Department of Electrical Engineering
College of Engineering and Applied Sciences
State University of New York at Stony Brook
Stony Brook, New York 11794-2350

Artificial Neural Networks for the Diagnosis of Coronary Artery Disease

by

K. Wendy Tang and Girish Pingle

Technical Report # 718

October 6, 1995

Artificial Neural Networks for the Diagnosis of Coronary Artery Disease

K. Wendy Tang and Girish Pingle [1]
Department of Electrical Engineering
SUNY at Stony Brook, Stony Brook, NY 11794-2350.

ABSTRACT

Artificial Neural Networks (ANNs) have been widely advocated as tools for solving many decision modeling problems. In this paper, we use ANNs for the prediction of coronary artery disease. Real data from four major international medical organizations are used in the training and testing of the ANN algorithm. To speed up the training time, we implemented the algorithm in parallel on an Intel Paragon parallel computer. We have achieved an accuracy of $> 78\%$*, a comparable performance to probabilistic and statistical techniques. Furthermore, due to parallel implementation, we achieve the accuracy in* < 3 *minutes of training time. Comparing with statistical approach, such savings in time is substantial. We, therefore, conclude that ANN is a fast alternative to classical statistical techniques for prediction and modeling of experimental data.*

Keywords:Delta-Bar-Delta learning, RPROP learning, Pattern partitioning, Proben Database.

1 Introduction

Recently *artificial neural networks* (ANNs) have been widely advocated as tools for solving many decision modeling problems. Basically, most ANNs can be considered as non-linear, *non-parametric* regression [1, 2] techniques. Contrary to parametric regression in which rigid assumptions are made about the model structure, artificial neural networks, being non-parametric, make no assumption about the distribution of the data and are thus capable of "letting the data speak for itself'. Consequently, artificial neural networks are robust and are powerful tools for pattern recognition or classification of experimental data.

[1]This research was supported by the National Science Foundation under Grant No. ECS-9407363. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation.

The authors also like to thank Professors James Glimm and Yuefan Deng in the Center for Scientific Computing of the Applied Mathematics Department in SUNY Stony Brook for the use of the Intel Paragon parallel computer.

Inspired by the structure of the human brain, ANNs are *massively parallel systems* that rely on dense interconnection of simple processing elements, or neurons. The most dominant form of ANNs is the multi-layer feedforward network. It is a hierarchical design consisting of fully interconnected layers of neurons [3]. Learning is accomplished through a *back-propagation* algorithm. Although these ANNs have been proved to be universal function approximator [4, 5], the backpropagation learning algorithm is very slow. Various strategies have been proposed to speed up the learning process.

In this paper, we investigate the use of backpropagation ANNs for the diagnosis of coronary disease based on a patient's personal data such as *age, gender, smoking habits; subjective patient pain descriptions*; and results of various medical examinations such as *blood pressure and electro cardiogram* results. The ANN predicts if any one of the four major heart vessels are reduced in diameter by more than 50% and thus increase the patient's potential of a heart attack.

Real data collected from four major international medical organizations, namely the *Cleveland Clinic Foundation, Hungarian Institute of Cardiology, V.A. Medical Center Long Beach* and *the University Hospitals in Zurich and Basel, Switzerland* are used in training and testing of the algorithms. Each database has the same format. There are 76 raw attributes in each database but only 14 are used in our ANN algorithm. Since these are real data, most of the attributes have missing values. These missing values can be interpreted as "noisy" data. Among the four database, the one from the Cleveland Foundation is the "least noisy" as it has only two missing attributes overall. These databases have been used by other researchers to develop probabilistic models for the prediction of coronary disease. For example, Detrano *et al.* [6] have developed a probabilistic model for these data. In their case, they achieved approximately a 77% classification accuracy. Using some clustering technique, Gennari *et al.* [7] has achieved a 78.9% accuracy on the Cleveland database.

In this paper, we use backpropagation ANNs for prediction. To speed up the learning process, we use two popular algorithms, the *Delta-Bar-Delta* [8] and *RPROP* [9]. We found that both algorithms provide a prediction accuracy of about 80%. To further expedite the training process, we implement the algorithms in an Intel Paragon computer using 10 or 20 processors. With such parallel implementation, we are able to achieve training in < 3 minutes. Comparing with the probabilistic model [6] and the clustering technique [7], our parallel implementation of ANN provide comparable accuracy in a much shorter time. We believe, therefore, that ANNs, especially in parallel implementation, is a fast alternative to classical statistical techniques in prediction and modeling of experimental data.

Database	Class 0	Class 1	Class 2	Class 3	Class 4	Total
Cleveland	164	55	36	35	13	303
Hungarian	188	37	26	28	15	294
Switzerland	8	48	32	30	5	123
Long Beach VA	51	56	41	42	10	200
Total Database	411	196	135	135	43	920

Table 1: *The Distribution of the Heart Database.*

The report is organized as follows: in Section 2 we provide a description of the database being considered here. Section 3 is a review of the ANN algorithm, including the equations governing the basic algorithm and the *Delta-Bar-Delta* [8] and *RPROP* algorithms for faster convergence. Parallel implementation on an Intel Paragon parallel computer is discussed in Section 4. Preliminary results are presented in Section 5. Finally, in Section 6, some concluding remarks are included.

2 The Heart Disease Database

The heart database consists of real, experimental data from four international medical organizations. It is part of a set of benchmark problems called *Proben1*[2] for ANN learning. The name of the four major organizations and the person responsible for the data collections are: Andras Janosi, M.D., in the Hungarian Institute of Cardiology, Budapest; William Steinbrunn, M.D., in the University Hospital at Zurich, Switzerland; Matthias Pfisterer, M.D. in the University Hospital at Basel, Switzerland; and Robert Detrano, M.D., Ph.D., at the V.A. Medical Center, Long Beach and the Cleveland Clinic Foundation.

