) = \Phi^j(x)Q^j \tag{2.25}$$

where Q^j is a constant matrix that refines $\phi^j(x)$ to $\psi^{j-1}(x)$. Like scaling functions, $\psi^{j-1}(x)$ is expressible as a linear combination of finer scaling functions at level j.

Combining equations 2.23 and 2.25 gives :

$$\left[\begin{array}{c|c} \Phi^{j-1} & \psi^{j-1} \end{array} \right] = \Phi^j \left[\begin{array}{c|c} P^j & Q^j \end{array} \right] \tag{2.26}$$

The P^n and Q^n matrices combine to form the synthesis filters because they allow the recovery of the original coefficients C^n from C^{n-1} and the associated detail coefficients D^{n-1}. The synthesis process can be performed by the following calculation:

$$C^n = P^n C^{n-1} + Q^n D^{n-1} \tag{2.27}$$

The synthesis filters for the scaling functions in V^{n-1} and wavelets in W^{n-1} can be made from the scaling functions in V^n. For example, for $n = 2$ the unnormalized filters are described by figure 2.8. The normalized filters may be obtained by multiplying P^n and Q^n by $\frac{1}{\sqrt{2}}$.

In examining multiresolutional analysis it is necessary to discuss the decomposition process as well as the synthesis process. Let C^n be a function in some scaling function space V^n with M coefficients. C^n can be written as $C^n = \begin{bmatrix} C_0^n & C_1^n & C_2^n & \cdots & C_{n-1}^n \end{bmatrix}^T$. The decomposition of C^n to a lower resolution representation C^{n-1} can be done by the following equations.

$$C^{n-1} = A^n C^n \tag{2.28}$$

$$D^{n-1} = B^n C^n \tag{2.29}$$

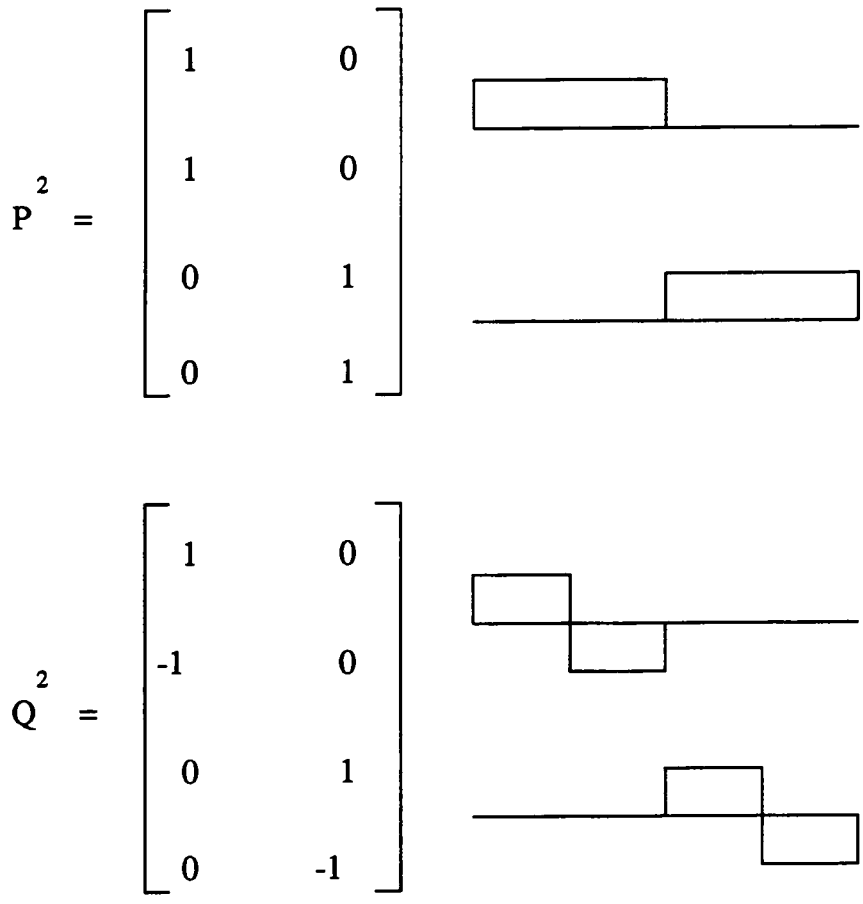

Figure 2.8: P and Q synthesis filter

Where A^n and B^n are decomposition filters that can be derived from the synthesis filters as shown.

$$A^n = \frac{(P^n)^T}{2} \tag{2.30}$$

$$B^n = \frac{(Q^n)^T}{2} \tag{2.31}$$

For normalized filters the equations become :

$$A^n = \frac{(P^n)^T}{\sqrt{2}} \tag{2.32}$$

$$B^n = \frac{(Q^n)^T}{\sqrt{2}} \tag{2.33}$$

The entire decomposition process can be described by figure 2.9

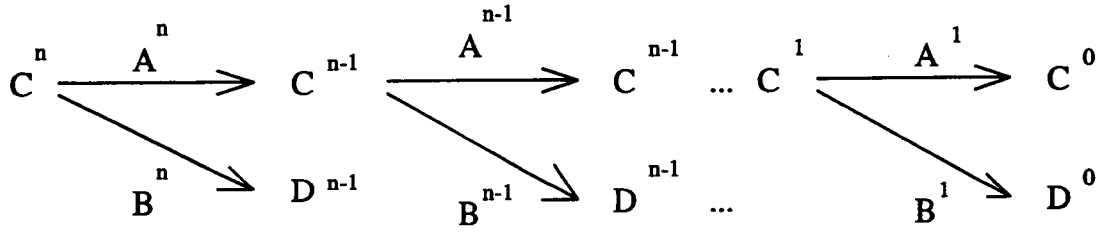

Figure 2.9: Filter bank. (From [3])

2.6 Daubechies Wavelet Basis Function

Unlike the Haar wavelet basis function, it is relatively difficult to obtain the filter coefficients for the Daubechies wavelets. However, according to [8] the filter coefficients can be obtained from estimates from an argument based on the Laurant expansion of the Fourier transform of the scaling function ϕ. Figure 2.10, which was generated using WaveLab [1], show what a Daubechies wavelet looks like.

