

# SANDIA REPORT

SAND98-2145

Unlimited Release

Printed October 1998

## Character Recognition Using Genetically Trained Neural Networks

Keith M. Stantz, Cecilia Diniz, Michael W. Trahan, John S. Wagner

Prepared by

Sandia National Laboratories

Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

Approved for public release; further dissemination unlimited.



**Sandia National Laboratories**

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

**NOTICE:** This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof, or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof, or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from  
Office of Scientific and Technical Information  
P.O. Box 62  
Oak Ridge, TN 37831

Prices available from (615) 576-8401, FTS 626-8401

Available to the public from  
National Technical Information Service  
U.S. Department of Commerce  
5285 Port Royal Rd  
Springfield, VA 22161

NTIS price codes  
Printed copy: A03  
Microfiche copy: A01



SAND98-2145  
Unlimited Release  
Printed October 1998

## **Character Recognition Using Genetically Trained Neural Networks**

Keith M. Stantz, Cecilia Diniz<sup>α</sup>, Michael W. Trahan, John S. Wagner  
Pulsed Power and Laser Initiatives Department

Sandia National Laboratories  
P.O. Box 5800  
Albuquerque, NM 87185-1188

### **Abstract**

Computationally intelligent recognition of characters and symbols addresses a wide range of applications including foreign language translation and chemical formula identification. The combination of intelligent learning and optimization algorithms with layered neural structures offers powerful techniques for character recognition. These techniques were originally developed by Sandia National Laboratories for pattern and spectral analysis; however, their ability to optimize vast amounts of data make them ideal for character recognition. An adaptation of the Neural Network Designer software allows the user to create a neural network (NN) trained by a genetic algorithm (GA) that correctly identifies multiple distinct characters. The initial successful recognition of standard capital letters can be expanded to include chemical and mathematical symbols and alphabets of foreign languages, especially Arabic and Chinese.

The NN model constructed for this project uses a three layer feed-forward architecture. To facilitate the input of characters and symbols, a graphic user interface (GUI) has been developed to convert the traditional representation of each character or symbol to a bitmap. The  $8 \times 8$  bitmap representations used for these tests are mapped onto the input nodes of the feed-forward neural network (FFNN) in a one-to-one correspondence. The input nodes feed forward into a hidden layer, and the hidden layer feeds into five output nodes correlated to possible character outcomes. During the training period the GA optimizes the weights of the NN until it can successfully recognize distinct characters. Systematic deviations from the base design test the network's range of applicability. Increasing capacity, the number of letters to be recognized, requires a nonlinear increase in the number of hidden layer neurodes. Optimal character recognition performance necessitates a minimum threshold for the number of cases when genetically training the net. And, the amount of noise significantly degrades character recognition efficiency, some of which can be overcome by adding noise during training and optimizing the form of the network's activation function.

---

<sup>α</sup> Summer Intern from Bryn Mawr College



## Contents

1	Introduction .....	7
2	Materials and Methods .....	7
	2.1 NN Training Process .....	9
	2.2 NN Validation Process .....	12
3	Results .....	12
	3.1 Base Design .....	12
	3.1.1 Neural Network Connection Map .....	13
	3.1.2 Weights Histogram .....	13
	3.1.3 Fitness .....	14
	3.1.4 Brainscan .....	14
	3.1.5 Character Recognition Results .....	15
	3.1.6 Contour Plot .....	15
	3.2 Systematics .....	16
	3.2.1 The Number of Hidden Neurodes .....	16
	3.2.2 The Number of Cases .....	17
	3.2.3 Noise .....	17
	3.2.4 Form of the Activation Function .....	19
4	Discussion .....	22
5	Conclusions .....	24
6	Recommendations .....	26
	References .....	27
	Appendix A : Neural Net Designer Parameters .....	28
	Endnotes .....	30

## Figures

1	Outline of NN training and validation processes.	8
2	The top picture represents the GUI used to develop the character bitmaps of the five capital Roman letters used for the FFNN tests.	10
3	Sigmoid activation function.	10
4	Trained neural network and weights histogram.	13
5	Fitness plots.	14
6	Brainscan of the NN displaying the distribution of weights.	14
7	Probability that a letter will be correctly identified for letters in the initial character set.	15
8	Contour plot depicting correlations between letters.	15
9	The number of hidden layer neurodes and multiplicative factor needed as a function of the number of characters.	17
10	(A) Character recognition efficiency of all five letters when trained with 1, 5, 7, 10, 17, and 25 number of cases. (B) The number of generations until the superbug and average of population converges.	18
11	Top 4 plots: character recognition efficiency of each character as measured with 0 through 10 bits flipped and trained with {0}, {0,1}, {0,1,2}, and {0,1,2,3,4} bits flipped. Bottom plot: results when all 5 character efficiency results are averaged.	20
12	Change in character recognition efficiency and average outputs for different numbers of bits flipped during validation and training sets.	21
13	Character recognition efficiency with a different form of the activation function.	22

## Tables

1	Tabulated results measuring the convergence and performance of the genetically-trained net as a function of character set size and number of hidden layer neurodes.	16
---	---	----

## 1 Introduction

The recent development of intelligent learning and optimization algorithms offers an innovative approach to character and symbol recognition. Previous work with these algorithms suggests a strong potential for the pattern recognition capabilities required for identification of characters and symbols. There are many applications for this form of recognition, but the dual foci of this project ultimately relates to foreign language translation and chemical formula classification applications. These applications have been addressed in proposals.

One application includes development of “a new systematic technique for the translation and interpretation of foreign language text.”[4] <sup>1</sup> A succession of artificial neural networks (NNs) could be trained in character recognition for a target foreign alphabet, grammar, and structural rules of the language. Internal libraries would aid translation. The NN discussed in this paper focuses on the identification of capital letters from the Roman alphabet, but the same program can be used to develop multiple NNs for recognition of a wide variety of characters and symbols. In language translation, high value foreign languages including Chinese and Arabic are probable targets. The ability to recognize and categorize foreign language documents could provide a fast, reliable translation methodology applicable in many disciplines.

Another application involves the identification and classification of chemical formulas and equations [5]. This program would require a succession of NNs to encapsulate a chemical formula in a block of text, “read” the formula, and then classify the chemicals present in the formula or the type of reaction. Such a technique requires an extensive internal library of chemicals and mathematical symbols.

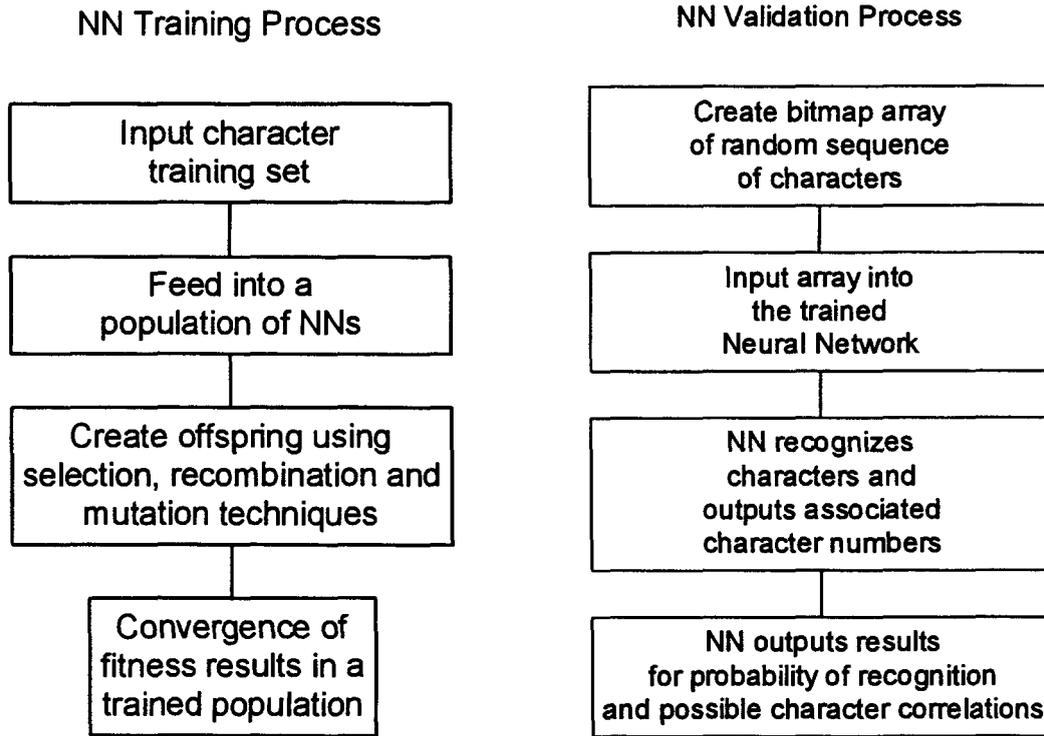
The learning capacities of individual NNs are subject to an upper bound, but multiple NNs can collectively perform complicated tasks. Ultimately, both of the proposed programs will require the development of a series of NNs with different capacities; however, both programs share the initial stage of character and symbol recognition. This paper focuses on the development of a feed-forward neural network which addresses this initial stage. The materials and methods of section 2 review the concepts of artificial neural networks and the genetic algorithms used to train them as realized by the neural network designer (NND). In section 3, results are tabulated and plotted. A discussion of these results ensue in section 4. Conclusions are summarized and recommendations are enumerated in sections 5 and 6, respectively.