The database has 76 raw attributes but only 14 of them are actually used in the ANN algorithm. A detailed description of these attributes are found in a file along with the database. For the reader's convenience, the 14 attributes used in ANN learning are summarized:

1. Age: age in years.

2. Gender: (1 = male; 0 = female)

[2]The database is available through the internet via ftp to the Neural Bench archive at Carnegie Mello University with internet address "ftp.cs.cmu.edu", directory "/afs/cs/project/connect/bench/contrib/prechelt"

3. Chest pain type:
   - Value 1: typical angina;
   - Value 2: atypical angina;
   - Value 3: non-anginal pain; and
   - Value 4: asymptomatic.

4. Resting blood pressure (in mm Hg on admission to the hospital).

5. Serum cholesterol in mg/dl.

6. Fasting blood sugar > 120 mg/dl; (1 = true; 0 = false).

7. Resting electrocardiographic results:
   - Value 0: normal
   - Value 1: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV)
   - Value 2: showing probable or definite left ventricular hypertrophy by Estes' criteria

8. Maximum heart rate achieved.

9. Exercise induced angina (1 = yes; 0 = no).

10. ST depression induced by exercise relative to rest.

11. The slope of the peak exercise ST segment
    - Value 1: upsloping
    - Value 2: flat
    - Value 3: downsloping

12. Number of major vessels (0-3) colored by flourosopy

13. thallium: 3 = normal; 6 = fixed defect; 7 = reversable defect

14. num: diagnosis of heart disease (angiographic disease status)
    - Value 0: < 50 % diameter narrowing
    - Value 1: > 50 % diameter narrowing

Attributes 1-13 are inputs to the artificial neural network algorithm and attribute 14 corresponds to the desired diagnosis for the patient. As will be discussed in Section 3, the ANN is to learn this input-output pattern for the database.

A detailed description of these attributes are found in [6, 10]. Since these are real data, most of the attributes have missing values. These missing values can be interpreted as "noise" of the data. Among the four database, the one from the Cleveland Foundation is the

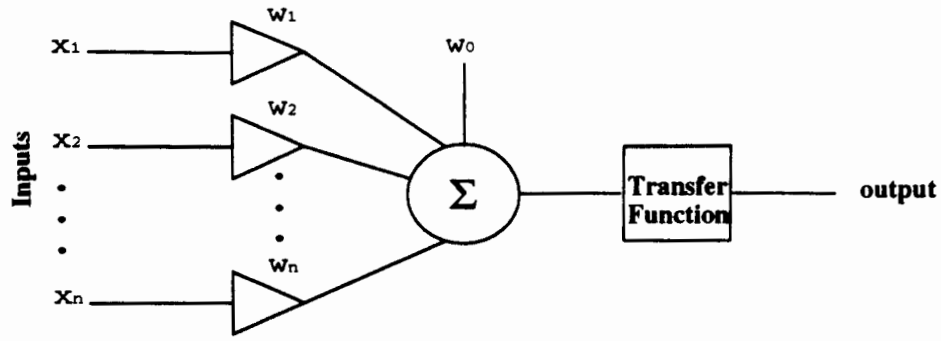

Figure 1: *A Mathematical Model of a Neuron.*

"least noisy" as it has only two missing attributes overall. For the four organizations, there are a total of 920 sets of data, each set with 14 attributes. Table 1 shows the distribution of these data among the four institution. Class 0 corresponds to patients with *no* coronary artery reduction; Class 1 to 4 for having the diameter of *one, two, three, four* coronary arteries reduced by more than 50%.

These databases have been used by other researchers to develop probabilistic models for the prediction of coronary disease. For example, Detrano *et al.* [6] has developed a probabilistic model for these data. In their case, they achieved approximately a 77% classification accuracy. Using some clustering technique, Gennari *et al.* [7] has achieved a 78.9% accuracy on the Cleveland database. In Section 5 we show that ANN prediction provides a slightly improved accuracy in substantially less amount of time. Through parallel implementation, we have achieved a prediction accuracy of 78.57% to 85.45% in < 3 minutes for the entire database; and an accuracy of 78.57% to 92.86% in < 0.5 minutes for the Cleveland database.

3 Artificial Neural Networks

Artificial Neural Networks (ANNs) are *massively parallel systems* that rely on dense interconnection of simple processing elements, or neurons. Each neuron has n inputs either from an external source or from other neurons. The weighted sum of these n values is transmitted to a non-linear function, the *transfer function*, to produce the output of a neuron. A diagram of the mathematical model of a neuron is shown in Figure 1 [11].

Neural networks offer several advantages over conventional computing architectures [12]. Calculations are carried out in parallel yielding speed advantages and programming is done

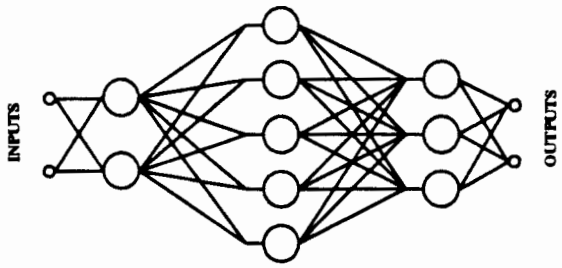

Figure 2: *A Three Hidden-Layer Backpropagation ANN Network.*

by training through examples. These networks are characterized by their *learning* and *generalization* capabilities and can be deployed as "black boxes" that map inputs to outputs with no explicit rules or analytic function [13]. The neural network "learns" the system model by training through a set of desired input-output patterns.