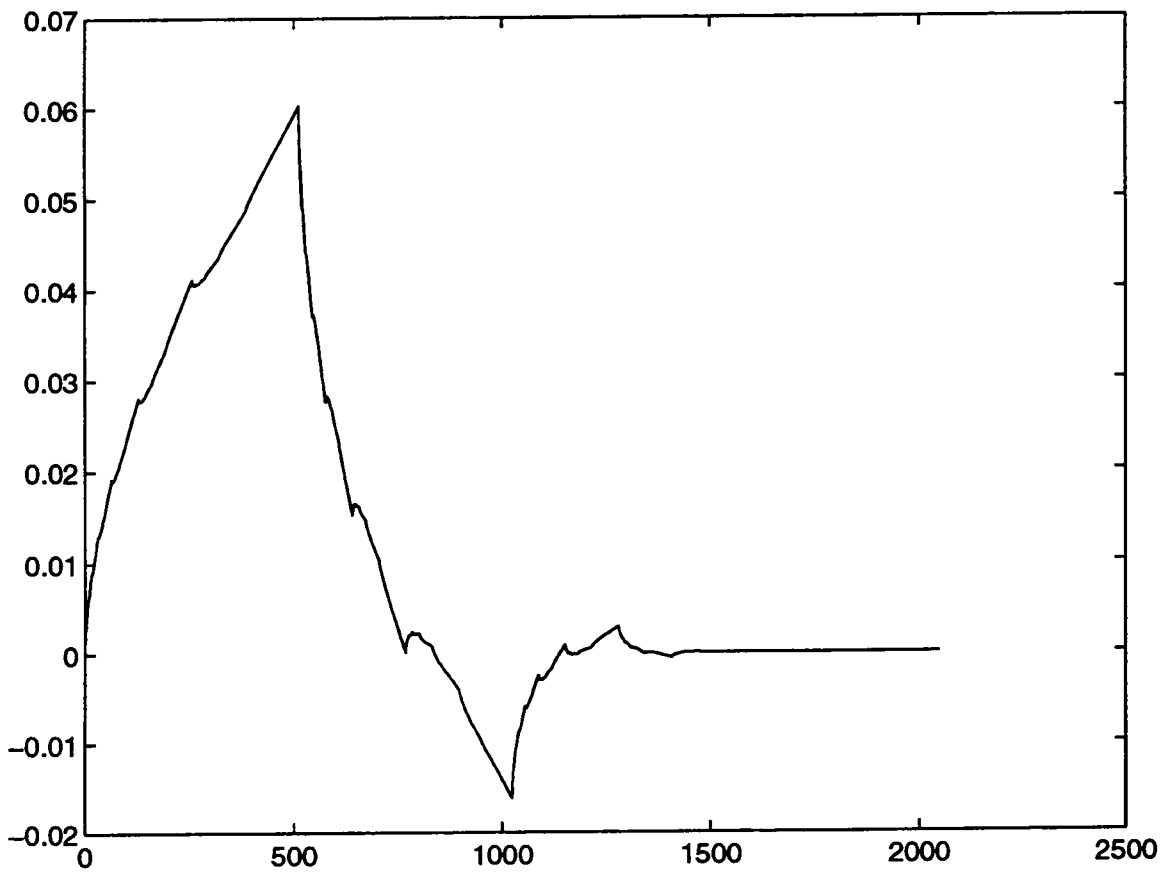

Figure 2.10: Daubechies mother wavelet with 4 vanishing points

2.7 Two Dimensional Wavelet Transformations

In order to work with the images in this research, it is necessary to perform two dimensional wavelet transformations. There are two means to decompose the pixel values of an image.

The first means of performing a two dimensional wavelet transformation results in a standard decomposition of an image. To perform this method, a one dimensional wavelet transform is applied to each row of pixel values. Then a one dimensional transform is applied to each column of the resulting matrix. These steps produce a matrix which contain all detail coefficients except for a single overall average coefficient.

The second means of performing a two dimensional wavelet transformation results in a non-standard decomposition. This process is done by first performing one step of horizontal pairwise averaging and differencing on the pixel values in each row of the image. This step is followed by applying pairwise averaging and differencing to each column of the resulting matrix. These steps are then repeated recursively on the area with averages in both directions.

Chapter 3

Artificial Neural Networks

This chapter discusses the relevant concepts of artificial neural networks that will be used in the work presented in later chapters of this thesis.

3.1 Introduction

Artifical neural networks (ANN) provide a computing architecture that was inspired by the structure of the brain. ANN's are massively parallel systems that derive their computing power from dense interconnections among simple computing elements. They got their name from the networks of nerve cells in the brain. At the current time ANN's can be at best considered an oversimplified model of the brain. Figure 3.1 gives a graphical comparison of speeds and connections among various organisms and current ANNs. However, despite poor comparisons with brains of living organisms, ANN are often used to address problems that

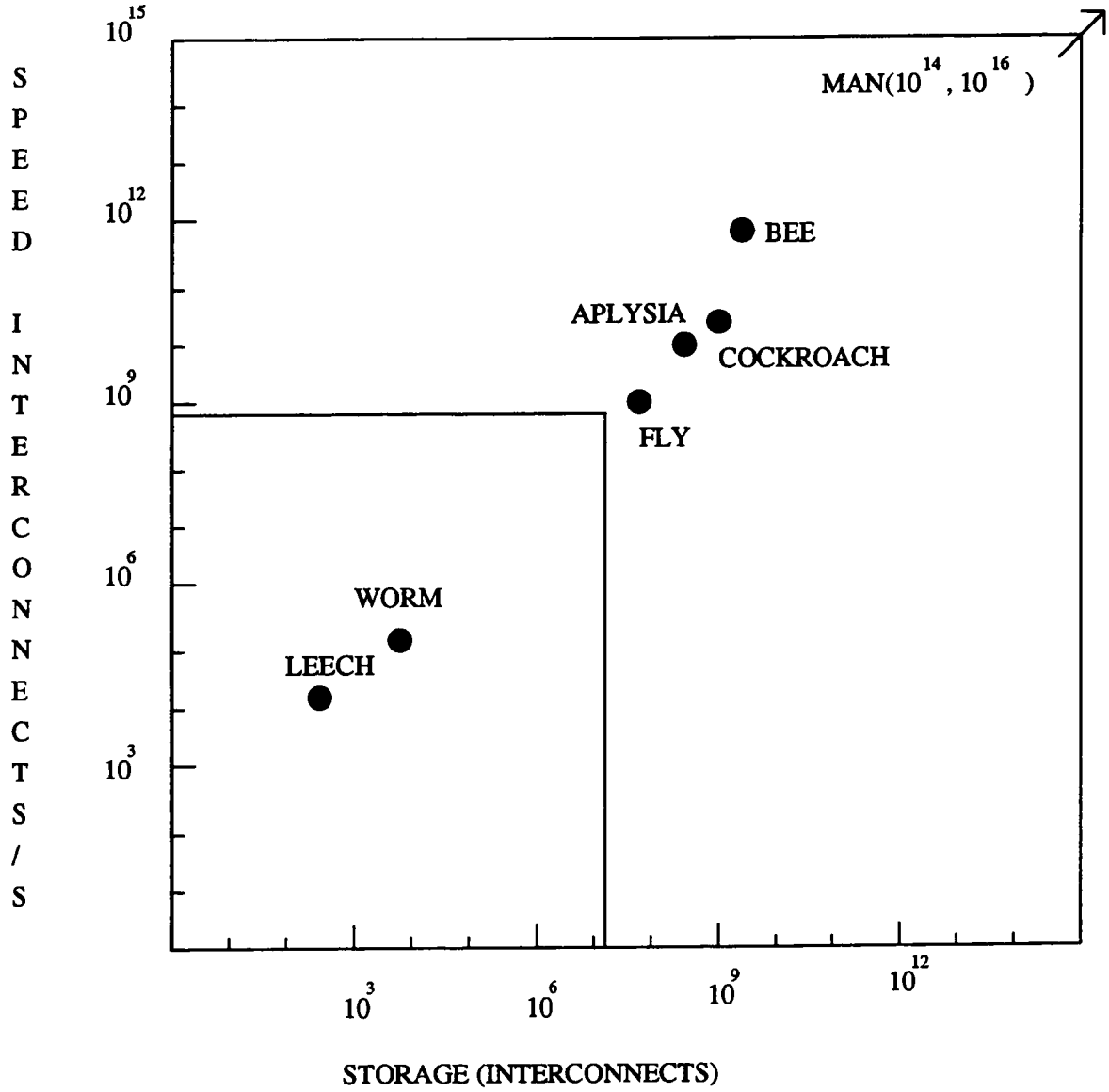

Figure 3.1: Speed vs storage for a number of organisms. Speed is measured in terms of interconnects/sec and storage is measured in terms of number of interconnects. The boxed area shows the power of current ANN simulators. (Figure obtained from [2]).

are intractable or cumbersome by traditional methods. ANNs are used to solve a large number of applications but they excel at solving problems involving patterns such as pattern mapping, pattern completion, and pattern recognition.

3.2 History

McCulloch and Pitts were generally credited with beginning the modern era of neural networks through their work which was published in a paper in 1943. Their paper described a logical calculus of neural networks. Their work inspired a flurry of research that initiated widespread interest in neural networks during the 1940's and 1950's.

In 1958, Rosenblatt introduced his work on the perceptron and the "perceptron convergence theorem." The discovery of the perceptron was soon followed by a similar discovery called the Adaline which was formulated by Widrow and Hoff in 1960. The Adaline was in turn followed by the Madaline structure which was proposed by Widrow and his students in 1962. The Madaline was one of the earliest trainable layered neural networks with adaptive elements.