This work constitutes a preliminary assessment, a feasibility study, on the use of genetically-trained neural networks on character and symbol identification problem domains. A brief assessment will be given at the end of the conclusions section.

## 2 Materials and Methods

The proposed technique for character and symbol recognition involves the implementation of a super neural structure. A super neural structure consists of “a hybrid layered information processing structure with at least one layer designed and trained by a genetic or evolutionary algorithm.”<sup>2</sup>

NNs [1,3] are “based on the known architecture of the brain, specifically the cerebral cortex,”<sup>3</sup> but the behavior of artificial NNs is significantly simpler than the activity of a real brain. There are approximately  $10^{11}$  neurons in the human brain. These neurons transmit a signal from one cell to another using a long nerve fiber known as an axon. The transmission will “raise or lower the electrical potential inside the body of the receiving cell. If this potential reaches a threshold, a pulse or action potential of fixed strength and duration is sent down the axon.”<sup>4</sup>



**Figure 1. Outline of NN training and validation processes.**

Using an activation function, the neurode of the artificial NN models the neuron’s binary threshold behavior. The excitatory or inhibitory behavior of the cell transmissions is modeled by the weights of the connections in the artificial NN. The fully connected architecture of artificial NNs does not emulate the complex connection schemes of biological NNs, thereby diminishing the applicability of artificial NNs as a model for the brain. However, artificial NN structure provides an intelligent method of efficient problem solving.

The internal weights of a NN’s structure undergo iterative adjustments during the optimization stage. There are two distinct methods of achieving optimization: unsupervised learning and supervised learning. In unsupervised learning, the “only available information is in the correlations of the input data or signals. The network is expected to create categories from these correlations, and to produce output signals corresponding to the input category.”<sup>5</sup> In contrast, supervised learning requires a direct comparison of the output of the network with known correct answers. The NN developed for character recognition has been trained using genetic algorithms in a supervised learning environment.

Genetic algorithms [2] (GAs) are “search procedures based on the mechanics of natural selection and natural genetics.”<sup>6</sup> Genetic training of neural networks produces a population of NNs rather

than a single network. Often the statistical variation of performance by the whole population provides additional information about the confidence and uncertainty of the population's answers. For a particular evaluation test set a "superbug" can be identified. These search procedures are stochastic; therefore, they do not require a smooth, continuous function in order to find the global optimum. GA optimization techniques evolve to yield the best possible solution known as the "superbug." There are four primary differences between genetic algorithms and more normal search and optimization procedures:

1. GAs work with a coding of the parameter set, not the parameters themselves.
2. GAs search from a population of points, not a single point.
3. GAs use payoff (objective function) information, not derivatives or other auxiliary knowledge.
4. GAs use probabilistic transition rules, not deterministic rules.<sup>7</sup>

GAs are described with the terminology of evolutionary genetics. The first generation consists of an entire population of potential solutions. Known as "bugs," each solution has unique "genetic material" thereby resulting in a wide range of performance capabilities. In the supervised learning environment, the solutions of each bug are compared to the desired solutions, and the fitness of the bug is evaluated. Applying the principle of the survival of the fittest, the most fit individuals experience a higher probability of reproduction. To create the next generation, the genetic material of the bugs is randomly combined using a crossover technique. Mutations are introduced with a pre-specified probability of occurrence. The increased probability of representation of fitter individuals results in an increasingly fit trend. However, the scope of the population allows a search of the entire solution space and diminishes the possibility of missing the global maximum in favor of a local maxima. This offers the GA a robust performance that is necessary for complicated systems with large amounts of information.

The character recognition analysis method consists of a (1) neural network training process followed by a (2) neural network validation process. An outline of these processes is presented in figure 1.

## 2.1 NN Training Process

The training process begins by producing a database of characters to be recognized. A graphics user interface (GUI), the top picture in figure 2, allows the user to develop many different character images and to save them as a bitmap. In these experiments, a character bitmap is an  $8 \times 8$  array of bits, where a +1.0 indicates an "on" bit and -1 an "off" bit. The database contains five capital Roman letters – A, B, C, D, and X – bottom of figure 2.

Next, the Neural Net Designer (NND) is called upon to develop the fully-connected, feed-forward neural network architecture (parameters listed in Appendix A). At the moment we will confine the NN architecture to a single hidden layer leaving two key parameters to be determined: the number of hidden-layer neurodes and the form of the activation function. By extracting the two-dimensional string of bits one row at a time<sup>β</sup>, a 64 bit vector represents the input pattern and defines the number of inputs. Each output represents a single character; therefore, the total number of possible outputs is 5. To determine the number of hidden layer neurodes, a commonly accepted equation,

---

<sup>β</sup> top to bottom, last bit of previous row followed by first bit of current row

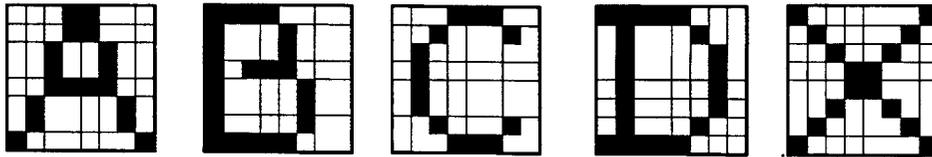
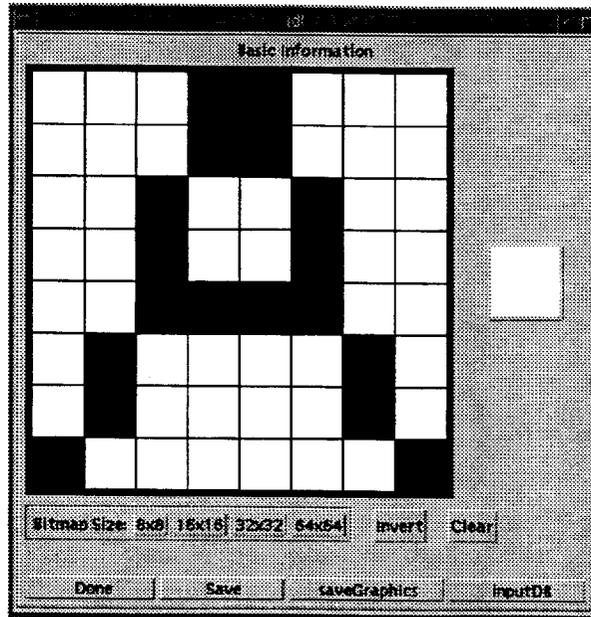


Figure 2. The top picture represents the GUI used to develop the character bitmaps. The bottom picture shows the 5 capital Roman letters used for the FFNN tests.

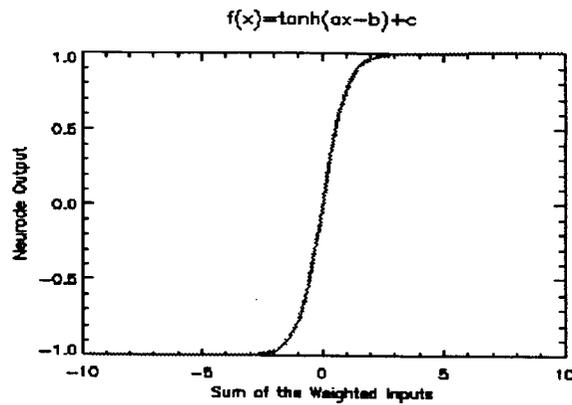


Figure 3. Sigmoid activation function (hyperbolic tangent) as chosen in the Neural Net Designer

$$nHLnodes = \sqrt{nInputs \cdot nOutputs}, \quad (1)$$

sets this value to 17.

The form of the activation function is typically sigmoidal. A hyperbolic tangent of the form

$$f(x) = \tanh(a \cdot x - b) + c \quad (2)$$

is chosen. Parameters  $b$  and  $c$  are set to zero. This positions the activation function such that when  $x < 0$ ,  $f(x) < 0$  and when  $x > 0$ ,  $f(x) > 0$ . Letting  $a = 5$  governs how fast the activated neurode approaches  $-1$  or  $+1$ . The form used is displayed in figure 3.

After developing a candidate character set and determining the architecture of the FFNN, each net within the population of 64 is shown a random sequence of 25 character images (the number of cases) and their corresponding outputs. The NN outputs are compared to the expected outputs (i.e., supervised learning). As an example, if the input vector is the image data for the letter B, the expected output vector is set to  $[-1, +1, -1, -1, -1]$ , where each output represents a letter as shown in figure 2. The network under the guidance of the GA “learns” how to recognize these patterns and categorize them as letters. The three GA operations used to optimize the net’s weights include selection, recombination, and mutation.

First, the parents are selected by implementing a technique called the roulette-wheel. Building the roulette-wheel is accomplished by calculating the percentage of fitness space occupied by each bug in the population,

$$P(\text{bug}) = \frac{\text{fitness}(\text{bug})}{\sum_{\text{population}} \text{fitness}(i)} \quad (3)$$

Depending on the selection methodology, a “pressure” is applied that evolves the entire population toward an optimal fitness. Our analysis defines the fitness function to be of the form

$$\text{fitness} = \frac{1}{0.1 + \left( \sum_{n\text{Cases}} \sum_{n\text{Outputs}} (\text{Outputs} - \text{ExpectedOutputs}) \right)^4}, \quad (4)$$

where  $n\text{Cases}$  is the number of cases or characters in the training set for each generation and is set to 25,  $n\text{Outputs}$  is the number of possible outputs (the set of character numbers associated with each character) and is set to 5,  $\text{Outputs}$  is the NN determined value for the given character ranging from  $-1$  to  $+1$ , and  $\text{ExpectedOutputs}$  is the character output value the NN is expected to recognize either  $-1$  or  $+1$ . In other words, the fitness depends on the output residuals for all the characters in the training set for each generation.