The most dominant forms of ANNs are the multi-layer backpropagation [14, 15]. Basically, it is a hierarchical design consisting of fully interconnected layers of neurons [3]. The weights associated with each neuron are updated by taking the gradient of the total squared error with respect to the weights and performing a gradient search of the weight space [16]. Errors are propagated backwards through the network, hence the name *back-propagation.* Figure 2 shows a backpropagation network with three hidden layers.

Although these multilayer feedforward ANNs have been proved to be universal function approximator, the backpropagation learning algorithm is very slow. Various strategies, such as the *Delta-Bar-Delta* rule [8] and the *RPROP* algorithm [9], have been proposed to speed up the learning process.

Figure 3 shows a flow chart of the training process. Basically, training is accomplished through iterations. For each iteration, there is a feedforward calculation of the outputs of each neuron from the current weight space. As a results of such feedforward calculation, error for all training samples are calculated. Such error are fedback to the network to calculate the gradient of the error with respect to the weight space. Once the feedback calculation is completed, weights are updated. To speed up the convergence rate of the backpropagation algorithm, these weights are often updated with some speed-up algorithm, such as the *Delta-Bar-Delta* and the *RPROP* algorithm.

In the following equations, we present the equations governing the backpropagation ANN algorithm and the *Delta-Bar-Delta* and the *RPROP* algorithms to speed up the convergence rate.

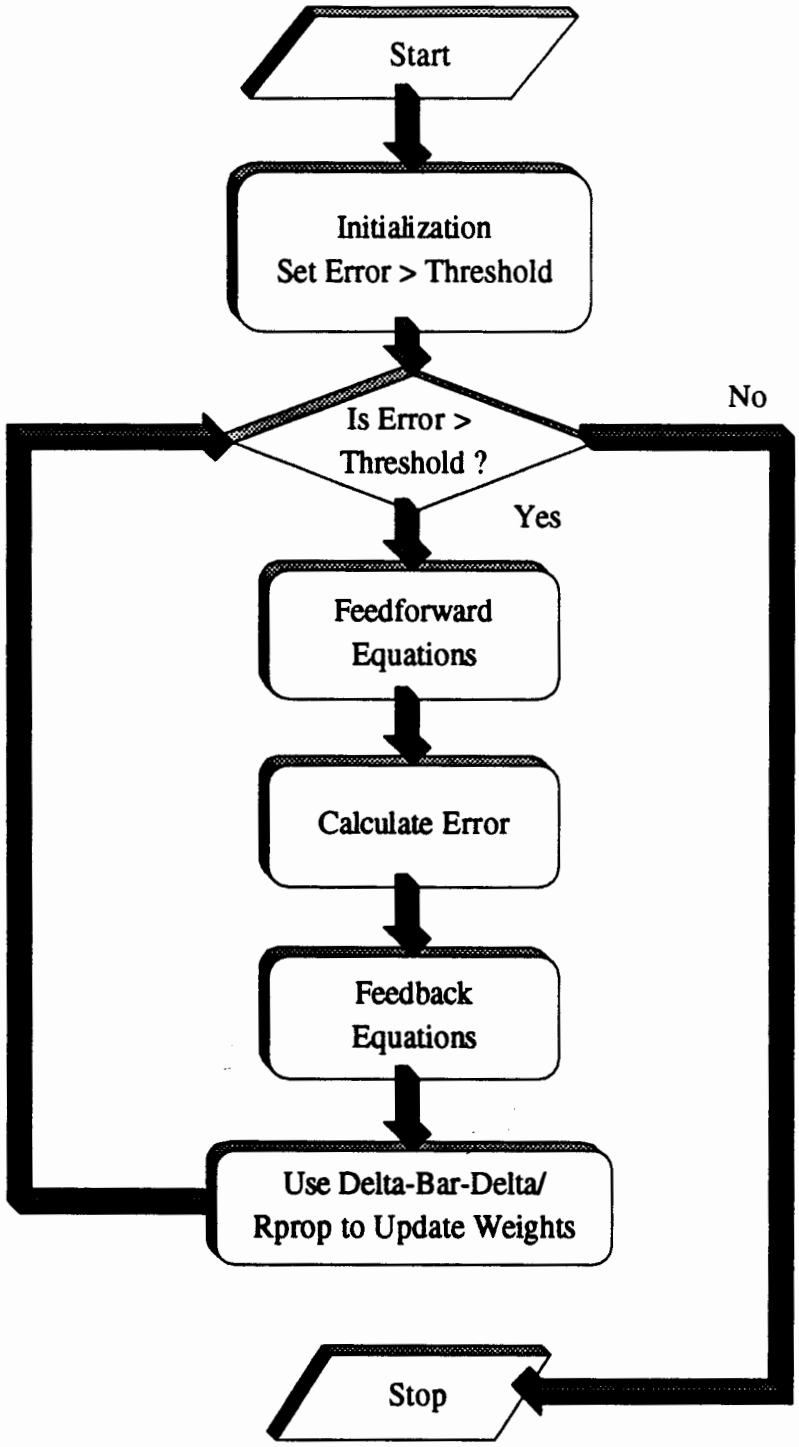

Figure 3: *A Flow-Chart for Training an ANN.*

3.1 Basic Backpropagation Algorithm

In this section, we review the basic equations for the basic backpropagation algorithm. For expository convenience, we assume there are m inputs, n outputs, H hidden nodes, and T training samples. The training samples are those that have the desired output values known. The training inputs are presented to the network as $X_i(t), \quad i = 1, \ldots, m, \quad t = 1, \ldots, T$ and the corresponding desired outputs are $Y_i(t), \quad i = 1, \ldots, n, \quad t = 1, \ldots, T$. The detailed of the algorithm can be found in [17]. For the reader's convenience, they are also summarized here.