During the 1960's the perceptron was mistakenly perceived as being able to accomplish any task. However, in 1969 Minsky and Papert proved that there were fundamental limitations to what a single layer perceptron can compute. They proved that the single layer perceptron can only classify patterns such as

the AND and OR functions, shown in figures 3.2 and 3.3, that are separable by a single line called linearly separable. The single layer perceptron is unable to classify patterns such as the XOR, shown in figure 3.4, that cannot be classified by a single line called linearly unseparable. The single layer perceptron was unable to classify patterns that are linearly unseparable because it only has one layer of adaptive weights.

input		output
0	0	0
0	1	0
1	0	0
1	1	1

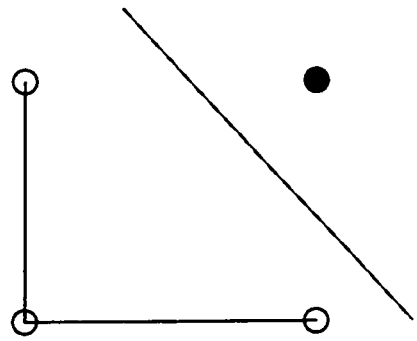

Figure 3.2: Linearly separable AND function (dark circle is a 1, light circle is a 0)

input		output
0	0	0
0	1	1
1	0	1
1	1	1

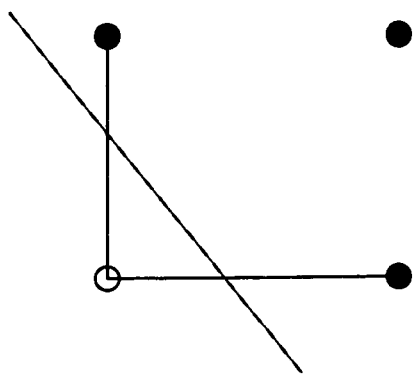

Figure 3.3: Linearly separable OR function (dark circle is a 1, light circle is a 0)

Minsky and Papert's discoveries about the limitations of the single layer perceptron were in large part responsible for the waning interest in neural networks

input		output
0	0	0
0	1	1
1	0	1
1	1	0

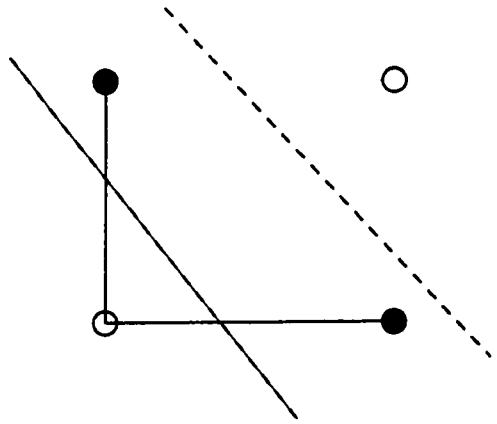

Figure 3.4: Linearly unseparable XOR function (dark circle is a 1, light circle is a 0)]

in the 1970's. Researchers in the 1970's were unable to develop a multiple layer perceptron that would be needed to classify linearly unseparable patterns. The difficulty in developing the multiple layer perceptron was due to the inability of researchers to solve the "credit assignment problem," which is the problem of assigning credit to neurons in a hidden layer of a multiple layer perceptron.

A number of major discoveries in the 1980's revived interests in neural networks. Perhaps the most notable of the discoveries in the 1980's was the development of the back propagation algorithm. Although Werbos developed the algorithm in 1974, back propagation was not widely known until Rumelhart and McClelland reported in 1986 its independent reinvention in 1982.

Back propagation was a vast improvement over the single layer perceptron. Since the single layer perceptron was limited to a single layer of adaptable weights, it was only able to classify patterns that were linearly separable. Because it uses a more sophisticated learning algorithm and can adapt two or more layers of

weights, back propagation was able to classify patterns that were not linearly separable. Each hidden layer unit acts as a feature detector that responds to specific features in the input pattern. The learning process organizes the feature detectors to accomplish the learning tasks presented to the network.

3.3 Back Propagation

Since back propagation (back prop) is one of the simplest learning algorithm for an artificial neural network (ANN), it is a good starting point for understanding ANN. A topology for a typical ANN using back prop is shown in figure 3.5. Although figure 3.5 shows a three layer, fully connected network, a back propagation network does not have to be fully connected. However, a back propagation network does have to have a minimum of three layers.

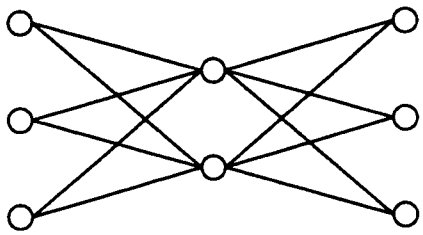

Figure 3.5: Typical back propagation ANN

The back propagation network uses a training method known as supervised learning. In supervised learning the network is presented with an input pattern

vector and an associated output vector. The network then processes the input pattern vector and obtains an output. The generated output is then compared to the target output vector to create error values, which in turn adjust the weights in the network.

The training of a back propagation network consists of two major processes. Forward propagation is started when an input pattern is presented to the network by means of the neurons, the fundamental processing units of an ANN. The input passes through the neurons in the input layer unmodified. The forward propagation step is actually performed by the remaining layers.

After the activation level a_i, the output of a neuron, is multiplied by weight w_{ji}, where w_{ji} is a from neuron j to neuron i, it enters the neuron of the next layer where it is summed with other inputs.

$$S_j = \sum_i a_i w_{ji} \tag{3.1}$$

S_j is then processed by a threshold function f to calculate $f(S_j)$. Although a number of differentiable functions can be used as a threshold function, the sigmoid function shown below is most often used.

$$f(S_j) = \frac{1}{1 + e^{-S_j}} = \frac{1}{1 + e^{-\sum a_i w_{ji}}} \tag{3.2}$$

$f(S_j)$ then become the activation level for the neuron j, which is sent to another

layer if the current layer is a hidden layer or becomes an output if the current layer is an output layer.

A bias unit can be added to a back propagation network to allow adjustment of the threshold function. A bias unit is a neuron that does not accept inputs, is fully connected to neurons in the next layer, and has a constant activation level of 1. In network that uses bias units, one bias unit is added for each layer of neurons except the input. Figure 3.6 is an example of a back propagation ANN that uses bias units.

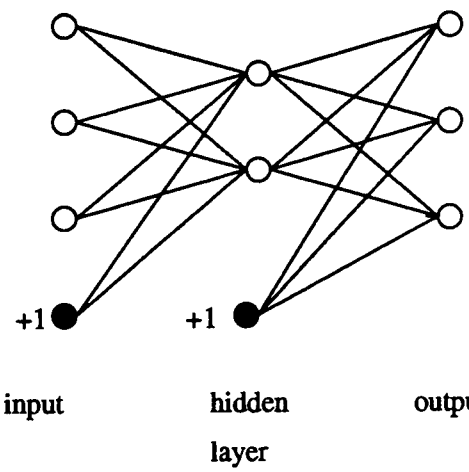

Figure 3.6: Typical back propagation ANN with bias units

The bias unit allows the threshold established by the threshold function to be adjusted. The adjustment is performed by translating the threshold function left or right. For example,

let

$$Z = \sum_{i=1}^{n} a_i w_{ji}$$

and

$$C = w_{bias}$$

then the value that is processed by the threshold function becomes:

$$S_j = Z + C$$

This C value provided by the bias unit effectively translates the threshold function from 0 to $-C$. Figure 3.7 shows how this translation is done for a sigmoid threshold function.

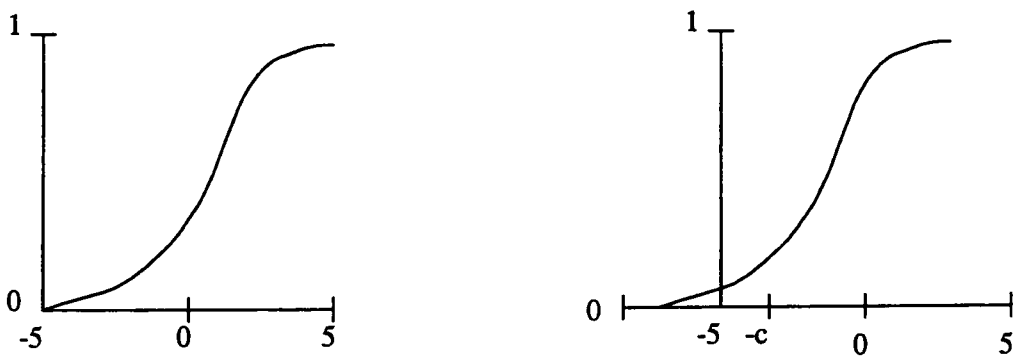

Figure 3.7: Effects of a bias unit on a sigmoid threshold function. (Figure from [2])

At the completion of the forward propagation process, the backward propagation

process is performed. After the activation levels are obtained from the neurons of the output layer, they are compared to a vector which contains the target output to get an error value. The error value for an output unit can be calculated by

$$\delta_j = (t_j - a_j) f'(S_j) \tag{3.3}$$

where t_j is the target value for unit j, a_j is the activation level and output value for unit j, $f'(x)$ is the derivative of the sigmoid threshold function, and S_j is the weighted sum of inputs to j. The derivative of the sigmoid function can be determined as shown.

$$a_j = f(S_j) = \frac{1}{1 + e^{-S_j}} \tag{3.4}$$

$$f'(S_j) = \frac{d}{dS_j} \frac{1}{1 + e^{-S_j}} \tag{3.5}$$

$$f'(S_j) = \frac{1}{(1 + e^{-S_j})^2} (-e^{-S_j}) \tag{3.6}$$

$$f'(S_j) = \frac{1}{(1 + e^{-S_j})} \frac{-e^{-S_j}}{(1 + e^{-S_j})} \tag{3.7}$$

$$f'(S_j) = a_j(1 - a_j) \tag{3.8}$$

The error value can now be written as :

$$\delta_j = (t_j - a_j)a_j(1 - a_j) \tag{3.9}$$

The error value for a neuron in a hidden layer is determined recursively, based on the neurons and the weights associated with those neurons and is given by

$$\delta_j = f'(S_j)\sum_{k=1}^{N_0} \delta_k w_{kj} \tag{3.10}$$

$$\delta_j = a_i(1 - a_i)\sum_{k=1}^{N_0} \delta_k w_{kj} \tag{3.11}$$

In order for learning to take place, the weights in the network must be adjusted. The weight adjustment is described by figure 3.8.