Second, a single-point crossover methodology recombines genetic sequences to produce the next generation’s offspring. By randomly choosing a position within the parents’ genetic code of weights (i.e., its network), contiguous groups of weights are exchanged and/or swapped. In order to maintain diversity and successful candidates, three control parameters apply various degrees of evolutionary pressure : elitism, propagation (crossover probability), fitness scaling. One, elitists are bugs with the highest fitnesses. After each generation, eight neural nets with the highest fitnesses automatically become offspring in the next generation. By providing a “memory” or history of past optimal bugs to intermix with the current gene pool, elitism helps local search convergence rates at the expense of global perspective. Two, only a fraction of the parents are allowed to successfully mate, in this case 90%, further increasing the convergence pressure. Three, fitness scaling normalizes the population’s fitness distribution preventing early domination by extraordinary individuals, while encouraging evolutionary competition amongst near equals toward the end of convergence. The method used normalizes the spread of fitnesses about the population’s mean in relation to the population’s standard deviation of raw fitnesses.

Third, the mutation operator randomly replaces 10% of the genes (or weights) in the FFNN population with a random number ranging from -1.0 to 1.0. This allows the GA to continually search the entire phase space of the fitness function for its globally optimal value. All nets in the population are subject to mutation except the elitists.

When the entire population (64 nets) converges toward a fitness of 10.0 and an optimal solution is found, the design of the superbug is tested (or validated).

## **2.2 NN Validation Process**

The validation process randomly chooses a sequence of characters (those in figure 2), represents these characters as one-dimensional bit-vectors, measures how well the optimal neural network categorizes these characters, and plots the results.

The first two validation steps in figure 1 implement the same routines used in the training process except that 100,000 randomly sequenced characters are analyzed, thus each character in figure 2 plus a blank is sampled approximately 15,000 times, producing a statistical error less than 1%. The blank pattern is defined as a vector of -1's with an output sequence of all -1's. Most of the results omit those of the blank symbol due to its perfect recognition efficiency (i.e., it is uninteresting).

The optimal NN (i.e., the superbug) successfully categorizes a character when the output activation representing the input letter exceeds a +0.9 threshold while all other outputs remain below +0.9. Using this definition, character recognition efficiency is defined as the fraction of times an input character exceeds the threshold plus keeping all other characters outputs below this threshold. These efficiencies and the raw output activations ( $-1.0 < \text{Outputs} < +1.0$ ) of all the character outputs for each character input are tabulated then plotted.

## **3 Results**

### **3.1 Base Design**

The initial or base design is a 3 layer feed-forward neural network with 64 input nodes, 5 output nodes, and 17 hidden layer nodes. The GA implements (1) a roulette-wheel selection method, (2) a one-point crossover technique keeping 8 (of 64) nets as elitists and allowing 90% of the population to crossover, and (3) a flat or random mutation rate effecting 10% of the population's genes (i.e., weights). After each generation, the results from 25 randomly sequenced characters (i.e., the number of cases) determine a net's fitness. The GA searches for and converges toward the global maximum of this fitness function, defined by equation 4. These and other parameters realizing this design within the framework of the NND are tabulated in appendix A.

The neural network accurately identifies the 64-bit character representations of the training set. After running the program for 250 generations at about 1 second per generation (a total run-time of approximately 4 minutes) on a Sun SPARCstation 10 workstation (50 MHz CPU), the fitness converges to its optimal level of 10. The validation process indicates that the five test characters (A, B, C, D, and X) have been identified successfully. The results of the character recognition program are summarized in several informative graphics shown in figures 4 through 8.

### 3.1.1 Neural Network Connection Map

The architecture of the neural network is graphically represented using nodes for the neurodes and edges for the connections among the neurodes. As seen in figure 4, this results in two fully connected graphs. Every input node is connected to every node in the hidden layer, and every node in the hidden layer is connected to every output node. The colors of the edges correlate to the weights of each edge. The key to the weight-color correlation is presented on the lower right of figure 4.

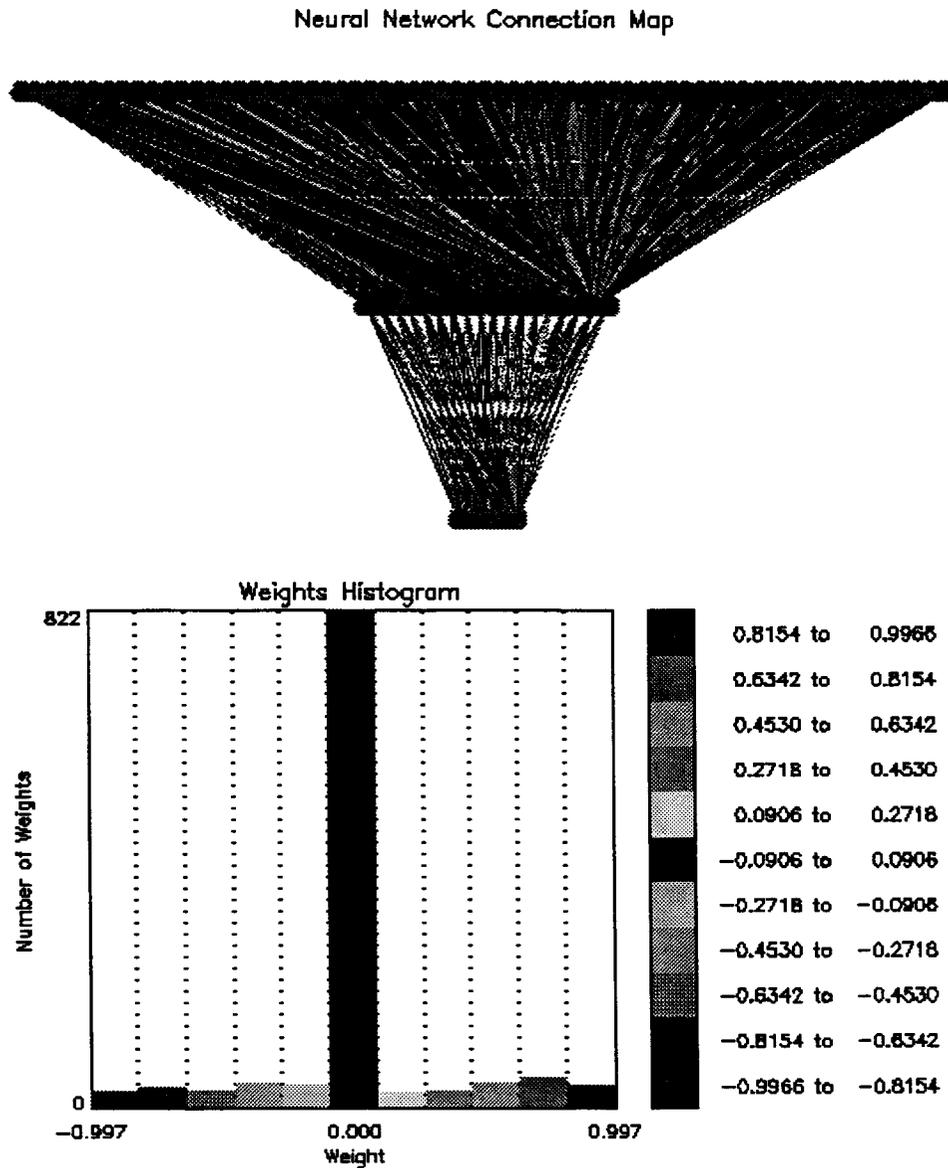


Figure 4. Trained neural network with weights designated by color.

### 3.1.2 Weights Histogram

The weights histogram on the lower left of figure 4 plots the number of times each weight is represented on the neural network connection map. The values of the weights range from -1 to 1.

### 3.1.3 Fitness

The four fitness plots in Figure offer information about the fitness of the neural network based on the fitness function. A perfect fitness is 10.0. The graph entitled “Raw Fitness” plots fitness as a function of each individual bug (64 in total) in the generation. The “Scaled Fitness” plots the (sigma-truncated) scaled fitness versus the individual bug. The “Best Raw Fitness” plots the fitness of the superbug versus the generation. The “Average Raw Fitness” plots the average fitness of the population versus the generation.