The network equations consist of the feedforward and feedback components. During the feedforward mode, the network calculates an estimated output $\hat{Y}$ as a function of the inputs and the weights associated with the neurons. An error function is then produced by comparing $\hat{Y}$ with the desired output Y. In the feedback mode, the gradients of this error with respect to the weight space are identified. Subsequently, the weights are updated through the steepest descent method. More specifically, for training samples, $t = 1, \ldots, T$, the feedforward equations are:

$$x_i(t) = X_i(t) \qquad 1 \leq i \leq m \tag{1}$$

$$\text{for } i = m+1 \text{ to } i = m+H+n,$$

$$\begin{cases} net_i(t) &= \sum_{j=1}^{i-1} W_{ij} x_j(t) \\ x_i(t) &= s(net_i(t)) \end{cases} \tag{2}$$

$$\hat{Y}_i(t) = x_{m+H+i}(t) \qquad 1 \leq i \leq n \tag{3}$$

The error of the network is obtained by comparing the actual and the desired outputs.

$$E = \sum_{t=1}^{T} E(t) = \sum_{t=1}^{T} \sum_{i=1}^{n} 0.5[\hat{Y}_i(t) - Y_i(t)]^2 \tag{4}$$

where $\hat{Y}_i(t)$ is the output of the neural network and $Y_i(t)$ is the desired outputs. This error is fed back to the network. The error gradient F_W_{ij} with respect to each weight, W_{ij} is calculated with the feedback equations:

For training samples, $t = 1, \ldots, T$, the feedback equations are:

$$F\_\hat{Y}_i(t) = \frac{\partial E}{\partial \hat{Y}_i(t)} = \hat{Y}_i(t) - Y_i(t) \qquad i = 1, \ldots, n \tag{5}$$

for $i = m + H + n$ to $i = m + 1$,

$$\begin{cases} F\_x_i(t) & = & F\_\hat{Y}_{i-m-H}(t) + \sum_{j=i+1}^{m+H+n} W_{ji} * F\_net_j(t) \\ F\_net_i(t) & = & s'(net_i) * F\_x_i(t) \end{cases} \tag{6}$$

$$F\_W_{ij} = \sum_{t=1}^{T} F\_net_i(t) * x_j(t) \qquad i, j = 1, \ldots, m + H + n \tag{7}$$

where $s(z)$ is the sigmoidal transfer function and $s'(z)$ is the derivative of $s(z)$. Also,

$$s(z) = 1/(1 + e^{-z}) \tag{8}$$

$$s'(z) = s(z) * (1 - s(z)) \tag{9}$$

Once F_W_{ij} (the gradient of E with respect to W_{ij}) is calculated, each weight is updated according to:

$$\text{New } W_{ij} = W_{ij} - \alpha * F\_W_{ij} \quad i, j = 1, \ldots, m + H + n \tag{10}$$

where α is the *learning rate.*

3.2 Delta-Bar-Delta Rule

To improve the convergence speed of the steepest descent/ascent method, R. Jacobs proposed the delta-bar-delta algorithm [8]. Basically, the algorithm is a special case of the *Adaptive Learning Rate* (ALR) discussed in [18]. Every weight of the network is given its own learning rate and that the rate changes with time. According to [8], the learning rate update rule is:

$$\Delta\alpha_{ij}(t) = \begin{cases} \kappa & \text{if } \bar{\delta}_{ij}(t-1)\delta_{ij}(t) > 0 \\ -\phi\alpha_{ij}(t-1) & \text{if } \bar{\delta}_{ij}(t-1)\delta_{ij}(t) < 0 \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

where

$$\begin{aligned} \delta_{ij}(t) &= F\_W_{ij} \\ \bar{\delta}_{ij}(t) &= (1-\theta)\delta_{ij}(t) + \theta\bar{\delta}_{ij}(t-1) \\ \alpha_{ij}(t) &= \alpha_{ij}(t-1) + \Delta\alpha_{ij}(t) \end{aligned}$$

In these equations, $\delta_{ij}(t)$ is the partial derivative of the error with respect to W_{ij} at time t and $\bar{\delta}_{ij}(t)$ is an exponential average of the current and past derivatives with θ as the base and time as the exponent[8]. If the current derivative of a weight and the exponential average of the weight's previous derivatives possess the same sign, the learning rate for that weight is incremented by a constant κ. If the current derivative of a weight and the exponential average of the weight's previous derivatives possess opposite signs, the learning rate for the weight is decremented by a proportion ϕ of its current value [8].

3.3 RPROP Learning Rule

RPROP (*Resilient PROPagation*) [9] is another weight-updating scheme which performs a direct adaptation of the weight based on local gradient information. Its main difference with the *Delta-Bar-Delta* learning scheme is that only the sign of the partial derivative of the weight is used for weight adaptation. This essentially eliminates the harmful influence of the actual size of the derivative caused in other algorithms. Its another advantage is that there are very few initial parameters to be set compared to the Delta-Bar-Delta rule. (Only the initial Weight update value Δ_{ij} and the magnification/reduction factors for the update-value, namely, η^+ and η^- are required to be set .)

For each weight, its individual update-value Δ_{ij} is introduced , which solely determines the size of the weight-update. This adaptive update-value evolves as the training progresses, and depends on the current and previous partial derivative of the weight and the previous update-value.

