Artificial Neural Networks for the Diagnosis of Coronary Artery Disease

K. Wendy Tang, Girish Pingle and Govardhan Srikant

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 > 76%, a comparable performance to probabilistic and statistical techniques. Furthermore, with parallel implementation, we achieve the accuracy in < 5 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.

Two popular weight-adaptation algorithms, RPROP and Delta-Bar-Delta rules are compared. The effect of network architecture and how to treat missing values for these two algorithms are also investigated. In general, RPROP is more robust and less affected by choice of architecture, order of data presentation, and effect of missing values.

Keywords: Delta-Bar-Delta learning, RPROP learning, Pattern partitioning, Proben1 Database, Pattern Recognition, Artificial Neural Networks.

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 [GBD92, Sar94] 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

\footnote{This research was supported by the National Science Foundation under Grant No. ECS-9407363.}
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.

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 [RM86]. Learning is accomplished through a backpropagation algorithm. Although these ANNs have been proved to be universal function approximator [HS89, Fun89], the backpropagation learning algorithm is very slow. Various strategies have been proposed to speed up the learning process [Jac88, RB93].

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; subjective patient pain descriptions; and results of various medical examinations such as blood pressure and electrocardiogram 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 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. [D+89] 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. [GLF89] 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 [Jac88] and RPROP [RB93]. To further expedite the training process, we implement the algorithms on an Intel Paragon computer using 5 processors. With such parallel implementation, we are able to achieve training in < 5 minutes. Comparing with the probabilistic model [D+89] and the clustering technique [GLF89], 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 experi-
mental data. Design issues such as the choice of network architecture and how the missing values are handled are also investigated.

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 \textit{Delta-Bar-Delta} \cite{Jac88} and \textit{RPROP} algorithms for faster convergence. Parallel implementation on an Intel Paragon parallel computer is discussed in Section 4. Results and discussions on design issues 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 \textit{Proben1}² for ANN learning \cite{Pre94}. In \cite{Pre94}, Prechelt has included 15 benchmarking data sets from 12 different domains for ANN prediction and classification applications. All but one of the database are data from real experiments. The heart disease data base is one of the 15 data sets available.

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.

\footnote{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” }
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 [Pre94] and also 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)

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 fluoroscopy

13. thallium: 3 = normal; 6 = fixed defect; 7 = reversible 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 [D+89, Pre94]. 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 "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. [D+89] 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. [GLF89] has achieved a 78.9% accuracy on the Cleveland database. Prechelt is one of the first researchers that uses the same database to investigate ANN prediction with RPROP as the speed-up algorithm [Pre94]. The best classification accuracy obtained by Prechelt [Pre94] is between 76.1% to 85.2% for the entire database; and 78.7% to 96.0% for the Cleveland database, depending on the order of presentation of the data.

In Section 5 we verify Prechelt's results independently. In our case, we implemented both the Delta-Bar-Delta rule and the RPROP algorithm to speed up the convergence rate. For Rprop, we are able to achieve an averaged prediction accuracy of 76.6% to 83.3% for the entire database; and an accuracy of 79.1% to 90.1% for the Cleveland database. For Delta-Bar-Delta using the averaged value to replace missing values, the averaged prediction accuracy is between 76.6% to 86.7% for the combined database; and between 79.0% to 93.7% for the Cleveland database. These results are comparable to that of Prechelt's [Pre94] which is slightly better than the probabilistic models [D+89, GLF89].

More importantly, we implemented the ANN learning algorithm in parallel through the "pattern partitioning" concept to be discussed in Section 4.2. Through such parallel implementation with 5 processors, training of 460 sets of data with 230 validation sets for the entire database can be achieved in < 5 minutes whereas training of 150 data sets with 75 validation sets takes < 1 minute for the Cleveland database. We, therefore, conclude that both the Delta-Bar-Delta rule and the RPROP algorithm can be used to speed-up conver-
gence and parallel implementation of ANN learning provides a fast alternative to classical statistical pattern recognition techniques.

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 [Kun93].

Neural networks offer several advantages over conventional computing architectures [Lip87]. Calculations are carried out in parallel yielding speed advantages and programming is done 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 [Day90]. 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 [HJV90, NP90]. Basically, it is a hierarchical design consisting of fully interconnected layers of neurons [RMS6]. 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 [WL90]. Errors are propagated backwards through the network, hence the name backpropagation. Figure 2 shows a backpropagation network with three hidden layers.
Although these multilayer feedforward ANNs have been proved to be universal function approximator [HS89, Fun89], the backpropagation learning algorithm is very slow. Various strategies, such as the Delta-Bar-Delta rule [Jac88] and the RPROP algorithm [RB93], 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 result of such feedforward calculation, errors for all training samples are calculated. Such errors are feedback 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 review the equations governing the backpropagation ANN algorithm and the Delta-Bar-Delta and the RPROP algorithms to speed up the convergence rate.

