2

Quarterly Progress Report Number 1

Transcutaneous Analyte Measuring Methods
(TAMM Phase II)

Dr. Kenneth J. Schlager

Biotronics Technologies, Inc.

September, October, November - 1991

Naval Medical Research and Development Command

Quarterly Progress Report Number 1

Transcutaneous Analyte Measuring Methods
(TAMM Phase II)

Prepared by
Principal Investigator
Dr. Kenneth J. Schlager

Biotronics Technologies, Inc.
W226 N555B Eastmound Drive
Waukesha, WI 53186

September, October, November - 1991
Contract Number N00014-91-C-0190

Prepared for
Naval Medical Research and Development Command
Bethesda, Maryland

TABLE OF CONTENTS

Abstract

The primary objectives of the first quarter of Phase II TAMM were the following:

1. The design of a near infrared (NIR)-800 photodiode array spectrometer, two of which would be used in clinical testing during 1992.

2. The development of advanced pattern recognition software for analyzing the data collected with the spectrometer.

3. The establishment of an ongoing, internal test program with the BI-102 infrared analyzer.

The status of the major activities outlined above is summarized below and detailed in the body of the report.

1. The primary task for the design of the NIR-800 during the first quarter of the project was the selection of a photodiode array. Two types of arrays were evaluated: a germanium (Ge) array and an indium-gallium-arsenide (InGaAs) array. The InGaAs array was tentatively selected.

   Two systems that will include the array selected will be ready for clinical testing at the Medical College of Wisconsin (Milwaukee County Medical Complex) and the Naval Medical Center (Bethesda) by March 1, 1992.

2. The advanced pattern recognition analytical software called NETGEN was developed for the system and is currently operational. This software shows a much improved performance over the previous pattern recognition software in analyzing information from the original 100-patient database.

3. An ongoing internal test program using the BI-102 infrared analyzer has been established. The preclinical testing will continue for the next three months. Both a cross-sectional analysis and serial tracking will be performed on a limited number of people.

Instrumentation

The central issue in the design of the TAMM instrumentation was the selection of the photodiode array. Two candidates were evaluated: a Ge photodiode array and an InGaAs photodiode array. Three criteria were used to evaluate the arrays:

1. cost
2. sensitivity
3. availability

Cost

Currently, the Ge array is less expensive. However, upcoming improvements to the InGaAs array promise a reduction in cost such that by the end of 1992 the cost of the InGaAs array will be equal to that of the Ge array.

Sensitivity

The InGaAs array is considerably more sensitive (by a factor of 10) than the Ge array. Also, the InGaAs array does not require cooling.

Availability

The availability of the InGaAs array is slightly better than that of the Ge array.

Based on an evaluation of the three criteria detailed above, the InGaAs array was selected. This array will be installed in the first two systems that will be built. The systems should be completed by January 15, 1991, allowing for 1 1/2 months of in-house testing before the clinical trials in March 1992.

Genetic Neural Network Analytical Software

The major effort during the first three months of the project was in developing the analytical software NETGEN. NETGEN is a set of analytical programs that combine the best features of neural networks and genetic algorithms. A complete description of the NETGEN system is included in Appendix I. The comments provided here will be of a general descriptive nature and will be related to test results resulting from the evaluation of the original 100-patient database.

Artificial neural networks (ANNs) are a form of distributed parallel processing of information that attempts to simulate the human brain. For application in TAMM, ANNs are an alternative to previous pattern recognition methods used for predicting blood analyte concentrations from NIR spectra. The advantages of using ANNs are as follows:

1. ANNs allow for the complex, nonlinear functions that relate spectra to blood chemistry.

2. ANNs perform beyond the best linear test results to date.

A disadvantage of ANNs is that they are local optimizers. They do not necessarily find the best final solution.

Genetic algorithms use an optimizing technique that when combined with ANNs can improve their performance in selecting the best solution. Genetic algorithms mimic natural genetics by providing for the survival of the fittest. These genetic algorithms use natural genetic methods of reproduction, cross breeding and mutation to produce a population of chromosomes from which the fittest are selected. Genetic algorithms are global optimizers that, when combined with ANNs, lead to the selection of the best solution. The pattern recognition software developed during the past three months at Biotronics, NETGEN, is a combination of ANNs and genetic algorithms. NETGEN solves for the best weights, selects the best learning set samples and selects the best wavelength variables for test set prediction.

A summary of NETGEN, nearest neighbor (original method) and straight backpropagation neural network test set variables and t-values are provided in Table 1. The neural network-genetic algorithm program (NG) in all cases performed better than either the nearest neighbor (NN) or straight backpropagation neural network (BP) programs in predicting results for calcium, potassium, sodium, BUN and glucose. NETGEN produced the smaller error and the higher t-value for tracking performance. Even better results are expected in the future as more features are added to the NETGEN software.

Experimental Preclinical Testing

The original BI-102 Infrared Analyzer is being used in experiments designed to provide preclinical experience with optrode configuration and software. It was decided to gain additional clinical experience by pursuing in-house testing before the new sys*em has been completely developed. The BI-102 is currently set up in Biotronics' chemical laboratory. Biotronics personnel are being tested for glucose using the BI-102. Boehringer Mannheim's Accu-Check glucose meter results are being used as a comparison for the outputs provided by the BI-102.

The NIR spectra collected with the BI-102 are being used for two types of analysis.

1. The results are being used for the serial tracking of patients. The goal of this type of analysis is to determine whether the glucose levels of individuals can be successfully tracked through normal fluctuations. In some cases, sugar diet injections will be given to cause fluctuations in glucose levels.

2. A cross-sectional analysis of the results will also be performed using a small database to determine whether the system can predict glucose accurately.

Glucose was selected over other analytes for the preclinical testing because it is easy to measure with currently available instrumentation. During experimentation, both transmissive (800 - 1100 nm) and reflective (1200 - 1800 nm) runs will be conducted. Several anatomical sites will be tested. The site originally selected was the arm. Locations over veins and arteries will also be evaluated. Testing will be conducted with the BI-102 until the new system is operational. Test results from the new system can then be compared with results collected earlier to establish the validity of the new system.

Clinical Test Procedures

Considerable thought is being given to the clinical test procedures and data handling techniques. The amount of data flow handling involved on the part of the operator, patient and data entry technician are illustrated in Figure 1. A discussion of the details of clinical test protocol will fcllow in later reports.