The learning rule is given as follows. See [9] for details.

$$\Delta_{ij}^{(t)} = \begin{cases} \eta^+ \Delta_{ij}^{(t-1)}, & \text{if} \frac{\partial E}{\partial w_{ij}}^{(t-1)} \times \frac{\partial E}{\partial w_{ij}}^{(t)} > 0 \\ \eta^- \Delta_{ij}^{(t-1)}, & \text{if} \frac{\partial E}{\partial w_{ij}}^{(t-1)} \times \frac{\partial E}{\partial w_{ij}}^{(t)} < 0 \\ \Delta_{ij}^{(t-1)}, & \text{otherwise.} \end{cases} \tag{12}$$

In other words, whenever the derivative changes its sign, it indicates that the previous update value was too large. In this case, the update value is decreased by a factor η^-. If the derivative retains the sign, the update value is increased by a factor η^+ to speedup the training process.

Once the update-value for each weight is adapted , the weights are updated by a simple rule. If the derivative is positive (increasing error), the weight is decreased by its update valued. if the derivative is negative, the update value is added to the current weight.

$$\Delta w_{ij}^{(t)} = \begin{cases} -\Delta_{ij}^{(t)}, & \text{if} \frac{\partial E}{\partial w_{ij}}^{(t)} > 0 \\ +\Delta_{ij}^{(t)}, & \text{if} \frac{\partial E}{\partial w_{ij}}^{(t)} < 0 \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

And,

$$w_{ij}^{t+1} = w_{ij}^{t} + \Delta {w_{ij}}^{t} \tag{14}$$

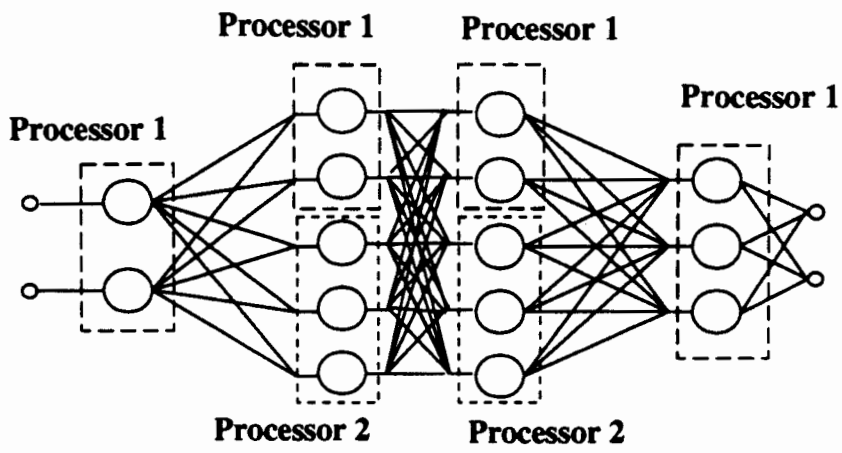

Figure 4: *Node-Partitioning of ANN Network.*

4 Parallel Implementation

To exploit the inherent parallel nature of ANN algorithms, we implement the backpropagation algorithm with the Delta-Bar-Delta the RPROP rules in an Intel Paragon parallel computer with 110 nodes. The ANN algorithms can be parallelized by two techniques - *node partitioning* and *pattern partitioning* [19].

Basically, node-partitioning implies that the entire network is partitioned among different processors, each computing for the whole set of training samples. Pattern-partitioning, on the other hand, partitions the training patterns among the processors with each one representing the entire network. In the next subsection, we include a brief description of node-partitioning. Section 4.2 describes pattern partitioning.

4.1 Node-Partitioning

In node partitioning, nodes of the entire network are partitioned among different processors. Our strategy is to divide the number of hidden nodes in each hidden layer equally among the given number of processors. Because of the small number of nodes in the input and output layers, these layers are assigned to a specific processor. Figure 4 shows conceptually, how a 2-5-5-3 ANN are partitioned among two processors. For simplicity, we have each processor keep a copy of all weights of the network. In the forward loop, each processor calculates node activations $x_i(t)$ in parallel and broadcasts it to other processors for computation of next layer activations.

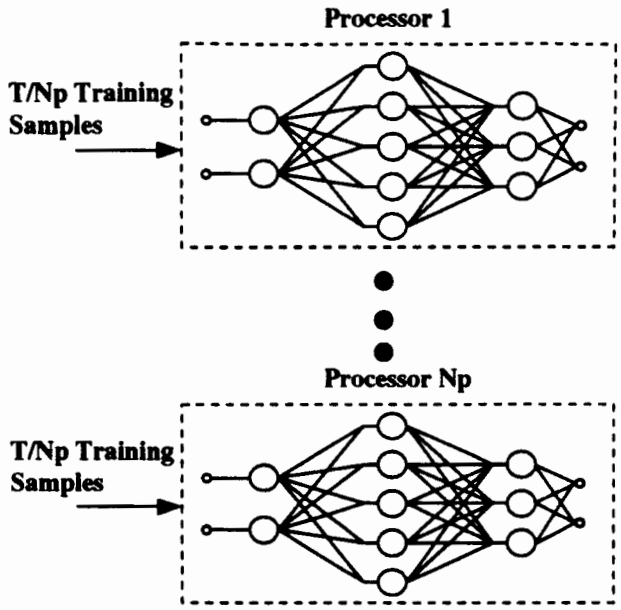

Figure 5: *Pattern-Partitioning of ANN Network.*

In the backward loop, each processor computes node errors and weight changes required for its incoming weights. Node errors are broadcasted to other processors for computation of previous layer errors. Each processor also computes total weight change required for all its incoming weights. At the end of backward loop, each processor broadcasts the weight changes calculated to update the other processor's buffer. (Each processor keeps a copy of all the weights in the network.) After this, Delta-bar-Delta or the RPROP rule is used to calculate new weights. Our preliminary investigation found that node-partition helps to reduce execution time only for large networks. In our application, a relatively small ANN (about 10 or 20 nodes in each of the two hidden layers) is sufficient. For such small networks, the communication overhead involved in parallel implementation actually slows down the overall execution time. We, therefore, implemented only the pattern-partitioning scheme for the learning of the heart database.