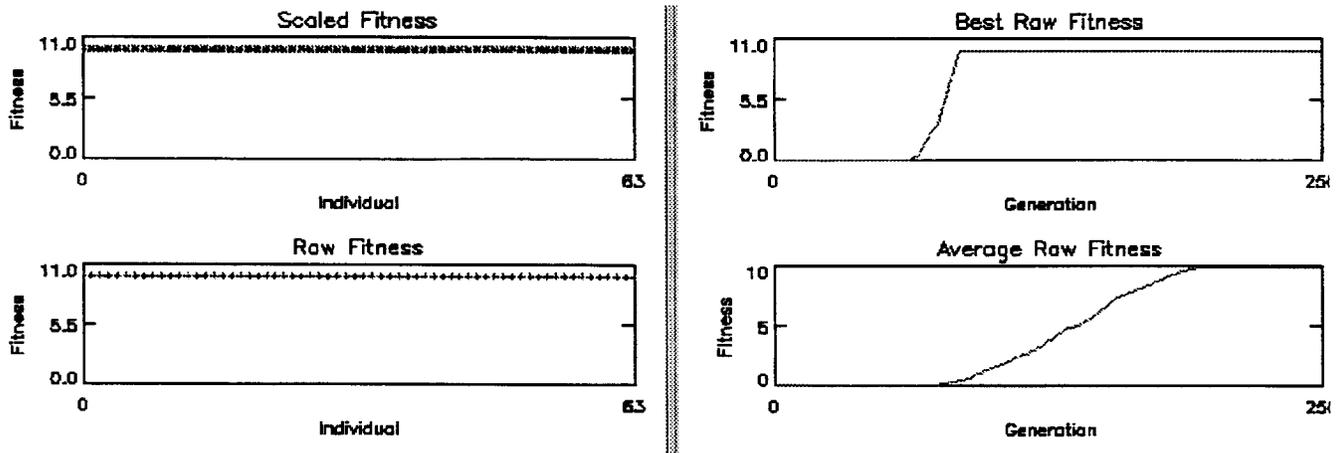


Figure 5. Fitness plots.

### 3.1.4 Brainscan

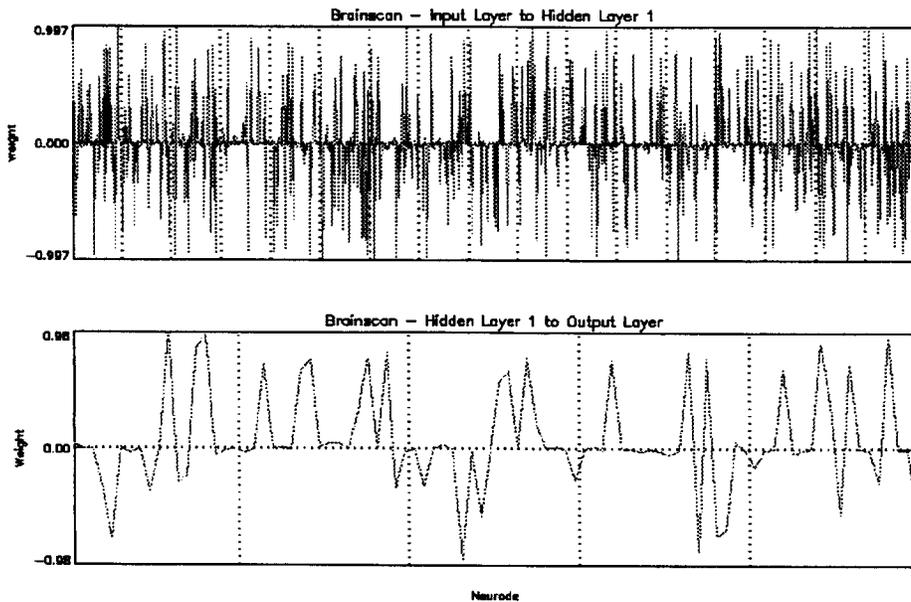


Figure 6. Brainscan of the NN displaying the distribution of weights.

The brainscan plots the weights of each connection between neurodes. The input layer to hidden layer 1 displays the weights of the connections between the input layer and the hidden layer, and the hidden layer 1 to output layer displays the weights of the connections between the hidden layer and the output layer. Such a plot reveals those input regions dominating (weights near +1), inhibiting (weights near -1), or unnecessary (weights near 0) to character recognition.

### 3.1.5 Character Recognition Results

The character recognition results in Figure plot the probability that the NN has successfully identified each letter. The red squares indicate the average outputs after recognition for each character. These probabilities provide useful information, but the NN measures its success based on the binary threshold model provided by the activation function. A probability of recognition about 0.9 is regarded as a perfect recognition. This allows a definitive recognition or lack thereof for each character, and it is indicated by the blue circles.

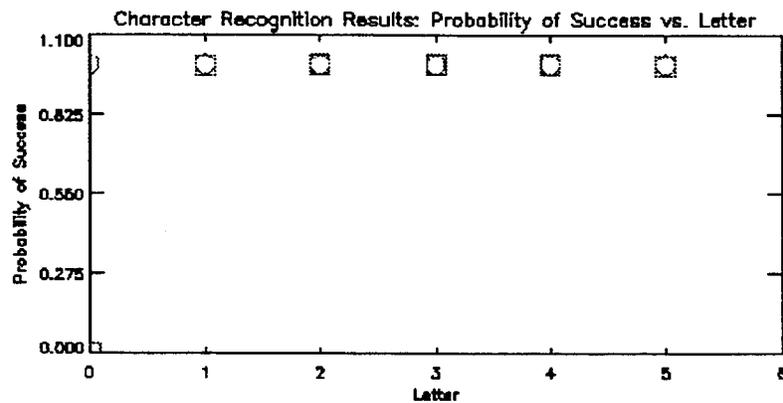


Figure 7. Probability that a letter will be correctly identified for letters in the initial character set.

### 3.1.6 Contour Array

As seen figure 8, the contour array plots the input character number versus the recognized output character number. This allows the user to identify any undesirable correlation between two separate characters. In a perfect recognition, the contour plots are evenly ranged around the  $y = x$  line. In an imperfect recognition, contour lines will appear in unexpected locations and alert the

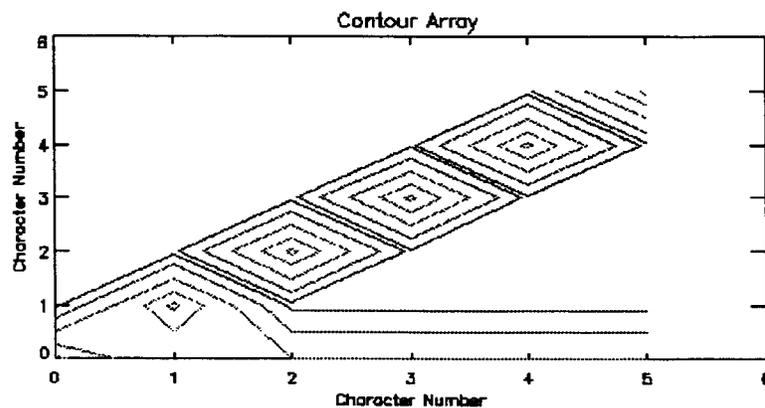


Figure 8. Contour plot depicting correlations between characters.

user to the NN's confusion. For example, if the neural net confuses a B and a P, then the contour array points (B, P) and (P, B) may have values that are not equal to -1. This would result in an island of contour lines appearing at those points, thereby indicating that the NN had confused the letters B and P.

<i>Training Set</i>	<i># of HL neurodes</i>	<i>Convergence</i>		<i>Validation Results</i>
		<i>Attempts</i>	<i>Successes</i>	<i>Successes</i>
A	1	3	3	3
A, B	2	3	3	2
	3	3	3	3
A, B, C	6	3	3	3
	2	3	1	0
	3	5	3	3
	4	3	3	1
A, B, C, D	5	3	3	3
	9	3	3	3
	4	7	3	1
	5	2	1	1
	7	8	3	2
	8	6	5	3
A, B, C, D, X	10	3	3	3
	5	4	0	-
	7	7	2	0
	9	7	2	2
	11	5	3	3
	13	5	3	3
	17	3	3	3

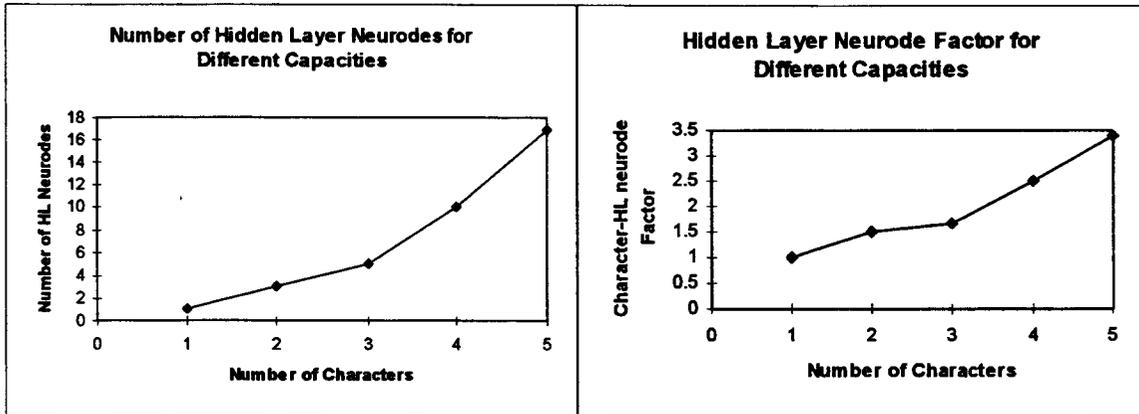
**Table 1. Tabulated results measuring the convergence and performance of the genetically-trained net as a function of character set size and number of hidden layer neurodes**

This collection of graphics allows a realistic evaluation of the capacity of the neural network. The graphics were developed in IDL and incorporated into the program through an interface.

### 3.2 Systematics

To understand the range of applicability of the current design, various key parameters influencing the network's ability to recognize characters are systematically modified. Three such variables include the number of hidden layer neurodes, the number of cases in the training set per generation, and noise level.