### 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 details of the algorithm can be found in [Wer90]. For the reader’s convenience, they are also summarized here.
Figure 3: A Flow-Chart for Training an ANN.
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:

\[
x_i(t) = X_i(t) \quad 1 \leq i \leq m
\]

for \( i = m + 1 \) to \( i = m + H + n \),

\[
\left\{ \begin{array}{l}
    net_i(t) = \sum_{j=1}^{i-1} W_{ij} x_j(t) \\
    x_i(t) = s(net_i(t))
\end{array} \right. \quad (2)
\]

\[
\hat{Y}_i(t) = x_{m+H+i}(t) \quad 1 \leq i \leq n
\]

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

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

where \( \hat{Y}_i(t) \) is the output of the neural network and \( Y_i(t) \) is the desired output. 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:

\[
F_{\hat{Y}_i}(t) = \frac{\partial E}{\partial \hat{Y}_i(t)} = \hat{Y}_i(t) - Y_i(t) \quad i = 1, \ldots, n
\]

for \( i = m + H + n \) to \( i = m + 1 \),

\[
\left\{ \begin{array}{l}
    F_x_i(t) = F_{\hat{Y}_{i-m-H}}(t) + \sum_{j=i+1}^{m+H+n} W_{ji} * F_{net_j}(t) \\
    F_{net_i}(t) = s'(net_i) * F_x_i(t)
\end{array} \right. \quad (6)
\]

\[
F_{W_{ij}} = \sum_{t=1}^{T} F_{net_i}(t) * x_j(t) \quad i, j = 1, \ldots, m + H + n \quad (7)
\]

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

\[
s(z) = 1/(1 + e^{-z}) \quad (8)
\]

\[
s'(z) = s(z) * (1 - s(z)) \quad (9)
\]
Once $F \cdot W_{ij}$ (the gradient of $E$ with respect to $W_{ij}$) is calculated, each weight is updated according to:

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

(10)

where $\alpha$ 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 [Jac88]. Basically, the algorithm is a special case of the Adaptive Learning Rate (ALR) discussed in [Wer74]. Every weight of the network is given its own learning rate and that the rate changes with time. According to [Jac88], the learning rate update rule is:

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

(11)

where

$$\begin{align*}
\delta_{ij}(t) &= F \cdot W_{ij} \\
\bar{\delta}_{ij}(t) &= (1-\theta)\delta_{ij}(t) + \theta \bar{\delta}_{ij}(t-1) \\
\alpha_{ij}(t) &= \alpha_{ij}(t-1) + \Delta \alpha_{ij}(t)
\end{align*}$$

(12)

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 $\theta$ as the base and time as the exponent [Jac88]. 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 $\kappa$. 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 $\phi$ of its current value [Jac88].

### 3.3 RPROP Learning Rule

RPROP (Resilient PROPagation) [RB93] 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 \( \Delta_{ij} \) and the magnification/reduction factors for the update-value, namely, \( \eta^+ \) and \( \eta^- \) are required to be set.)

For each weight, its individual update-value \( \Delta_{ij} \) 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 [RB93] for details.

\[
\Delta_{ij}(t) = \begin{cases} 
\eta^+ \Delta_{ij}(t-1), & \text{if } \frac{\partial E}{\partial w_{ij}}(t-1) \times \frac{\partial E}{\partial w_{ij}}(t) > 0 \\
\eta^- \Delta_{ij}(t-1), & \text{if } \frac{\partial E}{\partial w_{ij}}(t-1) \times \frac{\partial E}{\partial w_{ij}}(t) < 0 \\
\Delta_{ij}(t-1), & \text{otherwise.}
\end{cases}
\] (13)

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 \( \eta^- \). If the derivative retains the sign, the update value is increased by a factor \( \eta^+ \) 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 value. If the derivative is negative, the update value is added to the current weight. That is, Equation 10 is replaced by the following:

\[
\Delta w_{ij}(t) = \begin{cases} 
-\Delta_{ij}(t), & \text{if } \frac{\partial E}{\partial w_{ij}}(t) > 0 \\
+\Delta_{ij}(t), & \text{if } \frac{\partial E}{\partial w_{ij}}(t) < 0 \\
0, & \text{otherwise.}
\end{cases}
\] (14)

And,

\[
\text{New } W_{ij} = W_{ij}(t+1) = W_{ij}(t) + \Delta w_{ij}(t)
\] (15)

4 PARALLEL IMPLEMENTATION

To exploit the inherent parallel nature of ANN algorithms, we implement the backpropagation algorithm with the Delta-Bar-Delta and the RPROP rules on an Intel Paragon parallel computer. The Intel Paragon is a multiple instruction multiple data (MIMD) asynchronous machine that supports message-passing communications. The ANN algorithms can
be parallelized on this machine by two techniques - node partitioning and pattern partitioning [KSA94].

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 $x_i(t)$ (Equation 2) in parallel and broadcasts it to other processors for computation of next layer activations.