4.2 Pattern-Partitioning

In our implementation of the pattern partitioning scheme, training samples are equally divided among the number of processors. That is, for T training sets, and N_p number of processors, each processor computes both the feedforward and the feedback components of the $T_p = \frac{T}{N_p}$ training samples. (We assume that N_p divides T). Figure 5 shows conceptu-

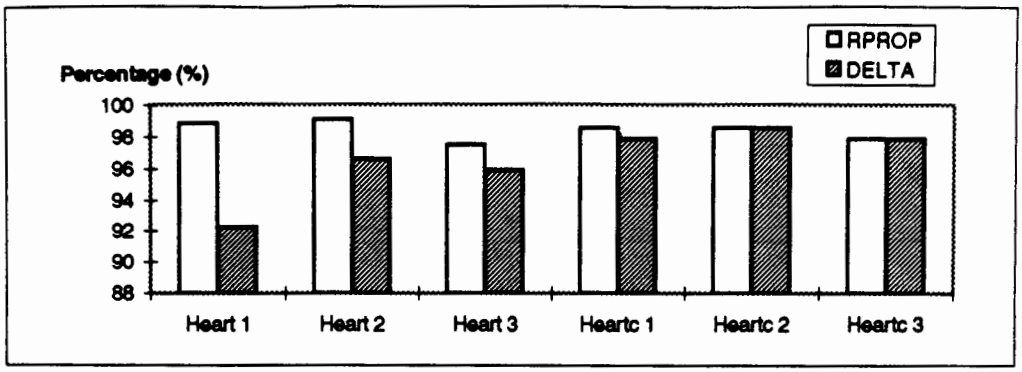

Figure 6: *Percentage of Correct Classification for Training Data.*

ally that in pattern-partitioning, each of the N_p processors contains the entire copy of the network but training only with T_p training samples.

At the end of the backward loop, the weight changes computed based on the subset of the training samples of each processor are broadcasted. Once this information is received by all processors, the total weight changes F_W_{ij} are computed at every processor:

$$F\_W_{ij} = \sum_{k=1}^{N_p} F\_W_{ij}(k)$$

where $F\_W_{ij}(k)$ (Equation 7) is the weight gradient of processor k computed based on its own subset of training samples. Upon obtaining the total weight gradient, delta-bar-delta or the RPROP rule is applied at all processors to update the weights which completes one iteration.

5 Preliminary Results

In Section 2 we have presented the heart data base collected from real data and in Sections 3 and 4 we have outlined the concept of ANN and its parallel implementation. In this section, we explain how an ANN is used to apply to the heart database to predict whether a patient's major coronary arteries have diameter reduction of 50% or more.

As explained in Section 3, ANNs learn by training with a number of training data. During the training process the input-output relation of the training samples are presented to the

network and the algorithm learned by adapting its actual output to the desired output. After the training process is completed, the ANN will have learned the input-output pattern of the application. To test how well the ANN algorithm predicts the outcome, a different set of data, called the *testing set*, is often presented to the network, the output of the feedforward calculation of the ANN algorithm is then the predicted output. This predicted output is used to compare with the desired output which has not been presented to the ANN. The accuracy in the testing set indicates how well the network has learned the underlying pattern of the application.

In other words, the dataset used in ANN prediction must be divided into at least *two* sets, the *training set* and the *testing set.* The training set is used in adapting the weight of the network and the testing set is used to find out how the trained network performs with a new set of data. In many cases, however, the training set is further divided into the *actual training* set and a *validation set.* The latter is used as an intermediate test to avoid *overfitting.* The *overfitting* phenomenon occurs when a network learns also the specific noise associated with the training set and will therefore unable to provide accurate prediction to a new testing data. Readers interested in more discussions on the overfitting issue are referred to [20].

The heart database described in Section 2 has a total of 920 input-output data. We divide the database into *three* sets, the *actual training* set with 460 data; the *validation* set with 230 data; and the *testing* set with 230 data. The *actual training* set is used in adapting the weights of the ANN. For every five iterations, we use the *validation* set to test the predicted outcome. If the predicted outcome is worse, then the weights adapted during the last five iterations is abandoned. When the training process is completed, we present the *testing* set to the ANN. The accuracy of the output of the ANN is then calculated.

Because the basic principle of backpropagation ANN is to adapt the weight to fit the training set, the composition of the training set should therefore consists of a sufficient representation of the input-output relations of the application. To ensure the group of data in the *training, validation* and *testing* set is a mixed of all the data, it is suggested in [10] that the original heart database be permuted differently to three databases, called *Heart1, Heart2, Heart3.* These three database are identical to the original one except the order of the individual data is different. These permuted datafiles are also part of the Proben1 database [10].

Besides the original heart database, the Proben1 database also consists of a separate datafile for data from the Cleveland Foundation. These data are the least "noisy" of all four

sets because there are only two missing attributes. Like the original heart database, there are also three different permutations of the Cleveland Foundation database. They are called *Heartc1, Heartc2, Heartc3.*

In each of the *Heart1, Heart2, Heart3* data file, there are 460 sets of actual training data, 230 validation data, and 230 testing data. In our parallel implementation, we use pattern partitioning with 20 processors each handling $460/20 = 23$ sets of training samples. For files containing data from the Cleveland Foundation only (*Heartc1, Heartc2, Heartc3*), there are 303 data set. Since we are using pattern-partitioning for parallel implementation, the number of processors must divide the number of patterns, we use 10 processors with 140 data for actual training, 70 for validation, and 70 for testing. Once the ANN is trained, the prediction accuracy is calculated using the testing set which is never shown to the network for training.