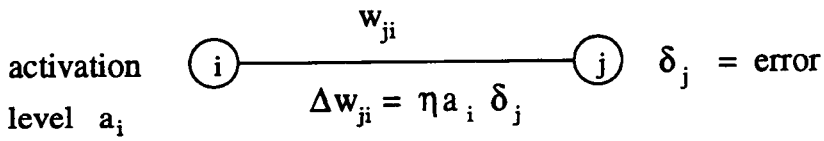

Figure 3.8: process of updating a weight update for back propagation. (Figure from [2])

The variable η is called the learning rate and is set by the user. The magnitude of η determines how fast the ANN learns. However the value of η must be chosen with care since an excessively large value would cause the network to become

unstable. Values of η that are too small would cause very slow learning.

The entire training process consists of many iterations. Each iteration consists of presenting one input/output vector pair to the network then performing one forward propagation step followed by one backward propagation step. The weights of the network are not adjusted until all of the patterns in the training set and their associated outputs have been presented, which is known as an epoch.

It is important to determine a stopping criterion for a training session. The root-mean-squared (RMS) error, which can be calculated by :

$$\sqrt{\frac{\sum_p \sum_j (t_{jp} - x_{jp}^2)}{n_p n_o}} \tag{3.12}$$

$$n_p = \text{number of } \textit{patterns}$$

$$n_o = \text{number of } \textit{output units}$$

is often used as a stopping criterion for a back propagation network. As the network learns the RMS error decreases. When the RMS error decreases below a set limit then training can be stopped.

The training process of a back propagation network is essentially the same as the gradient descent method. This process is analogous to a person starting at a random point on a mountainous terrain and making his way down the mountain.

He selects the direction of steepest descent and walks a pre-determined distance in that direction. This process is continued until he reaches a minimum in the terrain. However, this minimum is not necessarily a global minimum. It is possible to encounter a local minimum similar to the one shown in figure 3.9.

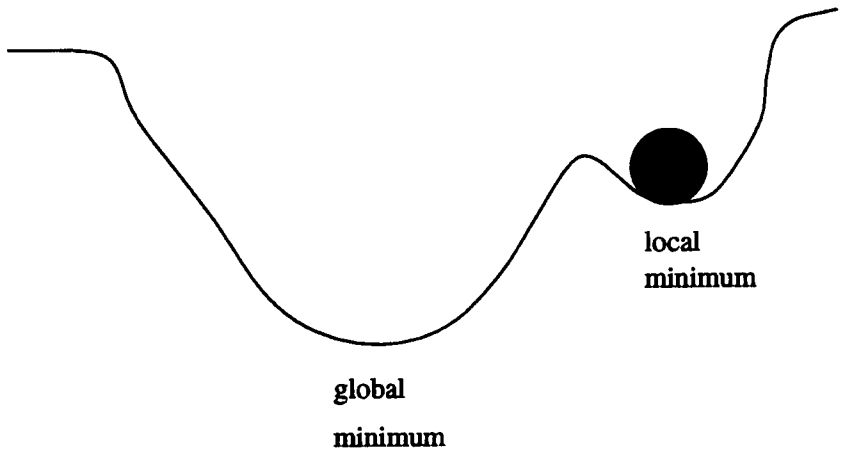

Figure 3.9: Local minimum on error surface

There are a number of methods that help to avoid a local minimum. One way is to change the terrain by changing the learning parameters or number of hidden units. Another way is to add small random values to the weights in the network. This process has the effect of moving a small distance away from the valley of the local minimum. If this distance is sufficiently large then the network would be able to escape the local minimum and move away from it.

3.4 Quick Prop

One of the biggest problems with back propagation is the large amount of time required for the network to converge to a solution. For larger problems it often takes hundreds or thousands of epochs for a back propagation network to converge to a solution.

Back propagation works by computing the partial derivative of the overall error with respect to the weights ($\frac{\partial E}{\partial w}$). The derivatives are used to perform gradient descent in weight space. If infinitesimal steps are taken down the gradient vector and a training epoch is performed to recompute the gradient after each step then a local minimum will eventually be reached.

The reason for the slow convergence time for the back propagation network is that it takes small steps. In order to learn faster the largest step possible needs to be taken. However, if the step size is too large the network will not converge to a solution. A reasonable step size can be chosen if the slope of the error function and higher derivatives or curvature at the current point in weight space is known. Information about higher derivatives is not available to back propagation.

One way to speed up the convergence of back propagation without having information of higher derivatives is by using a momentum term. The idea is to add a fractional portion of the previous weight change to the current weight change. This method is essentially an attempt to dynamically adjust the learning

rate for each weight. Although the use of momentum term increases the learning rate to a degree, the gain is not significant.

In 1988 Scott Fahlman introduced the "quick prop" algorithm which is a faster alternative to the back propagation algorithm. Quick prop is similar to back propagation except it keeps a copy of the error derivative from the previous epoch ($\frac{\partial E}{\partial w(t-1)}$) along with the difference between the current and previous value of this weight. For each weight, a parabola with arms opened upwards is constructed using previous and current error slopes and the weight change between the points at which these slopes were measured. The weight update is then calculated by determining the minimum point of this parabola. The calculation is shown below:

$$\Delta w(t) = \frac{S(t)}{S(t-1) - S(t)} \Delta w(t-1) \tag{3.13}$$

$$S(t) = \frac{\partial E}{\partial w(t)} \tag{3.14}$$

$$S(t-1) = \frac{\partial E}{\partial w(t-1)} \tag{3.15}$$

One problem that quick prop encounters, occurs when the current slope is in the same direction as the previous slope and is the same size or larger in magnitude. For this case, the weight update results in either an infinite step size

or causes the network to move backwards and away from the desired minimum. This problem was solved by creating what Fahlman [4] called a "maximum growth factor" μ, which limits the weight step to μ times the previous step for that weight. Fahlman reports that a value of 1.75 is optimal for most problems. Although a reasonable value for μ helps the network to converge, an overly large value of μ will prevent it from converging.

Another problem that quick prop faces is that the weights grow very large for certain problems, which leads to a floating point overflow. This problem was solved by adding a weight decay term to the slope allowing the weights to remain within a reasonable range.

Since the weight update in quick prop is dependent on the previous weight update, a means to start the process is needed. Furthermore, it is necessary to re-start the process if the weight previously had a step of zero but currently has a nonzero step size. Fahlman [4] that the training process could be started and restarted by adding the product of the learning rated and the current slope to the learning update.

Fahlman indicated that quick prop is faster than back propagation. His experiments involving the 10-5-10 encoder problem, which is a problem with 10 input neurons, 5 neurons in the hidden layers, and 10 output neurons, show that quick prop is nearly ten times faster than standard back propagation. These results are summarized in tables 3.1 and 3.2.

Table 3.1: Performance of back prop on the 10-5-10 encoder problem. (η = learning rate, r = range of random initial weights) (Table adapted from [4])

Problem	Trials	η	r	Max	Min	Average	S.D.
10-5-10	25	1.7	1.0	265	80	129	46

Table 3.2: Performance of quick prop on the 10-5-10 encoder problem. (η = learning rate, μ = maximum growth factor, r = range of random initial weights) (Table from [4])

Problem	Trials	η	μ	r	Max	Min	Average	S.D.
10-5-10	100	0.35	1.75	2.0	21	9	14.01	2.1

3.5 Cascade Correlation

The cascade correlation algorithm is a learning algorithm that was developed by Scott Fahlman to address the slow convergence problem of back propagation [5]. This algorithm works by adding one new unit to the hidden layer at a time, freezing its input weights and attempting to maximize the magnitude of the correlation between the new unit output and the remaining error. The initial cascade correlation network, shown in figure 3.10, consists of only the input units, a bias unit and output units. The network is then trained using quick prop until no significant error reduction occurs after a number of training iterations controlled by a "patience" parameter set by the user. If the error does not meet established criteria the cascade correlation algorithm adds a new unit to the hidden layer of

the active network in an attempt to further reduce the error.