Kenneth J. Schlager
Principal Investigator

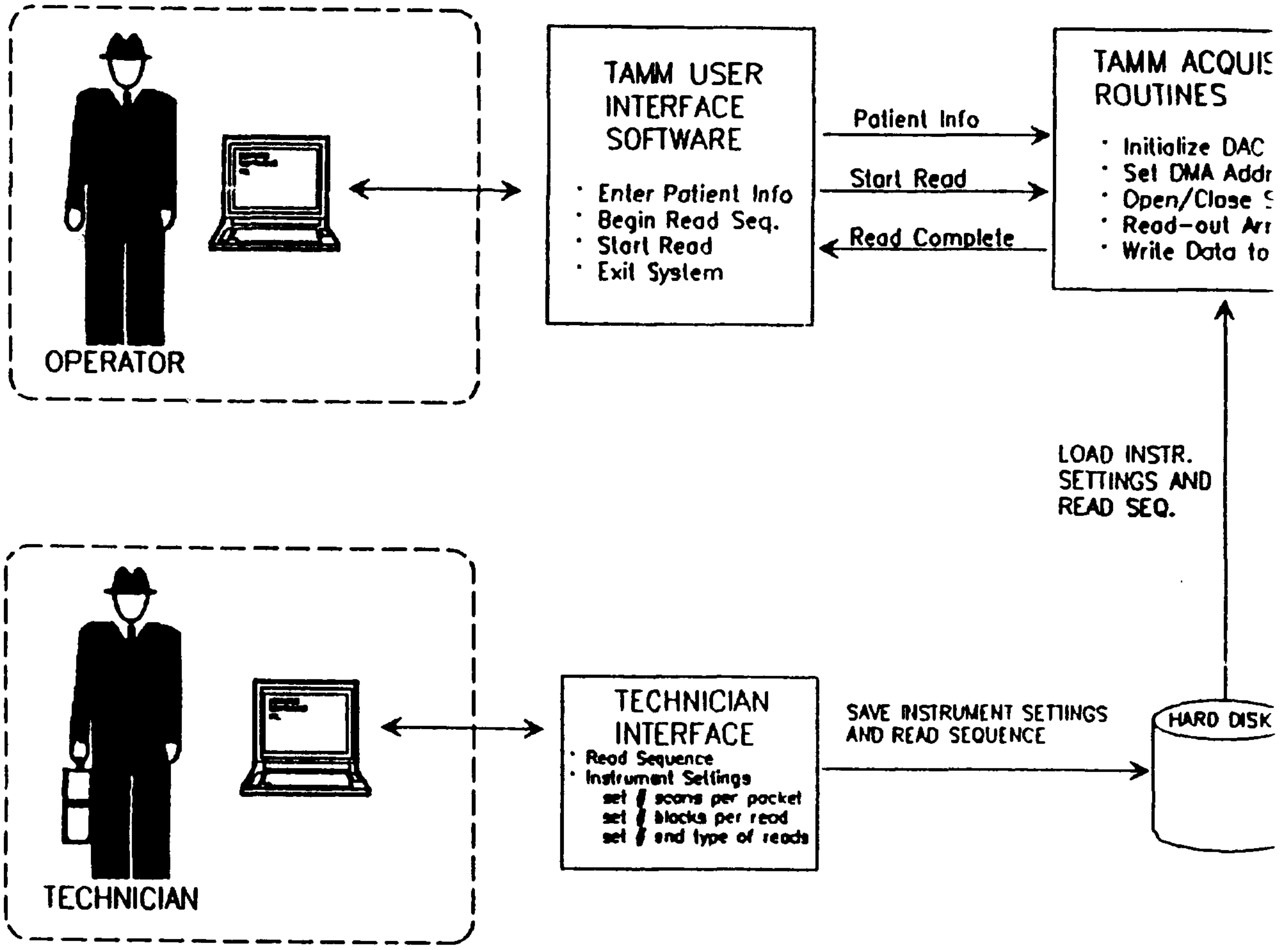
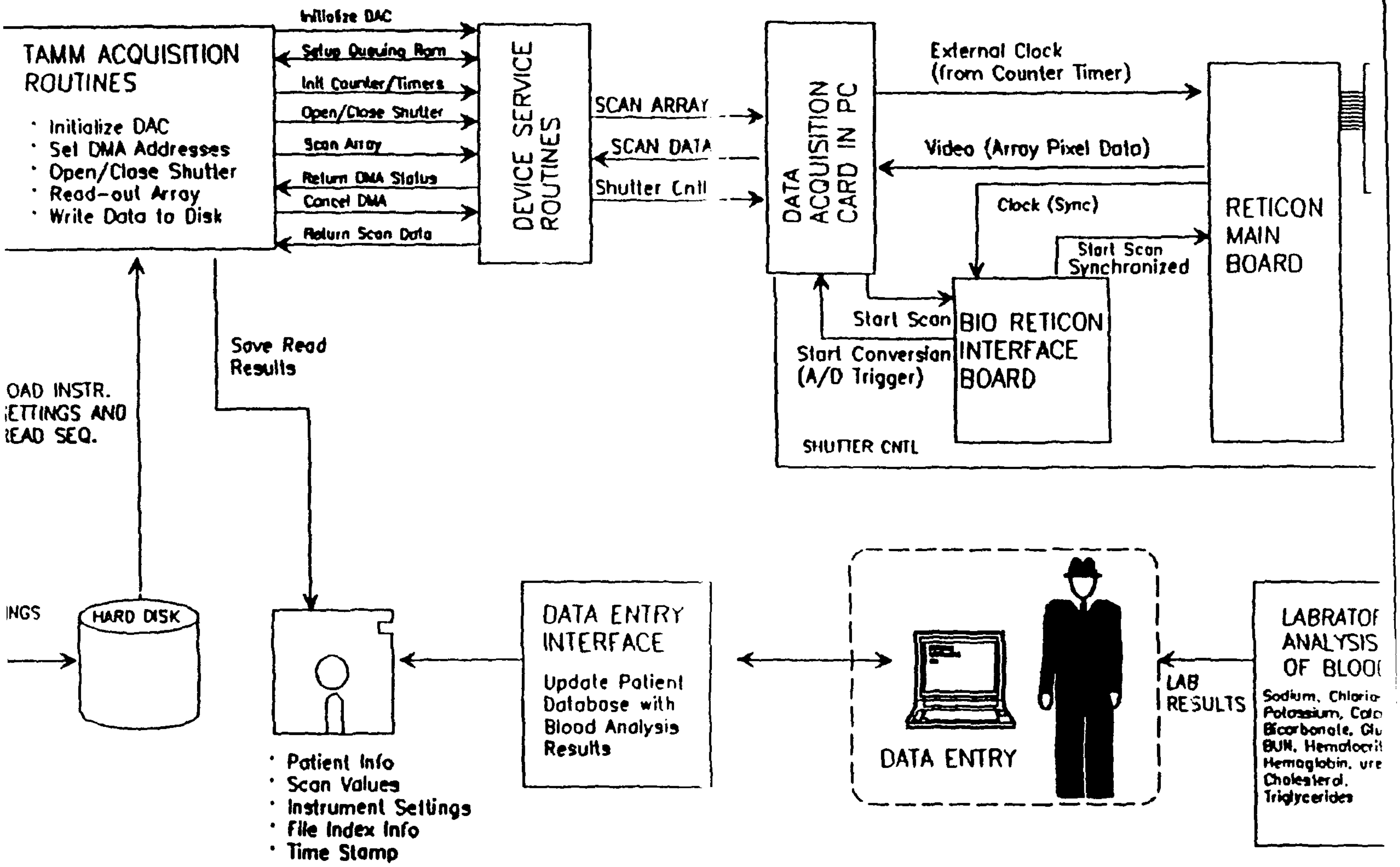

OPERATOR
TAMM USER INTERFACE SOFTWARE
· Enter Patient Info
· Begin Read Seq.
· Start Read
· Exit System
Patient Info
Start Read
Read Complete
TAMM ACQUIS
ROUTINES
· Initialize DAC
· Set DMA Addr
· Open/Close S
· Read-out Arr
· Write Data to
LOAD INSTR. SETTINGS AND READ SEQ.
TECHNICIAN
TECHNICIAN INTERFACE
· Read Sequence
· Instrument Settings
set # scans per packet
set # blocks per read
set # and type of reads
SAVE INSTRUMENT SETTINGS AND READ SEQUENCE
HARD DISK

FIGURE 1.