#### 3.2.1 The Number of Hidden Layer Neurodes



**Figure 9. The number of hidden layer neurodes and multiplicative factor (hidden neurodes/input characters) needed as a function of the number of characters.**

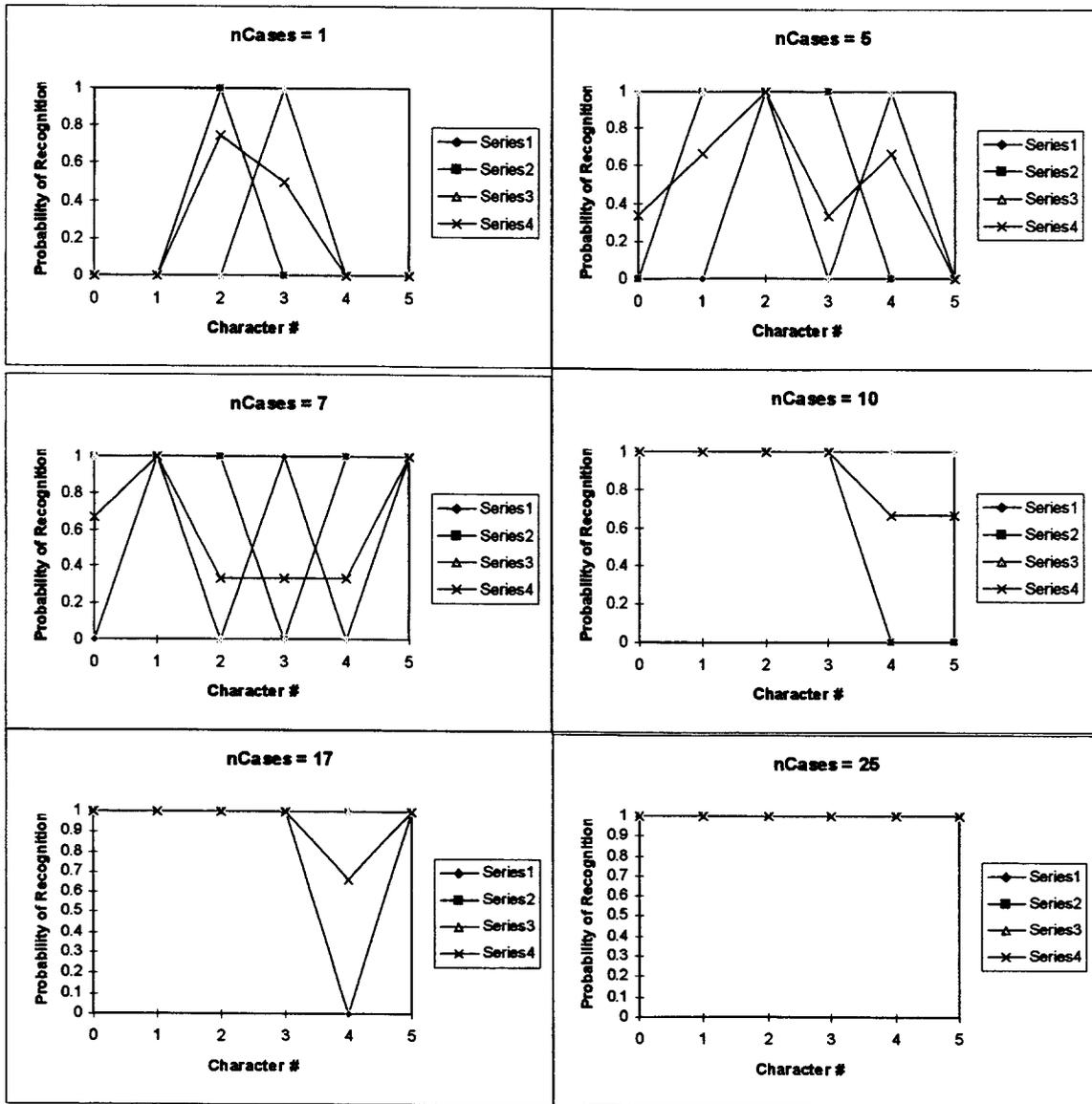
Tabulated in table 1 are the convergence and performance results for different character training set sizes (capacitance) and number of hidden layer neurodes. Columns 1 and 2 list the training and validation character sets and the number of hidden layer neurodes implemented in the net's design. Columns 3 and 4 record the attempts and convergence successes of the GA. Of those times the GA converged, the runs resulting in perfect validation results, 100% efficient character recognition, are shown in column 5. The shaded regions indicate perfect convergence and performance and are plotted in figure 9. These results reveal an increasingly nonlinear response of the hidden layer neurodes relative to the network's capacity.

### 3.2.2 The Number of Cases

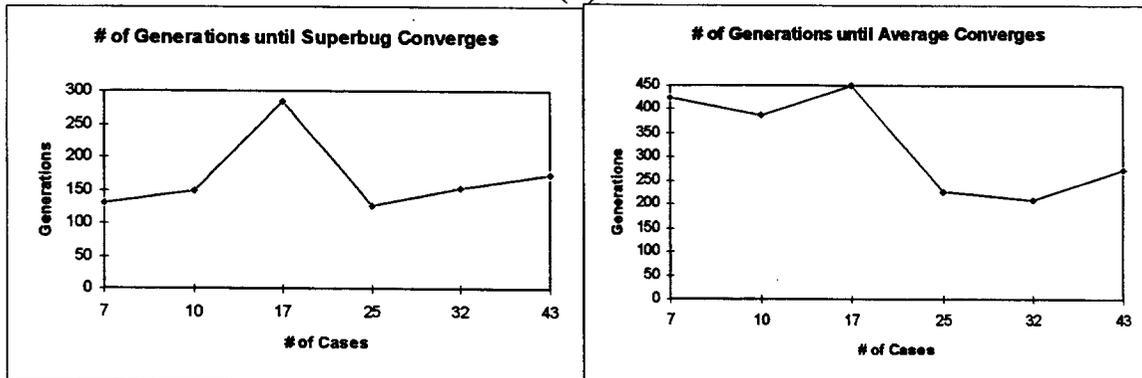
The top six graphs in figure 10 plot each character's recognition efficiency for three separate runs that converged (series 1, 2, and 3) and their average (series 4) for nCases, the number of cases parameter, equal to 1, 5, 7, 10, 17, and 25. When nCases ranges from 1 to 7, the genetically-trained network samples each character less than 1.4 times per generation. Even though the GA converged to an optimal state according to the fitness function, a value of 1.4 is insufficient for learning and the network's performance remains poor. As nCases increases somewhere between 10 and 17, performance (efficiency) and learning (convergence) greatly improves, with only one of the three sample runs unable to concurrently recognize all five characters. Once beyond a case number of 25, the network's training and performance is optimal.

The graphs in figure 10(B) plot the average number of generations over the three runs for the GA to search out a superbug and for the entire population to converge to this state. Notice that the population's first found optimal bug is insensitive (i.e., flat) to the number of cases, while the convergence of the entire population improves for larger case numbers.

### 3.2.3 Noise



(A)



(B)

Figure 10. (A) Character recognition efficiency of all five letters when trained with 1, 5, 7, 10, 17, and 25 number of cases. (B) The number of generations until the superbug and average of population converges.

Scanners will not reproduce a character's bitmap image without inefficiencies caused by such effects as smearing, character variations (as in hand-written characters), or differing fonts. All of these effects are considered noise. To begin to understand the effects of noise, a battery of tests measure the effects of noise during the validation process where noise is defined by randomly flipping bits. This is followed by incorporating noise into the training session in the hope of improving the net's overall performance through generalization. Therefore, there are two sources of noise : one during training and the other during validation.

As stated above, noise is equivalent to flipping bits within a character's bitmap image. Tests include training the neural network with character images that have (i) no bits flipped (BF), (ii) half the images with 0 BF and the other half with 1 BF, (iii) a third of the images with 0 BF, a third with 1 BF, and a third with 2 BF, and (iv) a fifth of the images with 0 BF, a fifth with 1 BF, and so on up to 4 BF. The second source of noise, which is implemented a little differently compared to the training set, is during the validation process. Validation results measure eight different character recognition efficiencies for those images with BF set only to 0, 1, 2, 3, 4, 6, 8, and 10, respectively. Results from this analysis are displayed in figures 11 and 12.

In figure 11, the efficiency for each character is plotted as a function of both validation and training BF. Note the final (bottom) graph. This graph plots the average for all five characters for each training method.

In figure 12, changes in the character recognition efficiency and average outputs relative to the zero BF validation results are shown.

Additional results not shown indicate that training the network with character images having 0 through 4 BF and with nCases set at 25 causes the GA to converge to a non-optimal set of efficiencies. After increasing the number of cases to 100, the GA converges consistently to an optimal value.

### **3.2.4 Form of the Activation Function**

Figure 13 displays the results for a net trained using 250 cases per generation, an activation function with the parameter set to 25 which approaches the form of a step function, and training images containing 0 through 4 randomly flipped bits. Unfortunately, the fraction of times the GA converges within  $10^4$  generations is small (1 out of 4) and has not been pursued until faster machines and parallel algorithms can be employed.

The left graph in figure 13 plots the character recognition efficiency as a function of noise. The symbols B and D are much improved and begin to drop off precipitously beyond 4 BF, the BF training set limit. Other letters are still much improved but fall off more gradually with noise.