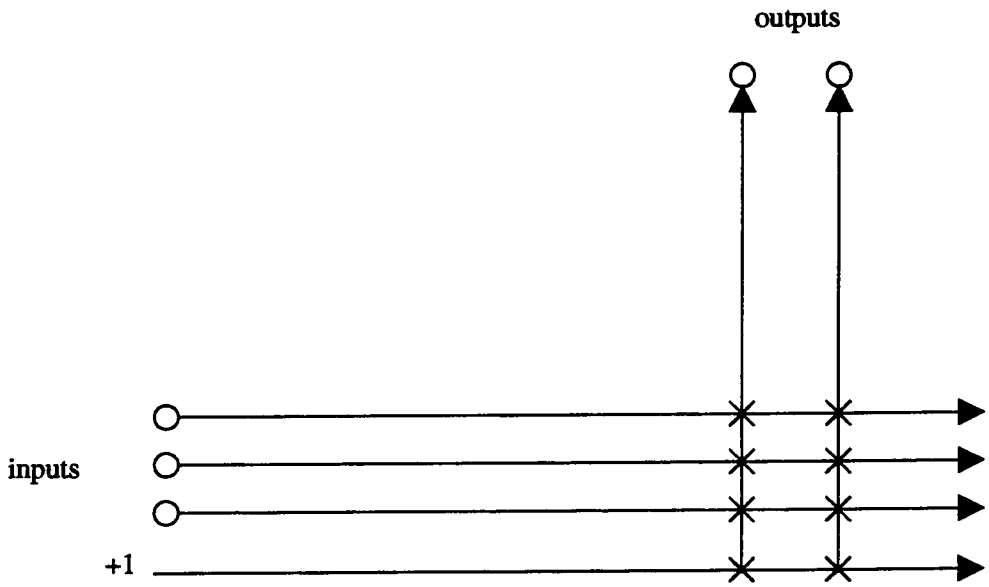

Figure 3.10: The initial state of an ANN using cascade correlation. The X's indicate weights that have not been fixed. Figure from ([5])

The process of adding a unit to the hidden layer in the active network begins by creating a pool of "candidate units" with outputs unattached to the active network. The external inputs of the active network along with the outputs from the units in the hidden layer in the active network are input to each candidate unit in the pool. A number of passes are made through the training set, adjusting the candidate's input weight after each pass, in order to maximize S which is defined by:

$$S = \sum_{o} \left| \sum_{p} (V_p - \overline{V})(E_{P_o} - \overline{E_o}) \right| \tag{3.16}$$

where o is the network output, p is the training pattern, V_p is the candidate unit's value for pattern p, $\overline{V}$ is the average value for a candidate unit for all patterns, E_{p_o} is the error observed at the output due to pattern p, and $\overline{E_o}$ is the average error for all patterns. S can be maximized by first computing $\frac{\partial S}{\partial w_i}$, the partial derivative of S with respect to a candidate unit's weight, to get

$$\frac{\partial S}{\partial w_i} = \sum_{p_o} \sigma_0 (E_{p_o} - \overline{E_o}) f'_p I_{ip} \tag{3.17}$$

where σ_0 is the sign of correlation between the value of the candidate unit and the output, f'_p is the derivative of the candidate unit's activation with respect to the sum of its input for pattern p, and I_{ip} is the input to the candidate unit from input i for pattern p.

Gradient ascent using quick prop is then performed for each unit in the candidate unit pool to maximize S. The candidate unit with the highest S is placed in the hidden layer of the active network with its input weights fixed. Figure 3.11 shows the network after a candidate unit has been installed in the hidden layer of an initial cascade correlation network.

The cascade correlation algorithm continues until enough units have been added to the hidden layer to suitably reduce the error or until addition of hidden layer units show negligible reduction of errors. Figure 3.12 shows an example of a completed cascade correlation network with 2 hidden layer units.

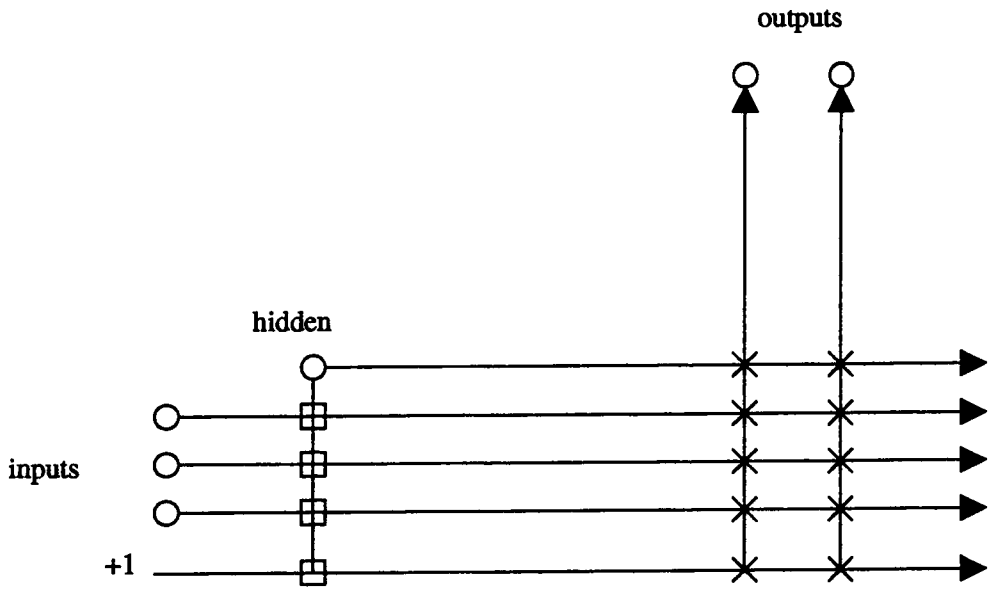

Figure 3.11: ANN using cascade correlation after addition of one hidden unit. The X's indicate weights that have not been fixed and the boxes are fixed weights. (Figure from [5])

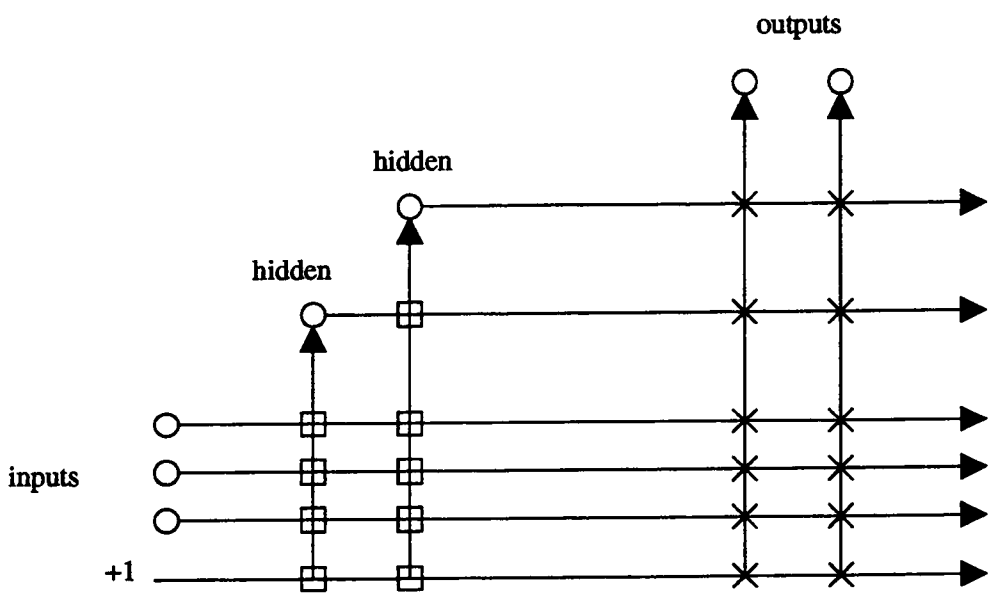

Figure 3.12: ANN using cascade correlation after addition of two hidden units. The X's indicate weights that have not been fixed and the boxes are fixed weights. (Figure from [5])

The cascade correlation algorithm solves a number of problems associated with back propagation. The primary problem that cascade correlation addresses is the slow speed at which back propagation converges to a solution. In fact Fahlman [5] reports that for the two spiral problem [5], cascade correlation is five times faster that quick prop and ten times faster that standard back propagation. Another problem that cascade correlation solves is the elimination of the need of the user to guess the size of the hidden layer in the network.