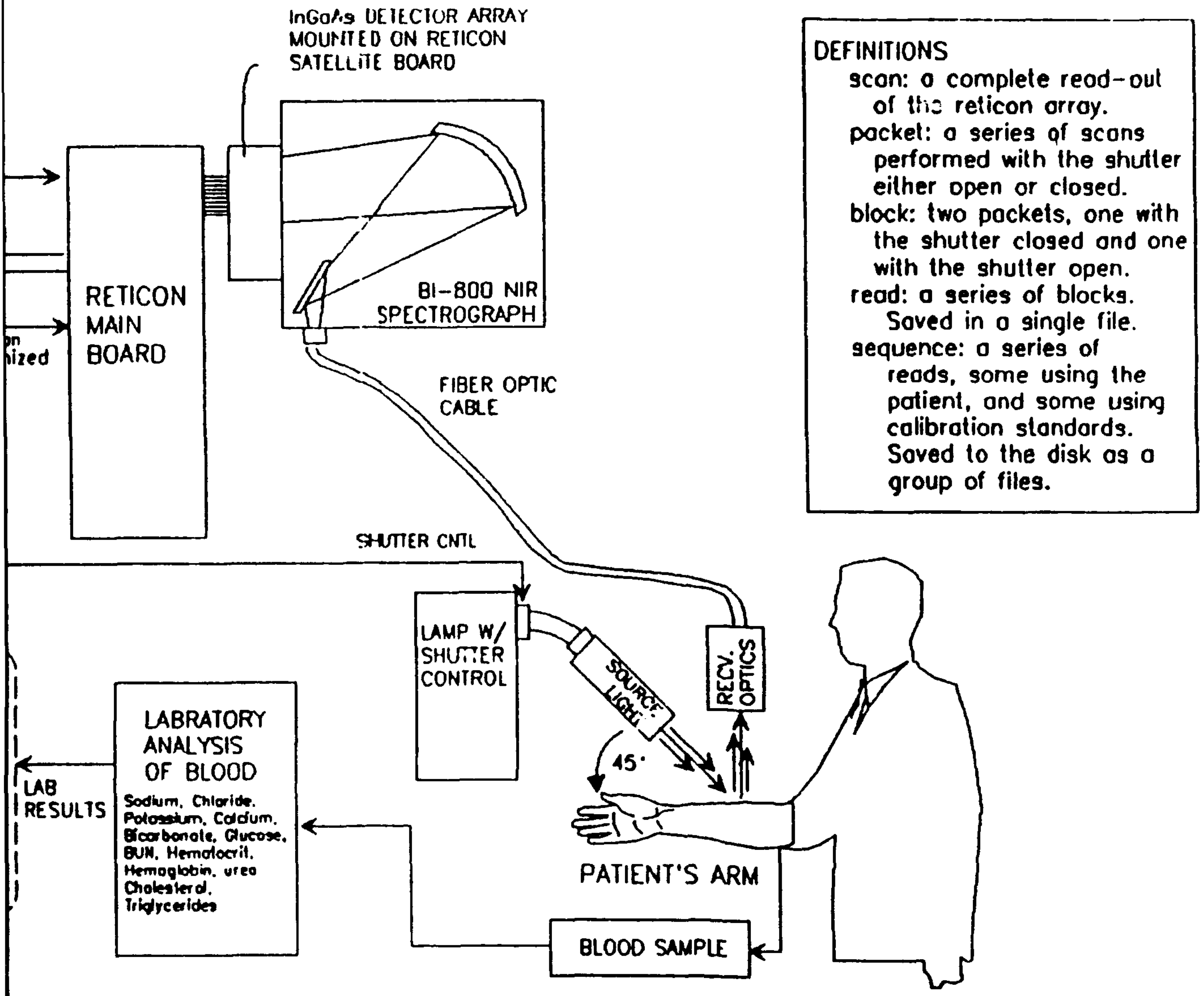


Biotronics Technologies, Inc.

SCALE:	APPROVED BY:	DRAWN BY: AGJ
DATE: 11-27-91		REVISED:

BI-800 NIR ARRAY SPECTROMETER SYSTEM BLOCK DIAGRAM

PAGE 1 OF 2	SOURCE #	DRAWING #



DEFINITIONS

scan: a complete read-out of the reticon array.

packet: a series of scans performed with the shutter either open or closed.

block: two packets, one with the shutter closed and one with the shutter open.

read: a series of blocks. Saved in a single file.

sequence: a series of reads, some using the patient, and some using calibration standards. Saved to the disk as a group of files.

Biotronics Technologies, Inc.		
SCALE: DATE: 11-27-91	APPROVED BY:	DRAWN BY: ACJ REVISED:

BI-800 NIR ARRAY SPECTROMETER SYSTEM BLOCK DIAGRAM

PAGE 2 OF 2	SOURCE #	DRAWING #

ANALYTE SUMMARY

ANALYTE TYPE	TRAINING METHOD[2]	TEST SET		TRAINING SET	
		% ERROR[1]	t-VALUE	% ERROR[1]	t-VALUE
CALCIUM	NN	3.15	0.053	N/A	2.33
CALCIUM	BP	2.85	0.43	3.25	1.93
CALCIUM	NG	2.62	1.50	3.15	2.96
POTASSIUM	NN	6.88	-0.14	N/A	2.46
POTASSIUM	BP	5.28	1.56	8.22	2.93
POTASSIUM	NG	5.21	2.24	8.37	5.15
SODIUM	NN	1.23	2.0	N/A	2.44
SODIUM	BP	1.35	2.91	1.32	8.32
SODIUM	NG	0.90	3.16	1.66	3.08
BUN	NN	N/A	N/A	N/A	4.02
BUN	BP	25.7	2.15	32.67	5.9
BUN	NG	23.81	3.65	30.65	4.46
GLUCOSE	NG	30.63	3.62	43.98	3.04
GLUCOSE GROUP 1	NG	9.36	17.2	37.9	5.2

[1] All errors are listed in average percent error relative to the mean.

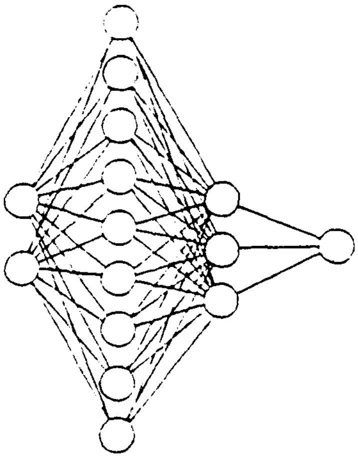
[2] Abbreviated training methods: NN = nearest neighbor, BP = straight backpropagation, NG = neurogenetics.

TABLE 1.

APPENDIX I

NETGEN

An integrated neural learning system for pattern recognition and classification



Timothy L. Ruchti
Biotronics Technologies, Inc.