The right graph in figure 13 plots the average character recognition efficiency over all five characters as a function of BF. Also plotted are the curves for the other four training techniques shown in figure 11. Tuning the activation function and increasing the number of cases show great improvement when overcoming a noisy environment.

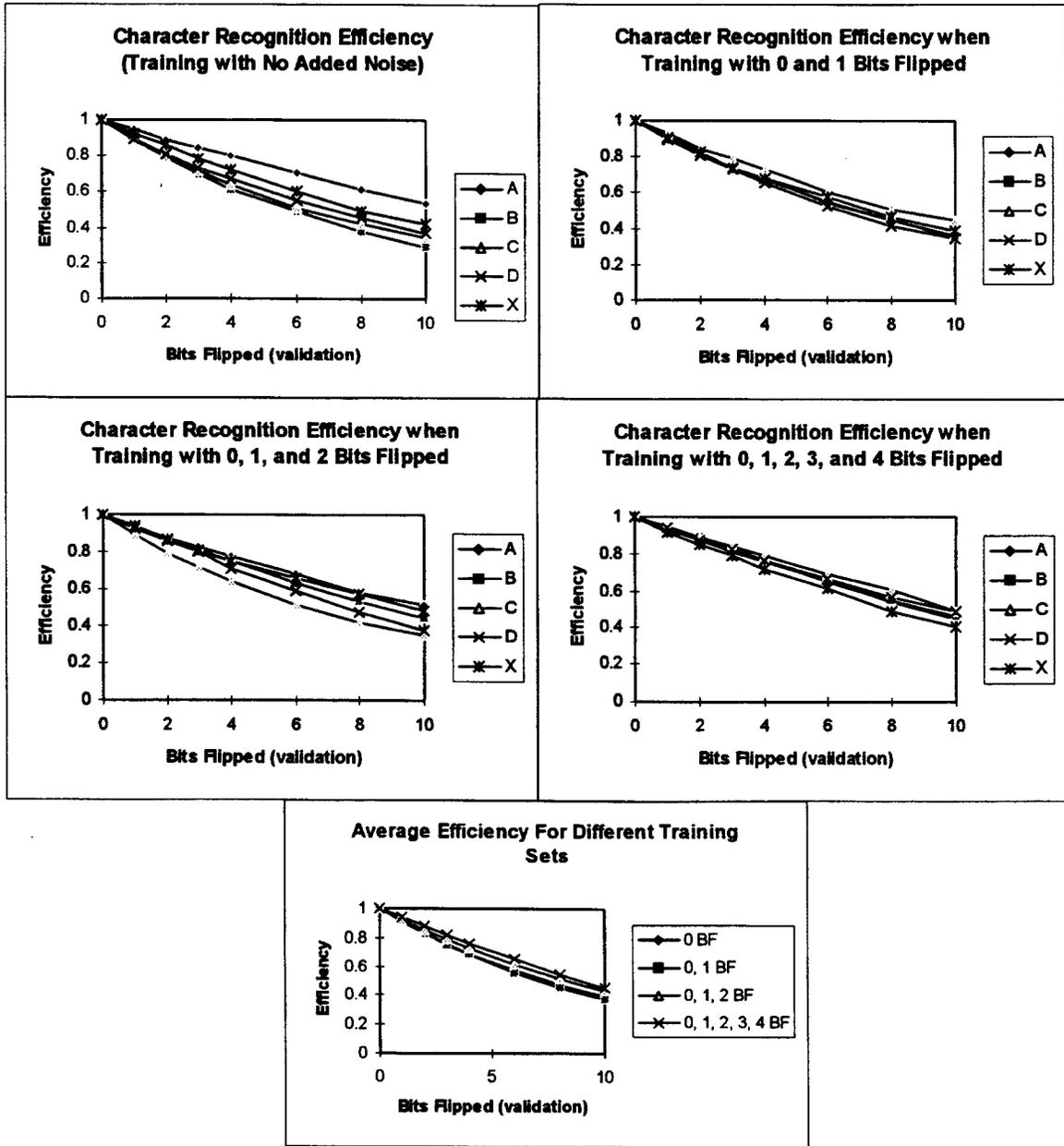


Figure 11. Top 4 plots: Character recognition efficiency of each character as measured with 0 through 10 bits flipped and trained with {0}, {0,1}, {0,1,2}, and {0,1,2,3,4} bits flipped. Bottom plot: Results when all 5 character efficiency results are averaged.

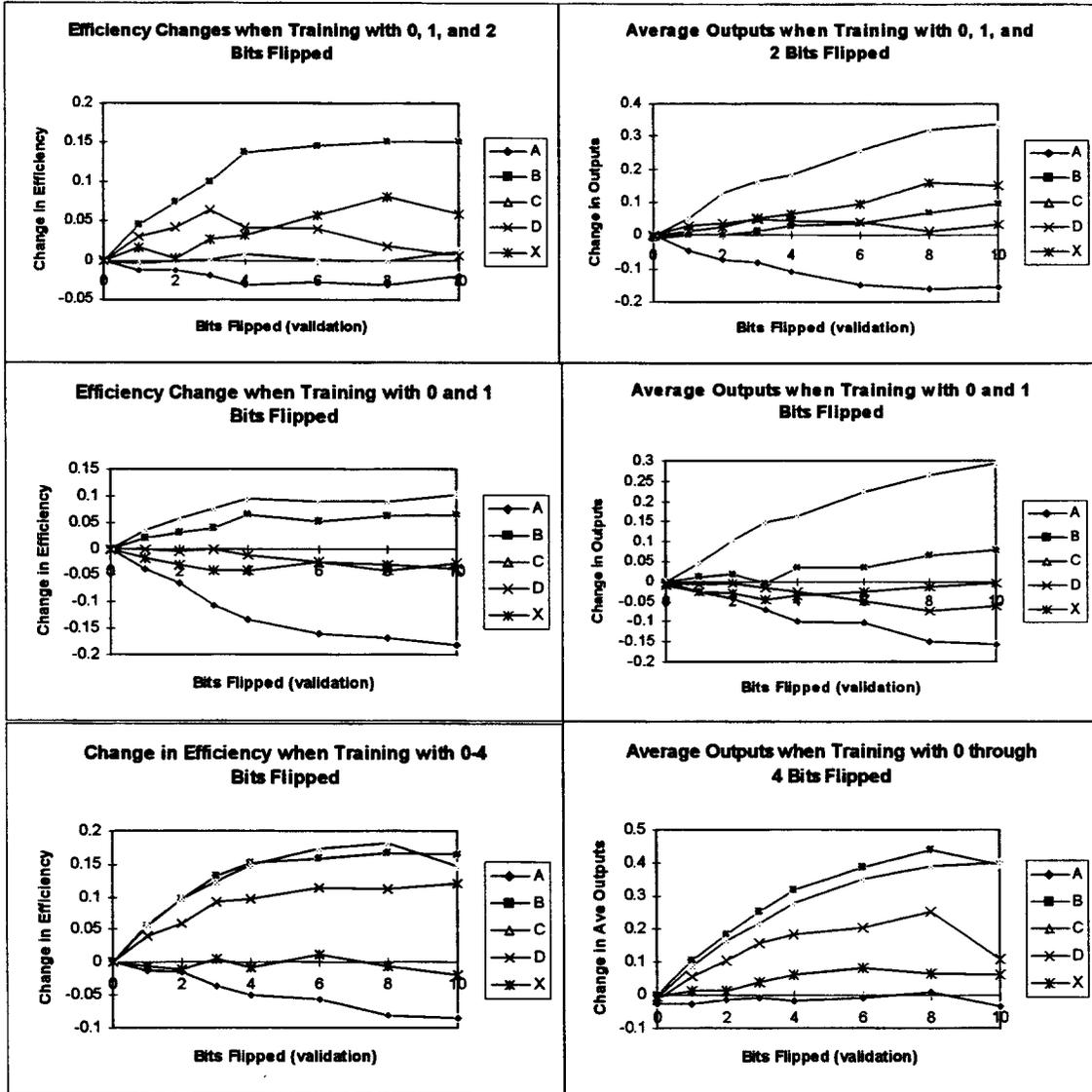
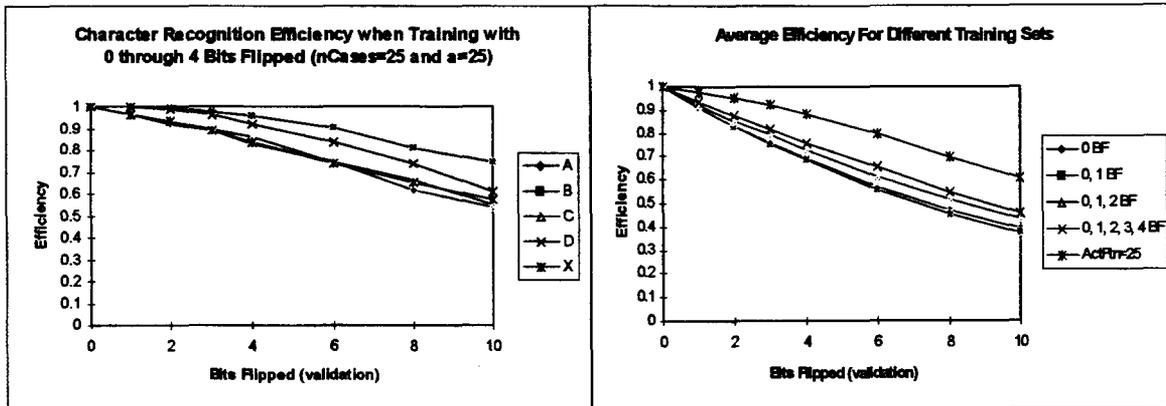


Figure 12. Change in character recognition efficiency (left) and average outputs (right) for different numbers of bits flipped during validation and different noisy training sets.