Chapter 4

Methods and Results

This chapter discusses the methods used in the identification of spiders and presents various results from experiments performed throughout the research.

4.1 Data

All living things can be placed into a refinable hierarchy of classification which is shown in table 4.1. This research is concerned with the identification of spiders at the genus and species levels to their position in the classification scheme. Due to the large number of genera and species of spiders in the order Araneae, focus will be placed on a limited number of samples in the family Lycosidae, shown in table 4.2, which were obtained through a loan from Dr. Charles D. Dondale from the Biosystematics Research Institute.

Spiders can be identified at the genus and species levels by the shape of their

Table 4.1: Hierarchy of Classification for living organisms

Classification
Kingdom
Phylum
Class
Order
Family
Genus
Species

Table 4.2: Classification of obtained samples

Genus	Species
Alopecosa	aculeata
Alopecosa	kochi
Pardosa	groenlandica
Pardosa	dromaea
Arctosa	rubicundi
Arctosa	emertoni

genitalia. Although identification can be made using the genitalia of either gender, the two-dimensional nature of the female epigynum makes it easier to use in a computer vision system than the three-dimensional structure of the male palp. Figures 4.1-4.6 give examples of the epigyna images used in this research.

The pattern classification system is based on an Artificial Neural Network (ANN), which can be trained to identify spiders based on the images of their epigyna. However, before any epigynum can be classified by ANN's, a number of preparatory tasks are needed, which include digitizing the epigyna of the spiders in the collection, processing of the resulting images, encoding of the resulting images, and processing of the images.

The epigyna were digitized by a CCD (Charged Coupled Device) camera which sends a magnified picture from a dissecting microscope to a video frame grabber and forms an image in TIFF (Tag Image Format). The images then undergo a number of processing steps in order to prepare them for encoding.

4.2 Data Processing and Encoding

Simulation of a large ANN is very computing intensive, so encoding of the images is necessary to reduce the input size and therefore the size of the ANN needed to process them. The encoding method that was selected for this research is the Daubechies 4 wavelet transformation. The wavelet transformation decomposes

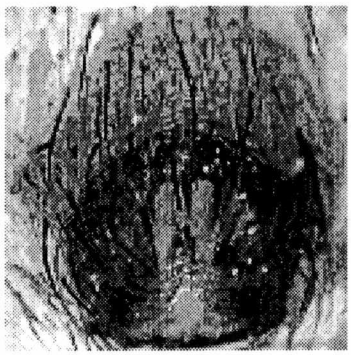

Figure 4.1: Image of an epigynum of a sample belonging to the genus Alopecosa, species aculeata

Figure 4.2: Imange of an epigynum of a sample belonging to the genus Alopecosa, species kochi

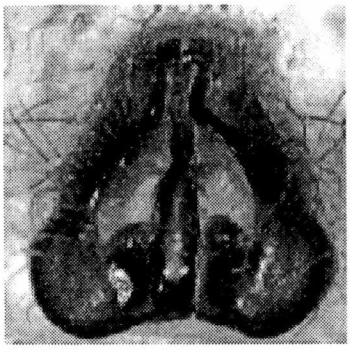

Figure 4.3: Image of an epigynum of a sample belonging to the genus Pardosa, species dromaea

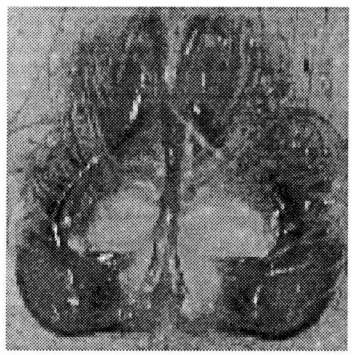

Figure 4.4: Image of an epigynum of a sample belonging to the genus Pardosa, species groenlandica

Figure 4.5: Image of an epigynum of a sample belonging to the genus Arctosa, species rubicundi

Figure 4.6: Image of an epigynum of a sample belonging to the genus Arctosa, species emertoni

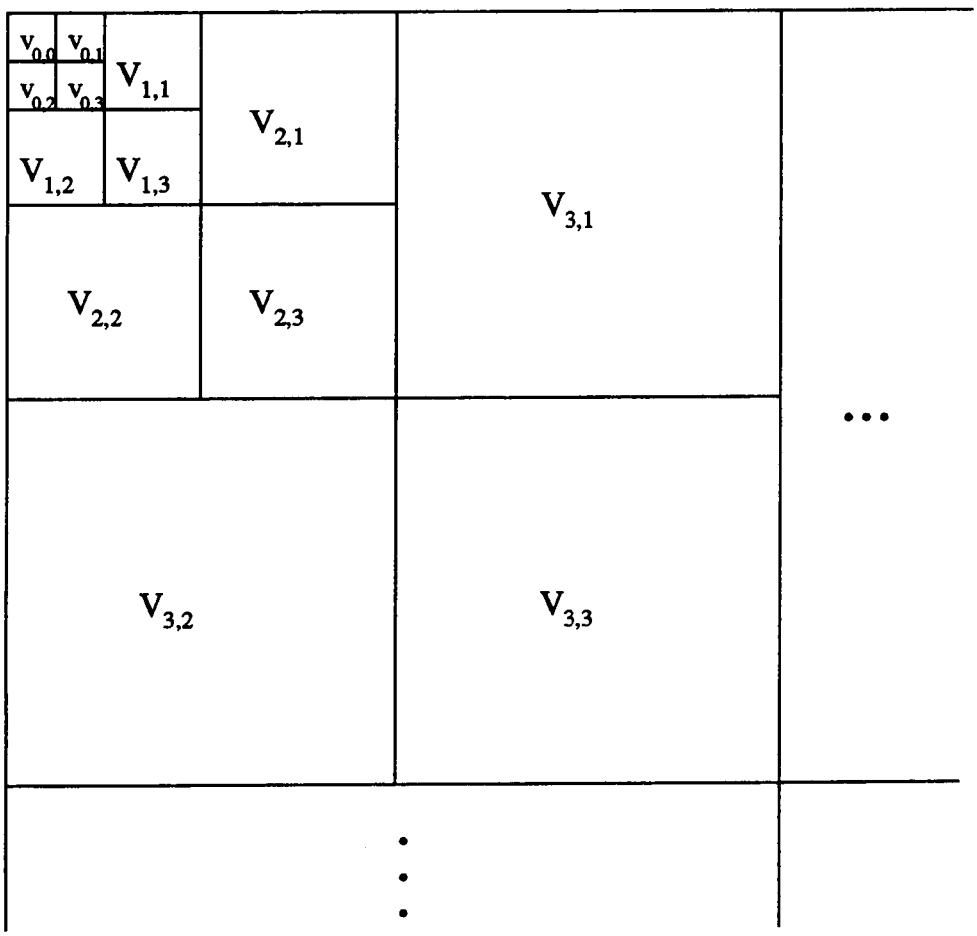

Figure 4.7: Location of vector spaces

the image into data of varying resolution. Figure 4.7 gives a wavelet coefficient layout diagram for an image that has undergone wavelet transformation.

From chapter 2, which shows that $V^0 \subset V^1 \subset V^2 \cdots$, the higher resolution information is represented by the wavelet coefficients at the higher level vector spaces. Reduction of the inputs to the ANN can be achieved by eliminating the wavelet coefficients in the higher level vector spaces. Elimination of the higher resolution data still leaves a substantial amount of information that can be used

for identification. For example, the blurred image shown in figure 4.9, is the reconstruction of the original image in figure 4.8, after high resolution information from vector spaces five and above have been eliminated.

The encoding process is preceded by a number of image processing steps. The first step involves using Xview, an image processing and viewing program written by John Bradley, to crop the image and remove background information, unrelated to the epigynum, that is not needed for identification. Since the two dimensional wavelet transformation, described in chapter 2, requires that the input image be a square image with a dimension of $2^j \times 2^j$, Xview is again used to scale the images down to a dimension of 128×128 ($2^7 \times 2^7$). This step not only formats the images for encoding by wavelet transformation, but it reduces the size of the images and therefore the amount of processing required by the ANN.