September, 1991

I. INTRODUCTION

This report describes the use and application of the neural network software system called NETGEN developed by Biotronics Technologies, Inc. The purpose of this system is to apply artificial neural network (ANN) technology to difficult problems in pattern recognition and classification in which more conventional approaches have failed to uncover the underlying relationships between observable variables and desired information. The novelty of this package is the integration of two evolving technologies: *backpropagation* and *genetic algorithms*, into a hybrid learning system. The resulting system exhibits both an improved efficiency in processing time and a greater ability to recognize and predict patterns on the basis of a restricted training set.

Similar to other systems employing ANNs, one measure of "relative fitness" incorporated into this system is the average prediction error of a network over a particular set of samples. This measurement provides information concerning the closeness of the predictions to the actual values *on the average*. However, frequently an equally important consideration is the ability of the predictor to "track" the variable of interest over a particular range of values. Consequently, NETGEN also measures the tracking ability of each network over the provided sample sets by the so-called *t*-value. During training, the average error and *t*-value are displayed allowing the user to terminate the update process before overtraining occurs.

II. BACKGROUND

The pattern recognition problem is one of determining a mapping or a discriminant from a set of input variables to a set of output or desired variables on the basis of observed input/output samples. A desirable characteristic of such a transformation is the ability to generalize so that accurate output estimates can be made on the basis of a novel set of input variables. In applications, such as instrumentation, where the accuracy of output estimates is critical, the requirements on a pattern recognition becomes demanding. When the underlying relationship between the measurement and the output variable(s) are not apparent or when the transformation is nonlinear the problem becomes extremely complicated. An emerging technology which shows considerable promise to the problem defined above is that of artificial neural networks (ANNs).

Inspired from explorations in the life sciences, ANNs exploit known properties of biological neural networks to solve complex problems in which the human brain outperforms current technology. The resulting computational tool is characterized by a massively parallel structure composed of relatively simple but nonlinear processing elements. As a result of this parallel organization and inherent nonlinearity, ANNs posses the ability to model any nonlinear mapping to any degree of accuracy with properly selected parameters (e.g., weights) [9]. Therefore, it can be concluded that ANNs are structurally capable of solving the nonlinear mapping problem. Hence, assuming a proper learning algorithm, ANNs can be expected to perform functional approximation and signal filtering operations which are beyond the limits of optimal linear techniques.

The method of learning or selecting the parameters on the basis of input/output sample measurements, however, is not straight forward, and currently limits the utilization of this technology. As a result, new methods must be developed which supply the ANN with the ability to learn an appropriate set of parameters before widespread application of this technology will occur. The objective of this section, and the methods presented in this report is the development of more efficient and effective learning system for a general ANN structure. While the ability of a given ANN structure to model a transformation is necessary for successful application, the ability of the learning algorithm to improve towards an optimal parameter selection is equally important.

The following sections introduce the type of ANN employed and the two learning paradigms that have been integrate to form NETGEN. In addition, a brief background discussion is provided on the methods of performance measurement utilized in NETGEN and the data transformation called pre-scaling which is necessary before applying NETGEN to a data set.

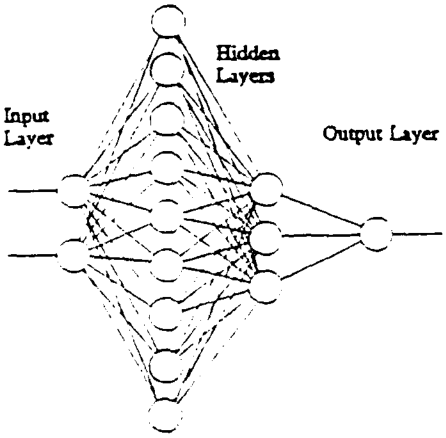


Figure 2 - *Typical four-layer feedforward artificial neural network denoted 2-9-3-1.*

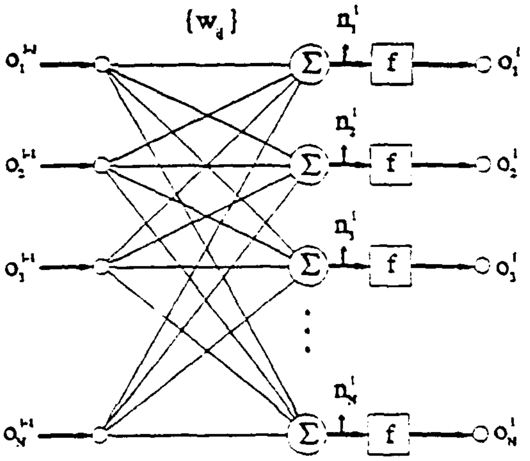


Figure 3 - *Detail of a single layer of a feedforward artificial neural network with N inputs and M outputs.*

A. MULTI-LAYER FEEDFORWARD ANNS

One of the most exciting discoveries in pattern recognition was that of the perceptron which autonomously recognized and classified patterns using a network of processing elements arrayed in a single layer [16]. However, this discovery was set back by the finding that a single layer of perceptrons could only distinguish linearly separable patterns and was unable to model nonlinear functions [11]. More sophisticated classifiers consisting of multiple layers of perceptrons were proposed but until recently were not used because effective training algorithms were not available [13]. As discussed above, given capable training methods, recent investigations have shown that a four-layer (counting the input layer) perceptron can form arbitrarily complex decision regions and can separate meshed classes. In fact, this type of network can form regions more complex than those formed using mixture distributions and nearest-neighbor classifiers [3].

The ANN structure employed by NETGEN is a generalization of the three-layer perceptron termed a *multiple-layer feedforward ANN*. The network consists of an input layer, a number of hidden layers and an output layer each composed of a number of processing elements called neurons. In this report the convention for naming a network structure will be: N-H_1-H_2- . . . -H_n-O where N is the number of neurons in the input layer, H_i is the number of neurons in the *i*th hidden layer and O is the number of neurons in the output layer. It is the convention in this report to count the input layer in the total number of layers although its function is somewhat different from the other layers as discussed below.

The input layer is simply a fanout device which accepts an input vector of equal dimension to itself and provides the subsequent layer access to it. The rest of the layers, including the output layer, consist of a number of neurons and weight connections from the previous layer's neuronal outputs to its own inputs. Each weight connection has associated with it an adjustable value which is multiplied by the value present on its input and relayed to the neuron upon which it resides. Each neuron performs two operations: the summation the input vector elements scaled by the values of the weights connected to produce its activation level or state and the transformation of its activation level via an activation function. An illustrative depiction of a single layer is shown in Figure 3. Each output is calculated by propagating the input vector through the network according to:

$$\begin{aligned} n_j^l &= \sum_{i=1}^{N} w_{ij}^l o_i^{l-1} \\ o_j^l &= f(n_j^l) \end{aligned} \tag{1}$$

where

o_j^l - the *j*th component of the output vector of the *l*th layer

n_j^l - the *j*th component of the state vector of the *l*th layer

f() - the *activation function*

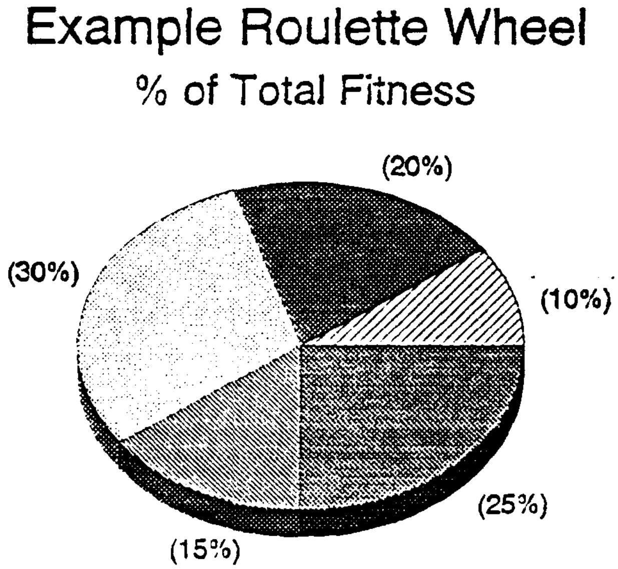
w_{ij}^l - the weight value from the *i*th neuron of the (*l*-1)th layer to the *j*th neuron of the *l*th layer