**Figure 13.** The left graph plots validation character reconstruction efficiencies for noisy image with BF from 0 to 10 when the NN is trained with noisy images (0 through 4 bits flipped), the number cases set at 250, and an activation function approaching a step function ( $y(x) = \tanh(25x)$ ). The bottom right graph plots the average efficiency against other training methods (see figure 11).

#### 4 Discussion

In its initial stage, the NN developed for character and symbol recognition has yielded promising results. If the NN is training properly, the fitness will converge to its optimal level. A perfect fitness is defined as having a value of 10.0. A convergence to the optimum fitness correlates with perfect character recognition. This is a strong indicator that an appropriate fitness function has been defined. The plot of the best raw fitness has a steeper slope than that of the average raw fitness because it plots the performance of the superbug of each generation. The average raw fitness indicates the speed (the number of generations) at which the population converges to a maximum. When the optimal fitness is attained, a set of ideal weights have been identified. The weights on individual nodes may differ from run to run, but consistent results of optimal fitness indicate that the NN is training properly. The optimal weights determine the multivariate solution: the superbug NN.

After training the NN, the program tests the success of recognition of the characters with a test set of data. The results from this test are plotted in the character recognition results. The character recognition results indicate perfect recognition of the characters in the test set approximately 90% of the time, in the base design. The runs in which recognition was imperfect also show that fitness did not converge. It is unclear whether the fitness would converge in these cases if the GA was allowed to proceed for additional generations.

As seen in both the Neural Network Connection Map and the Weights Histogram in figure 4, and in the Brainscan in Figure , there are many connections in the NN which are weighted at 0. This distribution suggests that the NN maybe capable of training for the information contained in a much larger character set. The solid recognition results of the initial set of five 64-bit characters demonstrates encouraging evidence that a single NN may be successfully trained to recognize multiple characters.

The contour plot seen figure 8 maintains an approximately regular distribution around the line  $y = x$ , thus indicating that the input of a given letter will result in the output of that letter. Moving away from the correlation points (A,A), (B,B), ..., (X,X) the contours show a fast descent from the 1 of correlation to the -1 of non-correlation. The contour plot has been effective in locating a NNs confusion when it either fails to recognize a character or it mistakes one character for another.

The number of hidden nodes has a direct effect on the capacity, convergence, and success of the net. A somewhat qualitative graph of the character capacity versus the number of hidden layer neurodes appears to be nonlinear (and in the wrong direction). In order to train a 3 layer feed-forward neural network, a prohibitively large number of hidden layer neurodes may be required (in contradiction to the large fraction of zero weights). However, the statistics are poor. More meaningful results requires a clear set of rules or thresholds to be introduced when deciding what is an appropriate convergence or success rate.

A successful search of the fitness function's optimal state depends on the number of cases, nCases. If increasing the number of times each character is seen by the genetically-trained net during each generation provides a more complete picture of the fitness function's space, performance is enhanced and near perfect character recognition efficiency is assured. A clue to the effect of nCases may be deduced from the plots in figure 10B. These plots indicate that the number of generations needed to search out an optimal solution is flat, while that of the entire population increases (fewer generations). One may surmise that the superbug search is dominantly stochastic while that of the entire population is dependent on evolutionary pressure. A larger number of cases may better represent (or define) the fitness function and its global optimum. However, execution speed times will increase requiring one to make a trade-off between nCases and performance.

Can the relatively abrupt change in net performance in relation to nCases be explained ?

First, let us assume that the input to the fitness function at each generation is a bit vector of length  $nInputs \cdot nCases$ . If the sequence of bits (or letters) is important, the population size of unique input patterns is equal to  $m^n$ , where  $m$  is the cardinality of the character set (nOutputs), and  $n$  is the number of cases (nCases). The lower bound on the number of (random) training samples providing an appropriate level of generalization for a FFNN with a single hidden layer is of the order [6]

$$\Omega\left(\frac{W}{\epsilon}\right), \quad (5)$$

where  $W$  is the number of weights and  $\epsilon$  is the acceptable error rate. With approximately  $10^3$  weights and a  $10^{-2}$  error rate, a sample group consisting of  $10^5$  elements is needed to draw from. Or,

$$nCases \approx \frac{\log(W/\epsilon)}{\log(m)} \approx \frac{5}{\log(5)} \approx 7.1. \quad (6)$$

Looking at the plots in figure 10A, a noticeable improvement in performance occurs between 7 and 10 cases, even though convergence rate remains low. It is not until the 17 to 25 range when the efficiency peaks (or flattens out). Instead of assuming all combinations of a As, b Bs, c Cs, d Ds, and x Xs to be distinct and equally distributed we should consider some combinations to be more probable than others (e.g., AABC occurs more often than BAAA). This effectively reduces  $m$  and increases the number of cases. Taking one step further, a random set of noisy characters may produce a much more complicated fitness function and or provide too large a training population. If the former, increasing the number of cases may be helpful or necessary.

Noise defined by the number of bits flipped bits (BF) can significantly effect character recognition efficiency – a reduction of ~30% when 4 bits are flipped (see figure 11). By training the net with character images consisting of up to 4 bits randomly flipped and with nCases at 100, a 6 to 7 percent improvement is seen. The plots in figure 12 reveal that each character performs differently. Improvements for the letters B, C, and D range from 11 to 17 percent. Part of the failures occurs when more than one output exceeds the 0.9 threshold. This effect can be inferred from the average outputs plots (right side of figure 12) which shows a significant change in the average outputs compared to the efficiencies. In addition, this effect is also corroborated by a few small islands off-axis in the contour plots (but not entirely).

Modifying the form of the activation function has proven beneficial. By increasing the parameter  $a$  from 5 to 25 in the sigmoidal function of equation 2, the activation approaches a step function. Combining this effect with an increase of nCases to 25 dramatically improves the character recognition efficiency – 20 percent over the base design. Therefore, optimizing the architectural design of the artificial neural network (e.g., the form of the activation functions) will be important.

A consequence of the improved training method is slower convergence and high CPU usage. Training with noisy characters requires 10× more generations to converge and 4× the number of cases to assure a global optimum. A parallel version of the NND becomes important.

## 5 Conclusions

Genetically training a 3 layer FFNN to recognize capital Roman letters of limited resolution has been shown to be successful. Basic rule-of-thumb calculations have been used to determine the architecture of the network and the parameters of the GA. Tests indicate increasing the capacity of the network in order to recall all 26 letters of the full alphabet would appear difficult unless additional character recognition techniques are incorporated. The number of cases plays a role in convergence rates but more importantly performance (i.e., character recognition efficiency). Attempts to quantify the minimum number of cases as a function of the neural network's architecture is instructive but somewhat limited. The major problem appears to be the ability of the network to overcome noise. Flipping bits within the character's bitmap image degrades recognition efficiency. Additional training with noisy characters regains some efficiency, but only a modest percentage – 4 bits flipped has a 30% loss while training with noise only gains 6 to 7% back. However, the greatest improvement comes from an activation function that approaches the form of a step function. On average, almost 70% percent of the base design losses are regained with a modified activation function. One can conclude that speed (a parallel version of the code) and improved techniques (architecture, training, character resolution) need to be explored.

Both the foreign language translation and the chemical formula classification applications require a higher degree of sensitivity than the  $8 \times 8$  representation can provide. To enhance the resolution of the characters and symbols in the training set of the NN, the bit map representation has been increased to 256 (a  $16 \times 16$  grid) and 1024 points (a  $32 \times 32$  grid), both successful at recognizing bitmap images of numbers and letters, respectively.

A graphical user interface (GUI) has been developed and incorporated into the Neural Net Designer to facilitate data entry as training sets for the NN increase. The GUI and relevant codes

that have been developed to convert a matrix representation to a bit array in C++ format offer the option of creating an  $8 \times 8$  grid, a  $16 \times 16$  grid, a  $32 \times 32$  grid, and a  $64 \times 64$  grid.

The resolution enhancement of the characters and symbols is an important issue for both the foreign language and the chemical classification applications. Visual assessment indicates that the  $32 \times 32$  grid will offer a level of resolution adequate for both applications. To utilize the GUI, the user simply clicks on boxes of the  $32 \times 32$  grid until the desired pattern is present. Then the bitmap is converted into an array of 1024 points and read into the Neural Net Designer software. This method will allow great flexibility of data input, and it will allow the program to be generalized for multiple character set databases used in various applications.

Current mainstream scanner and printer technology offers resolution of 600 dots per inch. This is approximately a factor of 2 greater than the  $32 \times 32$  grid representation, therefore appropriate scaling measures should be trivial to implement. It is reasonable to assume that the  $32 \times 32$  grid representation will be easily scaled from scanner input data.

The higher resolution offered by a 1024 point array will also provide a means by which the effect of noise in the data can be measured. The variances of font, orientation, or smudged paper will be minimized by the number of points in a 1024 point array. Noise introduction can be accomplished by randomly flipping bits. A systematic method for introducing random noise into the test set should be implemented for all of the characters during training of the NN with 1024 input nodes.