The wavelet transformation produces a matrix of large numbers capable of saturating the sigmoid threshold function in the ANN, which could lead to faulty results. One solution to this problem is to normalize the values in the matrix. Since the largest number in the matrix corresponds to the most averaged information in vector space zero, this number, located at the upper left corner of the matrix, can be used to normalize the matrix and create inputs suitable for processing by the ANN.

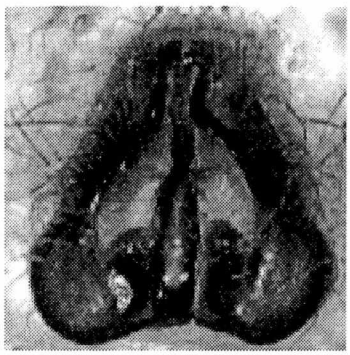

Figure 4.8: Original image of an epigynum of a sample belonging to the genus Pardosa, species dromaea

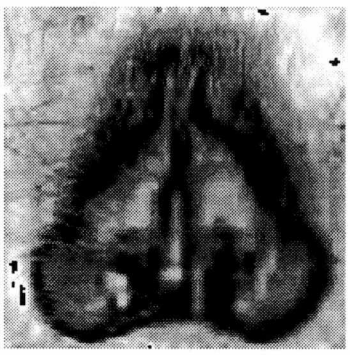

Figure 4.9: Image of an epigynum of a sample belonging to the genus Pardosa, species dromaea, after all information for vector spaces greater than 4 has been removed

4.3 Identification

The ANN used to perform the identification has six output neurons corresponding to each of the six species available. The ANN was trained on a 53 sample training set whose composition is shown in table 4.3. The images in the training set were encoded by Daubechies 4 wavelet transformation. Two experiments involving varying numbers of vector spaces were performed to determine the amount of information that will produce the most efficient system and provide the most accurate results. Table 4.4 gives the network training parameters that were used to train the ANNs for each of the experiments. The first experiment involved training the ANN using only wavelet coefficients in vector spaces V_0, V_1, V_2, and V_3. Since four vector spaces are used, the inputs became matrices with a dimension of $2^4 \times 2^4$, which then requires that the ANN have 256 neurons in the input layer. Experiments have shown that this is the largest ANN that can be trained in a reasonable amount of time on a Sparc 5 or a 486, which were the computers available. The second experiment involved training the ANN on the same training data, using wavelet coefficients only in vector spaces V_0, V_1, and V_2. The inputs became matrices with a dimension of $2^3 \times 2^3$, which required an ANN with 64 input neurons.

The trained ANNs were then tested over a 51 sample test set described in table 4.5. Table 4.6 and 4.7 shows that the best results were obtained from the larger

Table 4.3: Composition of the training set for the ANNs classifying all samples

Genus / Species	number of samples in training set
Alopecosa / aculeata	9
Alopecosa / kochi	8
Pardosa / groenlandica	9
Pardosa / dromaea	10
Arctosa / rubicundi	9
Arctosa / emertoni	8

Table 4.4: Network parameters for ANN classifying all samples

Parameter	space $V_0 - V_3$	space $V_0 - V_2$
N_{input} (number of input neurons)	256	64
N_{output} (number of output neurons)	6	6
η (learning rate)	0.05	0.05
momentum	0.95	0.95
number of candidate patience	12	12
minimum covariance change	0.04	0.04
maximum number of covariance update	200	200
number of epochs	300	300
stopping error	0.001	0.001

ANN, which processed inputs from vector spaces V_0, V_1, V_2, and V_3, indicating that information in V_3 is needed to match up minute features in the samples from the training set with those in the samples from the testing set. Although the ANN was capable of correctly identifying 37 out of 52 samples for inputs from vector spaces V_0, V_1, V_2, and V_3, it showed poor results in the identification of *Pardosa groenlandica* and *Arctosa rubicundi.*

One of the perceived reasons for the poor results in identifying the *P. groenlandica* samples is that their epigyna are similar to that of *P. dromaea* . For example, figures 4.4 and 4.3 show the similarities of the epigyna of the species groenlandica with the epigyna of the species dromaea, both of which belong to the genus Pardosa. Likewise, figures 4.5 and 4.6 show the similarities between the species *A. rubicundi* and *A. emertoni*, both of which belong to the genus Arctosa. The ANN's poor performance in identifying these species seems to suggest that a single ANN is not capable of simultaneously learning the differences among the different epigyna of the species in the data set.

One solution is to train ANN's that specialize in the classification of species within a given genus. Allowing the ANN to focus on the minute differences among the species of a given genus, instead of requiring it to learn to classify a number of species from several genera, may give it a better chance of classifying species with close resemblance. In order to determine the appropriate ANN to classify the species of a sample, an additional ANN can be trained to identify the genus

Table 4.5: Composition ofthe testing set for the ANNs classifying all samples

Genus / Species	number of samples in testing set
Alopecosa / aculeata	10
Alopecosa / kochi	7
Pardosa / groenlandica	9
Pardosa / dromaea	9
Arctosa / rubicundi	9
Arctosa / emertoni	8

Table 4.6: Performance of an ANN classifying all samples using vector spaces V_0, V_1, V_2, and V_3 over the testing set in table 4.5

Genus / Species	correct identifications	total #	% correct
Alopecosa / aculeata	8	10	80
Alopecosa / kochi	5	7	71
Pardosa / groenlandica	4	9	44
Pardosa / dromaea	8	9	89
Arctosa / rubicundi	5	9	56
Arctosa / emertoni	7	8	86
total	37	52	73

Table 4.7: Performance of an ANN classifying all samples using vector spaces V_0, V_1, and V_2 over the testing set in table 4.5

Genus / Species	correct identifications	total #	% correct
Alopecosa / aculeata	7	10	70
Alopecosa / kochi	4	7	57
Pardosa / groenlandica	4	9	44
Pardosa / dromaea	8	9	89
Arctosa / rubicundi	6	9	67
Arctosa / emertoni	7	8	86
total	36	52	69

of that sample. Once the genus of the unknown sample is determined, a proper ANN can be selected to classify the species of that unknown sample.

In order to test the hierarchical method of identification, one ANN was trained to identify the genus of a sample in the family Lycosidae (the Lycosidae ANN) and three other ANNs (Alopecosa ANN, Pardosa ANN, and Arctosa ANN) were trained to identify the species of the three genera in the research. Three sets of these ANNs were created in order to process Daubechies 4 wavelet transformed images for three vector space sizes $V_0 - V_1$, $V_0 - V_2$, and $V_0 - V_3$. Tables 4.8 and 4.9 give the composition of the training sets for the ANNs, and tables 4.10 and 4.11 gives the network parameters used to train the ANNs. The ANNs were then tested over the testing sets shown in tables 4.12 and 4.13. The test results shown in tables 4.14, 4.15, 4.16 and 4.17 show that the hierarchical method of identification gives, for the most part, more accurate results. The only exception is the accuracy in the identification of the species emertoni of the genus Arctosa. In both cases, an accuracy of 86% was obtained. The results also show that, in general, better results are obtained when wavelet coefficients in vector space V_3 are available.