For simplicity, this is written in vector notation as:

$$\begin{aligned} n^l &= Wo^{l-1} \\ o^l &= f(n^l) \end{aligned} \tag{2}$$

Given an input vector, which will be denoted by *x*, the output vector is found by presenting *x* to the input layer and propagating o^1 through each subsequent layer according to equation (2) to produce o^L where *L* represents the number of layers. For future discussion this output will represented by the vector *y*.

The activation function can be any linear or nonlinear function. However, for the representation of nonlinear mappings and the classification of patterns which are not linearly separable f() must be nonlinear. In fact, there is no advantage of using more than two layers if f() is linear. Typical functions include the signum and sigmoidal functions although in this report the sigmoidal function is considered exclusively because networks employing them are capable of forming internal representations that are more complex than the simple binary codes produced by the signum functions and can form more complicated decision regions [12]. The sigmoidal activation function is given by a fitness level and then normalized to a value equal to its fraction of the sum total of all fitness levels of the population. Each member is then given a pie-shaped space on a roulette wheel, the size of which is indicated by its fraction of the total fitness as shown by example in Figure 4. A random number is then generated for each selection representing a distinct place on the roulettes wheel. Since each space on the roulette wheel has an equal chance of being selected members that fill larger percentages of the wheel have a greater chance of being selected.



	Population	Fitness	Fraction of tlt
▨	Member 1	1.5	0.10
■	Member 2	3.0	0.20
░	Member 3	4.5	0.30
▧	Member 4	2.25	0.15
▓	Member 5	3.75	0.25
	Total	15.0	1.0

Figure 4 *Example of a roulette wheel assignments based on a five member population.*

Crossover

Crossover is the actual combination of two chromosome in which two offspring are formed from different parts of the parents. The actual method of crossover explains its name and is the most important operation of the GA. In one-point crossover a random chromosome position is picked and the section following this position is exchanged between the two parents forming different offspring. As an example consider the illustration presented in Box 2 in which two parent chromosomes, each 7 bits long, are being combined through crossover after their fourth bit to produce two fairly different offspring. The word fairly is used because, depending upon the genetic encoding, some of the genes between the parents and offspring are similar and each offspring contains alles which are identical to each of the parents.

$$o = \frac{1}{1 + e^{-nT}} \qquad (3)$$

where o is the function output, n is the function input and T is the temperature of the neuron representing its excitability.

In addition, a threshold or offset may be specified with a layer which adds a single input, set to one, through an adjustable weight to each neuron in the layer. The threshold operates only on its respective layer and the weight it passes though is adjusted in the same manner as all other network weights.

A principle problem encountered when using a feedforward artificial neural network is selecting the number of layers and neurons in each hidden layer to be used. Clearly the number of input and output neurons are specified by the mapping problem, however, the number of hidden layers and the number of neurons in each layer must be determined. Automated techniques are being developed which use the same genetic algorithm introduced later in this report to determine an optimal structure [6], however, the most common method is trial and error. By Kolmogorov's Theorem, a continuous mapping from n-space to m-space can be implemented exactly by a three-layer ANN having n-fanout neurons in the first layer, $(2n+1)$ neurons in the hidden layer, and m neurons in the output layer [7]. Although this may not be the optimal network, especially when generalization from a small number of training samples is required, it is a good starting point for a three-layer network.

If a four-layer network is utilized, a rule of thumb is to use three times as many neurons in the first hidden layer as are in the second [9]. The number of neurons in the second layer is dependent on the complexity of the problem and the amount of data available for training. Too large of a hidden layer will cause the network to memorize the input/output pattern rather than learning the discriminant. In addition, if a large number of neurons are selected with insufficient training samples, the weights will not be reliably determined regardless of the training method. On the other hand, if too few neurons are used, the network will be unable to represent the desired transformation.

B. LEARNING BY BACKPROPAGATION

The utility of feedforward ANNs as a discriminant or transformation in ultimately dependent on the successful selection of its parameters. Given the ANN structure, that is the number of layers, neurons per layer, and the activation function, the adjustable parameters of the network consist of the weights or synapses of each layer and the temperature associated with the activation function. Since T is multiplied by the state prior to calculating the layer output its adjustment has the effect of scaling the weight vector residing on the individual neuron. Therefore, the temperature will be assumed constant and all parameter adjustments will be performed upon the weights.

As previously mentioned, multiple layer ANNs pose a considerable training problem. The main reason for this is that in a single layer network the effect of a weight adjustment is directly measurable via the prediction error in the output vector. However, in multi-layer networks the effect of given weight on the output cannot be directly seen and weight adjustment is not straight forward since generally it is not known what the output of each hidden layer should be. That is, there is no direct method of computing a prediction error which reflects the maladjustment or misalignment of a particular weight.

A solution to this problem was found independently by Werbos [14] and Rumelhart [13] in what is now the most popular ANN training method called *backpropagation*. Backpropagation is a training algorithm which seeks to minimize the mean square prediction error of the network over an entire set of training samples Thus, the mean square error acts as the cost function and is given by

$$E_j = \frac{1}{2}\sum_{i=1}^{P} | d_j(i) - y_j(i) | \tag{4}$$

where E_j refers to the *j*th element of the output error vector *E*, $d_j(i)$ is the *j*th desired output variable of the *i*th sample, $y_j(i)$ is the *j*th predicted output variable of the *i*th sample, and *P* is the number of samples. The strategy followed by backpropagation is to adjust each weight in the opposite direction of the gradient of the instantaneous prediction error with respect to itself according to

$$\Delta w_{ij}^l = -\alpha \frac{\partial E}{\partial w_{ij}^l} \tag{5}$$

where α is a positive scaler known as the learning rate. The smaller the learning rate is the closer the method comes to actually performing a true descent along the gradient of the error space defined by (4). This technique is well know as the method of gradient descent. The problem, as mentioned before, is the determination of the partial derivative of the prediction error with a weight residing on a hidden layer.

The solution to this problem revolutionized ANNs and is summarized by the following

$$\Delta w_{ij}^l = \alpha \delta_j o_i^{l-1} \tag{6}$$

where, in the output layer

$$\delta_j^l = (d_j - y_j) \frac{df}{dn_j^l}(n_j^l) \tag{7}$$

and in any other layer

$$\delta_j^l = \frac{df}{dn_j^l}(n_j^l) \sum_{k=1}^{N} \delta_k^{l+1} w_{kj}^{l+1} \tag{8}$$

The notation for the weight matrices and the activation and output variables in the above equations is defined in the previous section. Notice that it is necessary that the derivative of the activation function exist and is known which is another reason for using the sigmoidal activation function.