Figures 6 and 7 are the percentage accuracy of the training and testing for the six data files (Heart1,2,3 and Heartc1,2,3). We have used both the Delta-Bar-Delta rule and Rprop algorithm to speed up the convergence. We found that both algorithms provide comparable performance in accuracy. For the Heart1,2,3 datafiles (the combined data set), the testing accuracy is between 78.64% to 85.45%. For the Heartc1,2,3 datafiles (data from Cleveland Foundation only), the accuracy is between 78.57% to 92.86%. Recall that the same database has been used by two other groups with probabilistic and classical statistical techniques. In those cases, the classification accuracy is 77% for the combined datafile [6] and 78.9% for data from the Cleveland Foundation [7]. Our results using ANNs are therefore slightly better in accuracy.

However, the time needed to achieve these results are significantly less than the classical statistical methods. Instead of performing an indepth analysis of the different parameters, in ANNs, learning is achieved through training by repeatedly presenting the pattern to the network. With parallel implementation (pattern-partitioning), we are able to accomplish training in < 3 minutes for the combined database using 20 processors on the Paragon computer (Figure 8). For the Cleveland data file, it takes < 0.5 minutes for training using 10 processors on the Paragon computer (Figure 9). The substantially less amount of time is due to the less noisy data and less number of data (In the Cleveland database, there are only 2 missing values; and total number of actual training data set is 140 as opposed to 460 in the combined database.)

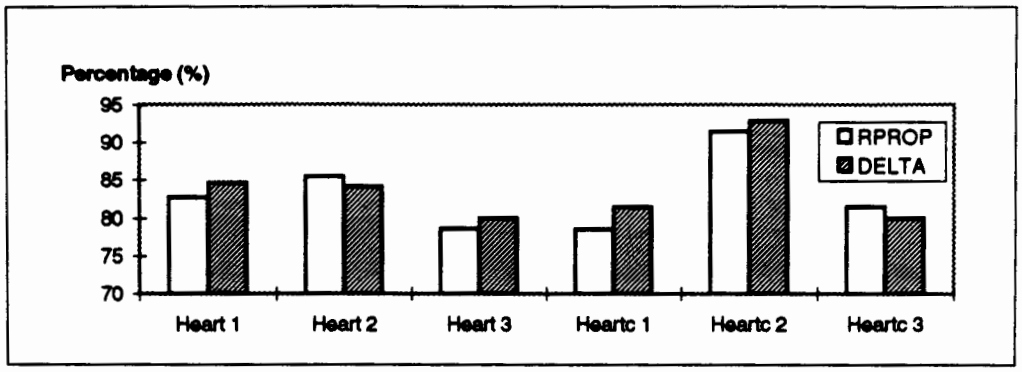

Figure 7: *Percentage of Correct Classification for Testing Data.*

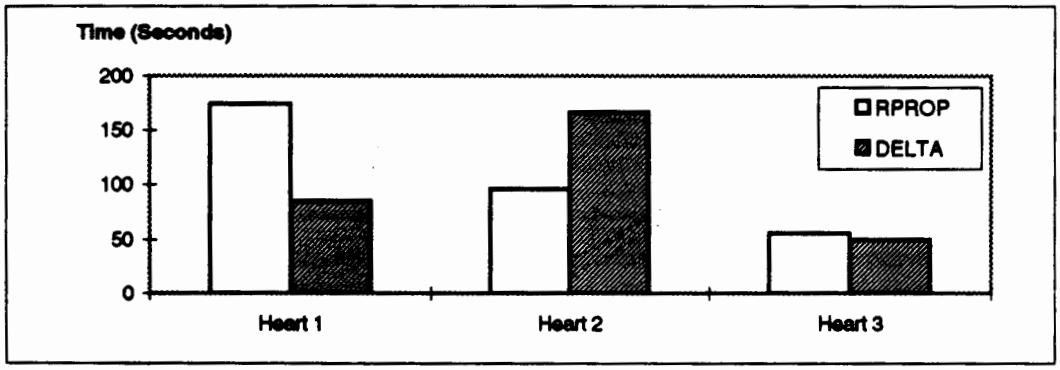

Figure 8: *CPU Time for Training of Heart Data.*

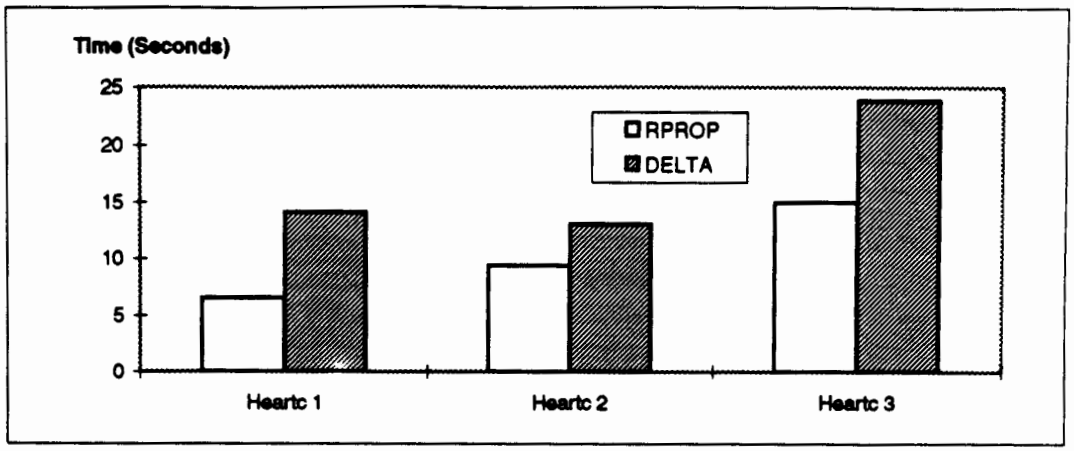

Figure 9: *CPU Time for Training of Heartc (Cleveland) Data.*

6 Concluding Remarks

In this paper, we investigate the use of backpropagation ANNs for the diagnosis of coronary disease based on a patient's personal data such as *age, gender, smoking habits; subjective patient pain descriptions*; and results of various medical examinations such as *blood pressure and electro cardiogram* results. The ANN predicts if any one of the four major heart vessels are reduced in diameter by more than 50% and thus increase the patient's potential of a heart attack.