In order to meet the demands of a high level of resolution, it will be necessary to determine the effects of a large number of inputs on the program run time. The current run time for 64 input nodes is approximately 1 generation per second on a Sun SPARCstation 10 with fitness convergence in 250 generations. The scaling of the run time with respect to the inputs is unknown, but it is suspected that 1024 input nodes may create a prohibitively long run time. Preliminary efforts have indicated that a NN with 1024 input nodes and a 67 node hidden layer will take over two hours to evolve 250 generations. In addition, it does not seem probable that fitness convergence will be achievable in 250 generations given these parameters.

There are several methods by which this run time could be significantly decreased. Firstly, the bitmap representations of the characters are essentially sparse arrays. This indicates that methods of sparse matrix manipulation may be useful. In addition, parallelization of the processing would contribute to a more efficient run time. It should also be noted that a faster machine would also significantly impact the run time of the program.

As stated in the introduction, this paper represents a feasibility study on the effectiveness of using neural networks to recognize characters and symbols and to assess the amount of resources needed. Increasing the level of sophistication in our training methodology -- the number of cases and training with noise -- enhances character recognition performance at the cost of computational time. This effect has increased with higher character resolutions. By moving to faster machines (50MHz to 400+MHz), implementing a parallel version of the NND, and incorporating simulated annealing techniques into the GA, convergence rates and times can be improved somewhere between two and three orders-of-magnitude. It is important to note that these techniques are already in use in our other project areas. As for man/woman-power, the vast majority of the analysis has been accomplished by a college student through the internship program at SNL. During this time frame, approximately 3 weeks of her time was spent understanding C++, IDL, and the NND code and

another 3 weeks to perform the analysis. Therefore, genetically-trained neural networks appear to be a viable technique to use when recognizing characters (and symbols). Further progression should require minimal man/woman-power and leverage available resources and current methodologies.

## 6 Recommendations

The initial stages of character recognition using genetically trained neural networks have been successfully demonstrated. The development of a GUI significantly facilitates the input of new character or symbol sets, and a series of neural networks can be trained to recognize a vast library of alphabets and symbols. Additional areas for exploration in the area of character recognition are as follows:

- further investigate the correlation between the number of input nodes and the number of nodes in the hidden layer;
- investigate ways to improve convergence and/or run time of the program;
- investigate techniques to train with and overcome the effects of noise, such as font or smudged background;
- determine the robustness of the NN under conditions of translation, rotation, or dilation of the test character (possibly through a second hidden layer of neurodes); and
- incorporate the network's architectural parameters as part of the genetics.

Following the initial stage of character recognition, the NNs necessary for completion of the foreign language translation or chemical classification projects would diverge.

Foreign language translation would require NNs that could be trained to recognize grammar principles of the target language. In addition, they would need to be able to effectively translate the meaning of foreign words in the absence of a specific English counterpart. Eventually, the NN could be robust enough to perform with noisy (possibly smudged or damaged) data input, a variety of fonts or handwriting styles, and with languages that differ significantly from English in terms of grammar and/or structure. The size of the character inputs could also be a factor for the NNs to address.

It is anticipated that the large cardinality of the Chinese character set and the cursive nature of Arabic will present additional challenges; however, these challenges are representative of work with foreign character sets. The techniques developed to overcome these challenges would be a useful and significant contribution to the field since the problems of cardinality and connected characters occur in many applications.

NN recognition of superscripts and subscripts would be especially important in the recognition and classification of chemical formulas. This could be implemented with a two part process in which the first stage would classify a character as normal, superscript, or subscript, and the secondary stage would complete the identification. Chemical formula classification would also require an expansive library of chemical nomenclature.

## References

- [1] Caudill, Maureen. "Using Neural Networks." AI Expert, June 1992.
- [2] Goldberg, David E. Genetic Algorithms in Search, Optimization & Machine Learning. Addison-Wesley Publishing Company, Inc.: Reading, Massachusetts, 1989.
- [3] Hertz, John, Anders Krogh, and Richard Palmer. Introduction to the Theory of Neural Computing. Santa Fe Institute Studies in the Sciences of Complexity. Addison-Wesley Publishing Company, Inc.: Reading, Massachusetts, 1991.
- [4] CIA/DOE Partnership Program Proposal for FY99 (Sandia National Laboratories Proposal), 1998.
- [5] Proposed Foreign Language Translation (Sandia National Laboratories Internal Proposal), 1998.
- [6] Baum, E. B. and Haussler, David. "What Size Net Give Valid Generalization ?", Neural Computation 1, p151-160, 1998

## Appendix A

The super neural structure for character and symbol recognition was developed using Neural Net Designer software created at Sandia National Laboratories. The Neural Net Designer includes parameters for both the GA and the NN. The values of each parameter used in the character recognition initialization file for the basis design are indicated in italics.

### PARAMETERS FOR THE GA

#### Crossover Method

This allows the designer to choose the method of recombination for the genetic material of the bugs during reproduction. The options are even/odd, *one point*, random, two point, and uniform.

#### Crossover Probability

This determines the probability that crossover will occur. It is generally between 0.8 and 1.0. *Character recognition NN value: 0.9.*

#### Fitness Scaling Method

This determines the method by which the fitness is scaled. The options are linear, no scaling, power law, and *sigma truncation*. In the final output, both the raw fitness data and the scaled fitness data are presented.

#### GA Type

This determines the type of genetic algorithm to be used during training. The options are deme, incremental, simple, and *steady-state*.

#### Mate Selection Method

This determines the method by which mates are selected during reproduction. The options are deterministic sampling (DS), rank, *roulette wheel*, stochastic remainder sampling, tournament, and uniform.

#### Mutation Method

This determines the method by which the bug's genetic material is mutated. The options are creep, geometric creep, *random*, replacement, and swap.

#### Mutation Probability

This is the probability that mutation will occur. Generally, it does not exceed 0.2. *Character recognition NN value: 0.05.*

#### Number Of Generations

This determines time that the GA is allowed to run. The number of generations must be sufficient for the fitness optimization. However, after the fitness attains its optimal level, additional generations add unnecessary run time to the program. *Character recognition NN value: 250.*

#### Optimization Mode

This dictates whether the GA will attempt to *maximize* or minimize its solution.

#### Population Size

This determines the number of bugs to be included in each generation. *Character recognition NN value: 64.*

### PARAMETERS FOR THE NN

#### Number of Hidden Layers

This determines the number of hidden layers in the NN architecture. Although the Neural Net Designer is capable of including up to four hidden layers, this number rarely exceeds one. *Character recognition NN value: 1.*

#### Number of Neurodes

This determines the number of neurodes,  $n$ , present in the hidden layer. The correlation between  $n$  and the number of inputs/outputs has not been extensively investigated. However, the literature suggests that  $n$  should be approximately the square root of the number of inputs multiplied by the number of outputs. In addition,  $n$  must be relatively prime to the number of neurodes in the input and/or output layers, therefore  $n$  should be prime in order to maintain the robustness of the program. *Character recognition NN value: 17.*

#### Activation Function

This determines the activation function used to provide a threshold above which the neurode “fires.” This function can be defined as constant function, Gaussian function, step function, or *sigmoid function* (either logistic or *hyperbolic tangent*). Each of these functions contains three parameters. As seen in figure 3, the Neural Net Designer provides the equation of the activation function in terms of the parameters,  $a$ ,  $b$ , and  $c$ . These parameters are listed as Neurode Activation FunctionParameter1 ( $a$ ), Neurode Activation Function Parameter2 ( $b$ ), and Neurode Activation Function Parameter3 ( $c$ ). *Parameter1 is set to 5.0, Parameter2 is to 0.0, and Parameter3 is set to 0.0.*

## Endnotes

---

<sup>1</sup> CIA/DOE Partnership Program Proposal for FY99

<sup>2</sup> CIA/DOE Partnership Program Proposal for FY99

<sup>3</sup> "Using Neural Networks" (Maureen Caudill), p. 8.

<sup>4</sup> Introduction to the Theory of Neural Computing (John Hertz, Anders Krogh, and Richard Palmer), p. 3.

<sup>5</sup> Introduction to the Theory of Neural Computing (John Hertz, Anders Krogh, and Richard Palmer), p. 10

<sup>6</sup> Genetic Algorithms in Search, Optimization, & Machine Learning (David Goldberg), p v.

<sup>7</sup> Genetic Algorithms in Search, Optimization, & Machine Learning (David Goldberg), p. 7.

External Distribution:

- 1 Cecelia Diniz  
C-1269  
101 N. Merion Ave.  
Bryn Mawr, PA 19010

Internal Distribution:

- 1 MS 1188 Stewart Cameron, 9512
- 1 MS 1188 Perry Gray, 9512
- 1 MS 1188 Roy Hamil, 9512
- 1 MS 1188 Eric Parker, 9512
- 10 MS 1188 Keith Stantz, 9512
- 1 MS 1188 Michael Trahan, 9512
- 1 MS 1207 Eleanor Walther, 5909
- 1 MS 1188 John Wagner, 9512
- 1 MS 1188 Christine Wehlburg, 9512
- 1 MS 9018 Central Technical Files, 8940-2
- 2 MS 0899 Technical Library, 4916
- 2 MS 0619 Review & Approval Desk, 15102  
For DOE/OSTI