In vector notation the general meaning of these equations can more clearly seen:

$$\Delta W^l = \alpha \delta^l o^{l-1^T} \tag{9}$$

Here δ is a backpropagated error vector and o^{l-1} is the input to the layer. The error vector is calculated by

$$\delta^{l} = (d - y)\, I\, \frac{df}{dn^{l}}(n^{l}) \tag{10}$$

in the output layer and by

$$\delta^{l} = \frac{df}{dn^{l}}(n^{l})\, I\, W^{l+1^{T}}\delta^{l+1} \tag{11}$$

in all other layers, where *I* is the identity matrix. As can be seen, each layers δ is propagated backwards to the prior layer before being scaled by the derivative of the activation function. Hence, backpropagation refers to the method of propagating the error backwards through the network and then calculating the partial derivatives of the output error with respect to the respective weights.

Since true gradient descent learning requires infinitesimal steps to be taken, the procedure outlined above can take considerable time to adjust weights. One method that increases the learning rate and at the same time reduces the likelihood of staying in a local minima is a momentum term. The momentum is included in the update equation (6) as follows

$$\Delta w_{ij}^{l}(k+1) = \alpha \delta_{j}^{l} o_{i}^{l+1} + \eta \Delta w_{ij}^{l}(n) \tag{12}$$

where η is the momentum constant which is greater than or equal to zero and less than one but i typically set in the neighborhood of 0.9.

The basic operation of the momentum term is to incorporate a portion of the last change made to a weight into the next weight update. Specifically, the change made to a particular weight during the previous training cycle is multiplied by the momentum factor and added to the current weight change. Hence, whenever a change is made to a weight during a training iteration, it will be made decreasingly over and over to the weight during subsequent training cycles. Several training cycles produce a collective momentum which will cause the weight to change in the desired direction very quickly.

A very general algorithm, which is followed by NETGEN, for implementing learning by backpropagation is summarized in Box 1. For more information concerning the details of this algorithm the reader is refereed to 9 and 13 and the listing at the end of this report. From this algorithm a training cycle is defined by steps 2-5 while a training iteration will be considered steps 2-6.

As discussed previously, backpropagation is an approximation of the method of gradient descent. As a result, training can take considerable processing time if α is selected small to ensure convergence or convergence may not occur at all if α is selected to large. In addition, while the inclusion of the nonlinearity provides the feedforward structure with the properties which make it advantageous, it also causes multiple local minima in the error space which are indistinguishable by a gradient descent leaning algorithm. These two problems have made the consideration of other learning procedures necessary.

C. GENETIC ALGORITHMS

Genetic algorithms (GAs) attempt to solve a given optimization problem by applying the general principles of natural selection and the mechanics of natural genetics to a population of possible solutions. Survival of the fittest tactics and random recombination operations are

BACKPROPAGATION ALGORITHM

1. Initialize the weights of the artificial neural network
2. Propagate a training set sample through the network via (2)
3. Calculate the error between the desired output and the actual output
4. Backpropagate the error according to (7), (8)
5. Adjust the weights of the network according to (12)
6. Repeat 2-5 for the entire sample set
7. Repeat 2-6 until the specified error tolerance is satisfied

Box 1 *Simple implementation of backpropagation algorithm.*

applied to a population of possible solutions to produce a global search which is extremely fascinating - and - extremely useful. In each training iteration, termed a generation, population members which are the most optimal or the most fit are recombined to produce a new population of offspring. This section provides a brief introduction to the basic principles and algortihms involved. The reader is referred to [4] and [2] for more information on this topic.

Different calculus based methods of optimization, such as gradient descent discussed above, seek local optimal by updating solutions based on the direction of the local gradient. The problems with this type of search technique are: they seek the best local point rather than a global point and they depend on the existence of derivatives. It is often the case that an optimization problem will contain a multitude of local minima (especially nonlinear ANNs), or the function derivative cannot be determined quantitatively, or the function is extremely noisy causing the derivative function to bias the solution in the wrong direction. Genetic algorithms are different in that:

1. They work on an encoding of the parameter set an can therefore be applied to a wide variety of problems
2. They search from a population of points rather than a single random point causing the search to be global
3. They do not use derivatives but simply require a fitness evaluation of each population member
4. They employ a probalistic search technique rather than deterministic which causes portions of the solution space to be considered which would normally be bypassed (in a sense they perform a creative search).

The general principles discussed in the subsequent paragraphs follow from the work of Goldberg [4] and Davis [2]. Some discrepancy between the two authors does exist in the naming of the basic operations implemented by a typical GA although both agree upon the nature of these operations. Because much of Davis' work was used in NETGEN his naming conventions will be used throughout this report.

ENCODING

The first notion of GA that must be understood is that of encoding. Very loosely, one may state that the structure of every living being is encoded within their chromosomes consisting of a long string of genes. Each gene may be thought of as a parameter representing some characteristic of the individual. Following this concept GAs operate on a population of "chromosomes" composed of binary '1's and '0's, called alles. which in some way represent the parameters of the function being optimized. Groupings of alles may be thought of as genes which directly or indirectly correspond to a particular parameter.

As an example one may consider the problem of minimizing some scaler function *f*(*x*), where x takes on values between ±1. The encoding process consists of determining how many bits, *m*, to represent *x* with and then mapping the real number $x\epsilon(-1,+1)$ into an integer value between zero and (2^n-1). Thus, the chromosome in this case consists of one gene of *m* bits. In a similar manner any optimization problem can be coupled to a genetic algorithm using an encoding process which maps its parameters into integer values which are linked together to form a single binary chromosome string.

EVALUATION

A second item that must be prepared before implementing a GA is a method to decode the chromosomes and to evaluate the fitness of a particular solution called a population member. Naturally, a given member of the animal kingdom is evaluated through its interaction with the environment which will determine its likelihood to survive and reproduce. The evaluation of a given chromosome by the function being optimized is analogous to this. In the example given above it simply means mapping each gene back into real space and then determining *f*(*x*). The evaluation of a population member may be though of as calculating the cost of the member or, more appropriately, the fitness of the member.

REPRODUCTION

The point at which evolution is actually thought to take place in during reproduction, when the chromosomes of different population members combine. During this recombination process random mutations may take place adding diversity to the population and additional characteristics. There are three basic operations which genetic algorithms utilize which perform the function of reproduction:

Selection

Given a population whose members have been evaluated, the selection process, also called reproduction by Goldberg, matches "couples" from the population for reproduction. Selection is performed randomly in a manner such that members with higher fitness levels have a grater chance of being selected and some members are selected more than once while others are not selected and therefore are removed from the population. The method of selection is described as a roulette wheel technique as follows. Each member of the population is assigned

ONE-POINT CROSSOVER

```
PARENT A:  0 1 0 0 0 0 1              CHILD 1  0 1 0 0 1 0 1
PARENT B:  1 0 0 1 1 0 1              CHILD 2  1 0 0 1 0 0 1
                 ↑
              CROSSOVER
              POINT
```

Box 2 *Example of one-point crossover.*

Two-point crossover, illustrated in Box 3, is similar to one-point except that two random points are selected instead of one and the information between the selected crossover points are exchanged. The principle advantage of two-point crossover is that beneficial traits, which may be encoded at the ends of a chromosome are not necessarily lost by slitting the chromosome in half.