Although classification of samples that belong to a known group can be useful, being able to recognize that a sample belongs to an unknown group (i.e. one not previously seen) can make the ANN an invaluable tool. In order to test this capability, a number of samples belonging to other genera in the family Lycosidae

Table 4.8: Composition of training set for the Lycosidae ANN (species making up the genus are in parentheses)

ANN	Genera	# in training set
Lycosidae ANN	Alopecosa (aculeata, kochi)	18
	Pardosa (groenlandica, dromaea)	18
	Arctosa (rubicundi, emertoni)	17

Table 4.9: Composition of training set for the Alopecosa ANN, the Pardosa ANN, and the Arctosa ANN

ANN	Species	# in training set
Alopecosa ANN	aculeata	11
	kochi	7
Pardosa ANN	groenlandica	8
	dromaea	10
Arctosa ANN	rubicundi	8
	emertoni	9

Table 4.10: Network parameters for the Lycosidae ANN

Parameter	space $V_0 - V_3$	space $V_0 - V_2$	space $V_0 - V_1$
N_{input} (# input neurons)	256	64	16
N_{output} (#output neurons)	3	3	3
η (learning rate)	0.05	0.05	0.05
momentum	0.95	0.95	0.95
candidate patience	12	12	12
min covariance change	0.04	0.04	0.04
max # covariance update	200	200	200
number of epochs	300	300	300
stopping error	0.001	0.001	0.001

Table 4.11: Network parameters for Alopecosa, Pardosa, and Arctosa ANN

Parameters	space $V_0 - V_3$	space $V_0 - V_2$	space $V_0 - V_1$
N_{input} (#input neurons)	256	64	16
N_{output} (#output neurons)	2	2	2
η (learning rate)	0.05	0.05	0.05
momentum	0.95	0.95	0.95
candidate patience	12	12	12
min covariance change	0.04	0.04	0.04
max # covariance update	200	200	200
number of epochs	300	300	300
stopping error	0.001	0.001	0.001

Table 4.12: Composition of the testing set for the Lycosidae ANN

ANN	Genera	# in testing set
Lycosidae ANN	Alopecosa (aculeata and kochi)	16
	Pardosa (groenlandica and dromaea)	19
	Arctosa (rubicundi and emertoni)	16

Table 4.13: Composition of the testing set for the Alopecosa ANN, the Pardosa ANN, the and Arctosa ANN

ANN	Species	# in training set
Alopecosa ANN	aculeata	8
	kochi	8
Pardosa ANN	groenlandica	10
	dromaea	9
Arctosa ANN	rubicundi	7
	emertoni	9

Table 4.14: Performance of the Lycosidae ANN using different vector spaces over the testing set in table 4.12

ANN	space	Genera	correct ID	total #	% correct
Lycosidae ANN	$V_0 - V_3$	Alopecosa	16	16	100
		Pardosa	19	19	100
		Arctosa	16	16	100
		total	51	51	100
	$V_0 - V_2$	Alopecosa	16	16	100
		Pardosa	19	19	100
		Arctosa	16	16	100
		total	51	51	100
	$V_0 - V_1$	Alopecosa	16	16	100
		Pardosa	14	19	74
		Arctosa	16	16	100
		total	46	51	90

Table 4.15: Performance of the Alopecosa ANN using different vector spaces over the testing set in table 4.13

ANN	space	species	correct ID	total #	% correct
Alopecosa ANN	$V_0 - V_3$	aculeata	8	8	100
		kochi	6	8	75
		total	14	16	88
	$V_0 - V_2$	aculeata	8	8	100
		kochi	8	8	100
		total	16	16	100
	$V_0 - V_1$	aculeata	6	8	75
		kochi	6	8	75
		total	12	16	75

Table 4.16: Performance of the Pardosa ANN using different vector spaces over the testing set in table 4.13

ANN	space	species	correct ID	total #	% correct
Pardosa ANN	$V_0 - V_3$	groenlandica	5	10	50
		dromaea	9	9	100
		total	14	19	74
	$V_0 - V_2$	groenlandica	5	10	50
		dromaea	9	9	100
		total	14	19	74
	$V_0 - V_1$	groenlandica	4	10	40
		dromaea	8	9	89
		total	12	19	63

Table 4.17: Performance of the Arctosa ANN using different vector spaces over the testing set in table 4.13

ANN	space	species	correct ID	total #	% correct
Arctosa ANN	$V_0 - V_3$	rubicundi	7	9	78
		emertoni	6	7	86
		total	13	16	81
	$V_0 - V_2$	rubicundi	5	9	56
		emertoni	6	7	86
		total	11	16	69
	$V_0 - V_1$	rubicundi	5	9	56
		emertoni	5	7	86
		total	10	16	63

were obtained to represent unknown samples. The limited availability of samples only permitted the testing of the capability of the Lycosidae ANN in identifying unknown samples. Ten images representing the unknown samples were added to the Lycosidae ANN training set and eleven images representing the unknown samples were added to the Lycosidae testing set. Table 4.18 shows the identification capabilities of the ANNs for the various vector space sizes when unknown samples are involved.

Although the overall results were good, the ANN misclassified six unknown samples as known samples. Despite the apparent poor results in identifying unknown samples, the 46% accuracy was good considering that the ANN was asked to classify samples belonging to a large group after "learning" from a training set with a small group of unknown samples.

Table 4.18: Performance of Lycosidae ANN using different vector spaces over the testing set in table 4.12 with unknowns

ANN	space	Genera	correct ID	total #	% correct
Lycosidae ANN	$V_0 - V_3$	Alopecosa	16	16	100
		Pardosa	19	19	100
		Arctosa	16	16	100
		unknown	5	11	46
		total	57	62	92
	$V_0 - V_2$	Alopecosa	16	16	100
		Pardosa	19	19	100
		Arctosa	16	16	100
		unknown	5	11	46
		total	57	62	92
	$V_0 - V_1$	Alopecosa	16	16	100
		Pardosa	14	19	100
		Arctosa	16	16	100
		unknown	4	11	36
		total	50	62	81

Chapter 5

Conclusion

The pattern classification system produced by this research gave relatively good identifications over the sample spiders in the Lycosidae family. The Lycosidae ANN (the ANN identifying the genus in the Lycosidae family) gave particularly good results. It correctly identified 51 samples in the genera Alopecosa, Pardosa, and Arctosa. Although the species ANNs (i.e. the Alopecosa ANN, Pardosa ANN, and Arctosa ANN) did not have the same level of success, they did produce good results. The species *groenlandica* and *dromaea* in the genus *Pardosa* shared unusual likeness, which made it difficult to differentiate between samples in each species.

This research is a good first step in the development of a practical and beneficial system. The pattern classification system was able to make relatively accurate identifications using the female epigynum which contains less information than the

male palp. The results indicate that this system can be developed into a practical system for scientists.

5.1 Further study

Although the pattern classification system produced by this research showed good results, there are a large number of improvements still possible. The performance of the species ANNs and particularly the Pardosa ANN, would improve vastly if the size of the training set were increased. Furthermore, the results would most likely improve by adding the vector space V_4, which may provide the extra details that are necessary to contrast the minute differences necessary to differentiate samples that are very similar. Improvements in technology have made computers with the power to simulate larger ANNs affordable.

Other areas of future studies that should be explored include the capability to identify three dimensional structures such as the palp in the male spiders, and to identify other organisms. The current system is only capable of looking at the two-dimensional epigynum in the female spiders. Also other ANNs and methods of encoding the input data should be compared with the current cascade correlation ANN and Daubechies 4 wavelet transformation, used in this research, to determine the combination that would create the best pattern classification system. Finally, adding the capability to input other types of data such as geographical location

where the sample was found should improve the accuracy of the system.

Bibliography

Bibliography

[1] Donoho Johnstone Scargle Buckheit, Chen. About wavelab. Technical report, Stanford University and Nasa Ames, 1995.

[2] Judith E. Dayhoff. *Neural Network Architectures : an Introduction.* Van Nostrand Reinhold, 1990.

[3] David H. Salesin Eric Stollnitz, Tony D. DeRose. Wavelets for computer graphics: A primer. Technical Report 94-09-11, University of Washington, Seattle, Wa, 1994.

[4] Scott E. Fahlman. An empirical study of learning speed in back propagation network. Technical Report CMU-CS-88-162, Carnegie Mellon University, Pittsburgh, Pa, 1988.

[5] Scott E. Fahlman and Christian Lebiere. The cascade-correlation learning architecture. Technical Report CMU-CS-90-100, Carnegie Mellon University, Pittsburgh, Pa, 1991.

[6] Amara Graps. An introduction to wavelets. *IEEE Computational Science and Engineering, Summer 1995*, 2(2), 1995.

[7] Diana Nelson Jones. Insect specialists, a disappearing species. *Pittsburgh Post-Gazette*, 1994.

[8] D. K. R. McCormack. *An investigation into the representation of data for the neural implementation of a handwritten static signature verification system.* PhD thesis, university of Wales, College of Cardiff, 1994.

[9] Stephanie Pain. Taxing times for 'old fashioned' biology. *New Scientist*, 1992.

Vita

Martin Thao Do was borned in Saigon Vietnam on January 3, 1968 and emigrated to the United States with his family in 1975. He graduated from Rhea County High School in 1985. In 1989 he earned a Bachelor of Science in Electrical Engineering from Tennessee Technological University. After graduation he worked for Raytheon's Missile System Division as a system engineer. He enrolled at the University of Tennessee in 1993. While at the University of Tennessee he spent time developing software for the Parallel Virtual Machine project and worked as system administrator and software developer for the Department of Computational Chemistry and Structural Biology Group at Oak Ridge National Laboratory.