Real data collected from four major international medical organizations, namely the *Cleveland Clinic Foundation, Hungarian Institute of Cardiology, V.A. Medical Center Long Beach* and *the University Hospital in Zurich and Basel, Switzerland* are used in training and testing of the algorithms. Previous research efforts have used these database to test prediction algorithm using probabilistic and statistical analysis. Classification accuracy for these efforts is 77% for the combined database [6] and 78.9% for the Cleveland Foundation data [7].

We use multilayered backpropagation ANN for prediction. To speed up the training time, we implemented two popular algorithms, the *Delta-Bar-Delta* and the *Rprop* algorithm. We found that the two speed-up routines provide similar performance. To further reduce the training time, we use a parallel computer, the Intel Paragon computer, with pattern-partitioning implementation.

We have tested the algorithm on two kinds of datafile, one is the combined datafile from all four medical organizations; and the second one consists of data from the Cleveland Foundation only. For the combined data file, we obtained a classification accuracy of between 78.57% to 85.45%. For the Cleveland data base, our classification accuracy is between 78.57% to 92.86%.

Comparing these results with earlier results from probabilistic and statistical techniques, our accuracy is slightly better. However, for the computation time, the ANN algorithm shows a stronger advantage over classical statistical techniques. Exploiting parallel computation, we are able to achieve training in < 3 minutes for the combined data file and < 0.5 minutes for the Cleveland database. We, therefore, conclude that ANN prediction is potentially a fast alternative to statistical techniques for applications that involves large amount of data, such as the prediction of the coronary artery disease.

References

[1] S. Geman, E. Beienenstocks, and R. Doursat. "Neural Networks and the Bias/Variance Dilemma". *Neural Computation*, pages 1–58, 1992.

[2] W.S. Sarle. "Neural Networks and Statistical Models". In *Proceedings of the 19th Annual SAS Users Group International Conference*, pages 1–13, April 1994.

[3] D. Rumelhart and J. McClelland. *Parallel Distributed Processing*, volume 1. MIT Press, Cambridge, MA, 1986.

[4] K. Hornik and M. Stinchcombe. "Multilayer Feedforward Networks are Universal Approximators". *Neural Networks*, 2:359–366, 1989.

[5] K.-I. Funahashi. "On the Approximate Realization of Continuous Mapping by Neural Networks". *Neural Networks*, 2(3):183–192, 1989.

[6] R. Detrano et al. "International Application of a New Probability Algorithm for the Diagnosis of Coronary Artery Disease". *American Journal of Cardiology*, 64:304–310, 1989.

[7] J.H. Gennari, P. Langley, and D. Fisher. "Models of Incremental Concept Formation". *Artificial Intelligence*, 40:11–61, 1989.

[8] Robert A. Jacobs. "Increased Rates of Convergence Through Learning Rate Adaptation". *Neural Networks*, 1:295–307, 1988.

[9] M. Riedmiller and H. Braun. "A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm". In *Proceedings of the IEEE International Conference on Neural Networks*, San Francisco, CA, April 1993.

[10] Lutz Prechelt. PROBEN1 - A Set of Neural Network Benchmark Problems and Benchmarking Rules. Technical Report Technical Report 21/94, Universitat Karlsruhe, Karlsruhe, Germany, September 1994.

[11] S.Y. Kung. *Digital Neural Networks.* Prentice Hall, Inc, Englewood Cliffs, New Jersey, 1993.

[12] Richard L. Lippmann. "An Introduction To Computing With Neural Nets". *IEEE ASSP Magazine*, pages 4–22, April 1987.

[13] Judith E. Dayhoff. *Neural Network Architectures.* Van Nostrand Reinhold, New York, 1990.

[14] W. Horne, M. Jamshidi, and N. Vadiee. "Neural Networks in Robotics: A Survey". *IEEE Journal of Intelligent and Robotic Systems*, 3:51–66, 1990.

[15] K.S. Narendra and K. Parthasarathy. "Identification and Control of Dynamical Systems Using Neural Networks". *IEEE Transactions on Neural Networks*, 1(1):4–27, March 1990.

[16] B. Widrow and M.A. Lehr. "30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation". *Proceedings of the IEEE*, 78(9):1415–1442, September 1990.

[17] Paul J. Werbos. "Backpropagation Through Time: What It Does and How to Do It". *Proceedings of the IEEE*, 78(10):1550–1560, October 1990.

[18] Paul J. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences.* PhD thesis, Harvard University, Cambridge, MA, November 1974.

[19] V. Kumar, S. Shekhar, and M.B. Amin. "A Scalable Parallel Formulation of the Backpropagation Algorithm for Hypercubes and Related Architecture". *IEEE Transactions on Parallel and Distributed Systems*, 5(10):1073–1089, October 1994.

[20] N.A. Gershenfeld and A.S. Weigend. The Future of Time Series: Learning and Understanding. In A.S. Weigend and N.A. Gershenfeld, editors, *Time Series Prediction: Forecasting the Future and Understanding the Past*, pages 1–70. Addison-Weseley, 1994.