TWO-POINT CROSSOVER

```
PARENT A:  0 1 0 0 0 0 1              CHILD 1  0 1 0 1 1 0 1
PARENT B:  1 0 0 1 1 0 1              CHILD 2  1 0 0 0 0 0 1
              ↑     ↑
             PT 1  PT 2
```

Box 3 *Example of two-point crossover.*

Crossover is the heart of the GA causing an efficient and effective global search as discussed below.

Mutation

Mutation occurs naturally when the genetic code of an individual is altered producing traits that are uncharacteristic of the population. Essentially this introduces new genetic material or new characteristics which can be considered for its suitability as a solution. While naturally no mutation has ever been observed which is advantageous to the survival of the creature, in a GA, introducing mutation adds a certain diversity to the population which lends itself to the global search, avoiding a stagnating premature convergence and "creatively" pursuing alternate solutions. The mutation operation is performed on a population member by randomly setting its chromosome bits to '0' or '1' at a rate of M_rate, where M_rate typically is one in one-thousand or 0.001.

Algorithm

The most simple GA is presented in Box 4 consisting primarily of the following reproductive operations: selection, crossover, and mutation. While this technique seems somewhat simple and random, it has been theoretically shown to provide a highly efficient heuristic for information gathering in complex spaces [8]. In addition, a number of experimental studies have shown that the GA's of this type exhibit impressive efficiency in practice.

To analyze the convergence properties of a GA Holland [8] developed the notion of a schema which represents a subgroup of chromosomes from the group of total possible chromosome bit encodings. A group of chromosomes consists of all those that match a particular schemata in all the bit locations with '0's and '1's but have either of the binary numbers in the bit locations where the schemata has a don't care symbol '#'. In the fundamental theorem of GAs Goldberg proves that high performance schema will be selected an exponentially increasing number of times in succeeding generations. While this is occurring, new schema are constantly being introduced into the population via the reproduction operations. As a result, higher quality schemata are introduced proving the ability of a GA to improve.

BASIC GENETIC ALGORITHM

1. Initialize the population of chromosomes
2. Evaluate each population member for fitness
3. *Reproduce* to from a new generation
    a. *Select* parent couples
    b. *Crossover* parents to produce an equal number of offspring
    c. Perform the *mutation* operator on the offspring
4. *Evaluate* each member of the new generation
5. Either stop the search or repeat 3-4

Box 4 *Simple genetic algorithm consisting of the operations selection, crossover, mutation, and evaluation.*

Improvements

There are many improvements and "ad hoc" features that make GAs more effective which are in common use. Most of these, including the few summarized in this section can be found in Davis' text Handbook of Genetic Algorithms [2]. The first deals with the scaling of the fitness of each member through *windowing*. Essentially windowing produces a window covering the range of fitness levels of the population over the most current generations and assigns each population member a fitness level based upon its position within the window. In this manner, populations whose fitness levels are grouped in high values, say between 99 and 100 can be more easily distinguished and the good members are assigned larger portions of the roulette wheel. As a result, higher quality schema have a tendency to reproduce more often.

In the discussions above each generation replaces succeeding generations completely.

Because of this, the best individual can be lost through crossover or mutation. *Elitism* is a strategy which copies the best structure into each succeeding generation.

Another alternative to the tradition algorithm is to replace only a portion of the population, given by a fraction called the generation gap, with the new offspring. A somewhat similar setting is the *crossover rate* which indicates the number of selected parents perform the crossover operation on. The generation gap guarantees that a group of parents will survive, usually constrained to be distinct or the best of the population. The crossover rate allows the selection operation to be performed on the entire generation and simply disables the crossover operation on a portion of the selected parents. The principle reason for each of settings is to prevent premature convergence of the population, or, in other words, to maintain diversity.

D. A HYBRID LEARNING SYSTEM

Due to its ability to generalize and avoid local minimal points it seems natural to employ GAs to the training of ANNs. In fact, considering the large search space of possible weights for a complicated ANNs, it can be expected that GAs will perform better than methods that start from a single random point and search locally for the best configuration. By treating the training of an ANN as an optimization problem, any ANN, whose performance is quantifiable, can be trained using GAs (including ANNs other than feedforward networks).

At the same time, however, it may also be beneficial to utilize backpropagation to develop a training method which uses the best features of both methods to produce an algorithm which is better than either one individually. The main reason for this is that while genetics algorithms perform global searches extremely effectively and efficiently, their random selection process cannot outperform the deterministic search of backpropagation on a purely local problem (i.e., a problem with tight constraints). Hence, the two very different optimization techniques can be combined to produce a hybrid system which converges very quickly near a global minima and then "fine tunes" the answer using backpropagation.

This section describes a hybrid algorithm which follows in principle from the work of [15] and [1] but was implemented via the neural network software developed by Biotronics Technologies, Inc. and the GENESIS [5] genetic algorithm package. For more information concerning the details of the genetic algorithm components discussed below the reader is referred to the GENESIS manual. The documented source codes of the final NETGEN package are provided at the end of this report. Then topics considered below are the encoding of the neural network parameters into a binary chromosome and the population evaluation procedure. Very general algorithms are included to give the reader the "big picture" of neurogentics.

ENCODING THE CHROMOSOMES

Encoding an ANN into a binary chromosome consists of associating each weight of the ANN (and any other ANN parameter for that a matter) with a single, distinct gene of the chromosome. For a given experiment, the gene length in bits, defining the resolution of each weight, must be constant. However, the bit length of the genes may vary from experiment to experiment depending on the required accuracy of the weights for a given application. Since each weight is a number, represented in floating point format, it must be converted to an integer in the range of 0 to 2^b-1 where *b* is the number bits in the weights gene representation. Defining the integer, g_{ij}, as the gene value associated with weight w_{ij}, the transformation from "weight space" to "gene space", known as packing, occurs according to

$$g_{ij} = \text{integer}\left[(2^{b} - 1) \frac{w_{ij} - w_{min}}{w_{max} - w_{min}} \right] \qquad (13)$$

where w_{min} and w_{max} are the minimum and maximum allowable values the ANN weights can assume and b is the number of ailes or bits in the gene g_{ij}.

In actual implementation, all of the weights of the ANN are grouped into one long string known as the floating point representation of the chromosome. In this representation the weights are in the same order as their associated binary genes which comprise the chromosomes, in "packed representation", upon which the genetic algorithm operates. The method of returning a given set of weights selected by the genetic algorithm consists of "unpacking" the binary coded chromosome via (13) in the floating point string accessed by the ANN.

THE EVALUATION FUNCTION

The average prediction error of a given network over the entire training set is the most basic performance measure of almost any ANN and is used to test the "fitness" of a given chromosome. The details of this measure are discussed below. The method of evaluating a population, then, consists of individually unpacking each population member into the floating point chromosome representation accessed by the network and evaluating the network over the entire training set and returning to the GA the average prediction error.

A SIMPLE NEUROGENTIC ALGORITHMS

With the encoding and evaluation procedure defined above, a simple genetic algorithm is presented in Box 5. While the actual method of evaluation and perhaps encoding may change from network to network, the actual genetic algorithm manipulating the binary chromosomes are the same. In experiments using NETGEN it was observed that this algorithm quickly found a group of weights close to a minimum point but then convergence proceeded quite slowly. To increase the rate of convergence, backpropagation is added to the algorithm as discussed below.

ADDING BACKPROPAGATION

Although training each member of the network could be performed between almost any step in Box 5, the most efficient point was determined to be in the evaluation procedure since each population member is already being iteratively tested at this point. The algorithm actually implemented is given in Box 6. The biggest difference in this algorithm compared to that of Box 5 is that the evaluation procedure actually modifies the chromosomes prior to evaluating them. This, however, will not effect the actual genetic operations since they do not use any past history when manipulating the current generation.

The overhead added to the training system is a significant increase in processing time. If, for example, each network is trained 5 iterations per generation and the population size is 500, then each generation contains 2500 backpropagation training iterations. A somewhat different strategy which is effective in many cases but does not involve the sizeable increase in processing time is to use the simple algorithm presented in Box 5 to get close to a good solution and then apply backpropagation to the best set of weights to "fine-tune" the solution.

A NEUROGENETIC ALGORITHM

1. Initialize the population of ANNs in packed representation
2. Evaluate each ANN as discussed above for fitness
   a. individually "unpack" each chromosome into the floating point weight matrices of the network
   b. determine the average prediction error over the entire population
3. *Reproduce* to form a new generation
   a. *Select* parent couples
   b. *Crossover* parents to produce an equal number of offspring
   c. Perform the *mutation* operator on the offspring
4. *Evaluate* each member of the new generation as per step 2
5. Either stop the search or repeat 3-4

Box 5 *Simple genetic algorithm applied to an artificial neural network training problem consisting of the operations selection, crossover, mutation, and evaluation.*

SAMPLE SELECTION

As mentioned in the section on encoding the network, any network parameter may be selected as a variable to be encoded into a particular chromosome for optimization. One particular variable that is of interest in pattern recognition is the set of training samples. If it is expected that a certain percentage of the training set does not accurately represent the actual input/output variable relationship then methods must be employed to select those samples for training which will cause the network to generalize. The method suggested in this report is to add a gene for each sample to be selected, at the end of each chromosome, which is encoded as discussed above That is, each sample is numbered with a floating point number and each binary gene is encoded analogous to (13) except that the minimum value is one and the maximum is the number of samples in the sample set. Evaluation and training, then, will occur only on the samples encoded in each chromosome.

Implementation of the sample select procedure must include a utility to constrain the selected samples for a given chromosome to be distinct. To accomplish this, prior to evaluation, each one of the sample select genes is examined to verify that the population member being evaluated has entirely distinct sample selections. If a selection is found which is not distinct, which occurs quite often following crossover, it is replace by a random value between 1 and the number of samples to be selected and then re-examined for distinctness. In addition, the sample

HYBRID NEUROGENETIC ALGORITHM

1. Initialize the population of ANNs in packed representation
2. Evaluate each ANN as discussed above for fitness
   a. individually "unpack" each chromosome into the floating point weight matrices of the network
   b. train the unpacked network for *num_its* iterations
   c. re-pack the network weights into the binary chromosome
   d. determine the average prediction error over the entire population
3. *Reproduce* to form a new generation
   a. *Select* parent couples
   b. *Crossover* parents to produce an equal number of offspring
   c. Perform the *mutation* operator on the offspring
4. *Evaluate* each member of the new generation as per step 2
5. Either stop the search or repeat 3-4

Box 6 *Hybrid genetic algorithm including backpropagation prior to each evaluation.*

selects are always truncated to an integer value and then increased to that integer value plus 0.5 prior to re-packing. The reason for this is that the resolution of the representation of a floating point number within the genetic algorithm is limited by the number of bits specified for each gene. Thus, a number such as 5.0 might be modified to 4.98 by the genetic algorithm during packing and subsequently truncated to 4 rather than 5 when the sample is recalled.

INPUT VARIABLES SELECTION

When a large number of input variables are provided with each sample it is advantageous to select a fraction of the entire set for network efficiency and to enhance the ability of the network to generalize (i.e., decrease the degrees of freedom to a size such that the parameters can be reliably determined from the size of the training set). When the input/output relationship is linear this is most easily accomplished through correlation techniques. However, when the relationship is nonlinear better methods must be developed.

The method of selecting input variables proposed in this report is to first encode a number of genes (equal to the number of desired input variable selections) with an input variable number at the end of each chromosome in the population. The method of encoding is the same as that of the sample select procedure outlined above. Then, only those inputs encoded on the chromosome of a particular population member are used to train and test the network. This method has been implemented in NETGEN and has been used with some success although it has not undergone rigorous testing.

The hinderance to this procedure is the fact that optimal weight selection is intimately linked with the inputs associated with the network. Hence, selecting a different input element for an existing set of weights will cause network performance to degrade. For this reason a strategy for selecting inputs included in NETGEN is to encode only the inputs into chromosomes and use the GA to select the input variables and nothing else. Evaluation then occurs by initializing an entirely new network, and training the network via backpropagation for a number of iterations which will allow convergence to begin. The returned network performance is the average prediction error after network training. The procedure, therefore, optimizes the input selections with the criteria of their ability to train the network. If a large population is used this technique may be particularly effective since a number of chromosomes may be identical but will start from different random points in the weight space. Since this technique is new, performance results are lacking although pattern recognition experiments have successfully selected input variables which, when later used with NETGEN to predict the output variable of interest outperform conventional linear techniques.

E. PERFORMANCE MEASUREMENT

Two performance measure are implemented with NETGEN defined below as the average prediction error and the *t*-value.

AVERAGE PREDICTION ERROR

The average prediction error of a network over an entire sample set is the most basic performance measure of almost any neural network or pattern recognition system. Given *n* desired or output values denoted y_i and *n* predicted values by the network denoted x_i, the average prediction error is simply

$$E = \frac{1}{n} \sum_{i=1}^{n} | x_i - y_i | \tag{14}$$

t-Value

The *t*-value is so named because a T-test on the slope relationship between the predicted and actual output values is performed. Calculation of the *t*-value is performed as follows: given *n* desired or output values denoted y_i and *n* predicted values denoted x_i, first form a linear regression model of x_i versus y_i:

$$b = \text{slope} = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2} \tag{15}$$

$$a = \text{intercept} = \bar{y} - b\bar{x}$$

Then the *t*-value is calculated through

$$T = \frac{b}{SE(b)} \tag{16}$$

where *SE(b)* is the standard error of the slope:

$$SE(b) = \sqrt{\frac{s^2}{\sum_{i=1}^{n} (x_i - \bar{x})}} \tag{17}$$

$$s^2 = \frac{1}{n-2} \sum_{i=1}^{n} (y_i - a - bx_i)^2$$

The examination of the *t*-value can provide more information concerning the convergence of a network than the average error and is a better test of the network performance when output variable tracking is critical.

F. PRESCALING

Within the neural network employed by NETGEN, the output of each layer is restricted to be between zero and one by a sigmoidal function. As a result, all input and output variables should be mapped into the range zero to one or even more preferably 0.1 to 0.9. The suggested method for scaling a variable y_i over *n* samples is as follows

$$\tilde{y}_i = 0.1 + 0.9 \frac{(y_i - y_{min})}{(y_{max} - y_{min})} \tag{18}$$

where y_{min} and y_{max} are the minimum and maximum expected or measure values y_i will assume. This method maps the range of y_i into the range of the neural network further increasing the chances that the network track the output variable rather than simply averaging.