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 conceptually 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 $F_{\Delta W_{ij}}$ are computed at every processor:

$$F_{\Delta W_{ij}} = \sum_{k=1}^{N_p} F_{\Delta W_{ij}}(k)$$

where $F_{\Delta W_{ij}}(k)$ (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 RESULTS

In Section 2 we have presented the heart database 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 [Pre94]. 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 is therefore unable to generalize and provide an accurate
Stop Use Delta-Bar-Delta/Prop Update Weights Feedback Equations
Forward Equations

Is (# Iter < 5000) or (GL > GL Limit) or (Pk < Pk Limit)

Initialization

Set # Iter = 0, GL = 0, Pk = 1000

Start

Calculate the Validation Error (EvA_Curr)

EvA_Opt = Min (EvA_Curr)

GL = (EvA_Curr / EvA_Opt - 1.0) * 100

Pk = (Etr_Avg / Etr_Best - 1.0) * 1000

Calculate Error

Etr_Best = Min (Error of last W Iterations)

Etr_Avg = Avg (Error of last W Iterations)


Figure 6: Training Algorithm with Validation Set.
prediction to a new testing data. Readers interested in more discussions on the overfitting issue are referred to [GW94].

The heart database described in Section 2 has a total of 920 input-output data. Using the datafiles from [Pre94], the database is divided 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 all iterations, we keep check of the current validation error ($E_{va\_Curr}$) and the minimum validation error ($E_{va\_Opt}$). A scaled ratio of these two numbers gives an indication of how much of the generalization ability is lost as a result of more training. Prechelt called such an indicator, $GL$, the generalization lost parameter [Pre94]. Besides, $GL$, there is the progress indicator $PK$ which is a scaled ratio of the averaged training error and the optimal training error within the last $W$ iterations ($W = 10$ in our implementation). The $PK$ parameter indicates the progress in reducing the training error. The algorithm terminates from one of the three stopping criteria: (1) if there is too many iterations (maximum iteration of 5,000 is used); or (2) when there is large generalization loss ($GL > GL\_Limit$); or (3) when there is little progress in further training ($PK < PK\_Limit$). Figure 6 summarizes the training algorithm and provides the exact formulation of the $GL$ and $PK$ parameters. 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 melange of all the data, it is suggested in [Pre94] 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 [Pre94].

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 our preliminary investigation, we found that relatively small size networks are sufficient for this application. As will be explained in subsequent sections, we use two-layered
networks with each layer having 3, 5, 10 neurons. Furthermore, there are 460 training samples and 230 validation and testing samples. With this relatively large number of training samples on small networks, we found that pattern-partitioning is the more appropriate parallelization technique. Therefore, in this paper, we only implement the pattern-partitioning technique. The following issues are investigated: (1) the number of processors used in parallel implementation; (2) the ANN architecture (number of neurons in each layer); and (3) the handling of the missing values (noisy data).

5.1 Issue 1: Number of Processors

For pattern-partitioning, the entire set of training samples are equally divided among the different processors. To determine the optimal number of processors to be deployed, we use the database Heart1 with $T = 440$ training samples and 220 validation and 220 testing samples on $N_p = 1, 2, 4, 5, 10, 20, 40$ processors. Each processor computes the forward and backward loops for $T_p = T/N_p = 440, 220, 110, 88, 44, 22, 11$ training samples respectively. Only 440 training samples are used mainly because the number of training samples must be divisible by the number of processors. We ran both the RPROP and Delta-Bar-Delta algorithms for a two-layered neural network with 5 hidden nodes in each layer. For each case, 10 runs were performed and the average time per iteration is calculated. The time
for each run varies slightly due to the changing load of the parallel system and the physical location of the processors. The paragon computer used here has a total of 110 nodes. When large number of processors are used, say \( N_p \geq 10 \), some of the processors may be far away from others and therefore incurred a longer communication delay.

Figure 7 plots the time per iteration vs the different number of processors. For clarity, we only plotted the results from the Delta-Bar-Delta algorithm as the RPROP results are very similar. This is due to the fact that the two algorithms differ only in how the weights are adjusted and have little effect on the time required per iteration. As expected, the time per iteration decreases with the number of processors initially. However, when the number of processors \( N_p \geq 20 \), the time per iteration starts to increase. This is due to the fact that with such large number of processors, each processor is only responsible for 22 or less number of samples. From this figure, we observe that when \( N_p = 10 \), the algorithm is most time-efficient. However, the time per iteration for 5 processors (\( N_p = 5 \)) is only slightly higher than that of \( N_p = 10 \). For cost-effectiveness, we use 5 processors for our subsequent investigations.

5.2 Issue 2: ANN Architecture

To compare the effect of the network architecture, we investigate three different size two-layered networks with \( H_1, H_2 \) neurons in each layer, where \( H_1 = H_2 = 3, 5, 10 \) respectively. As explained in earlier section that there are three criteria to terminate the algorithm: (1) if the number of iterations exceeds 5,000; (2) if the generalization loss parameter is bigger than its limit (i.e., \( GL > GL_{Limit} \)); or (3) if the PK indicator is smaller than its limit (i.e., \( PK < PK_{Limit} \)). Throughout our investigation, the numerical values for these limits are \( GL_{Limit} = 100, PK_{Limit} = 0.1 \). These values are chosen according to [Pre94].

The results for the Heart1, Heart2, and Heart3 databases are summarized in Tables 2 and 3. The tables include the averaged number of iterations that the algorithm terminated, averaged accuracy of the test set which was not presented to the network during training; and the averaged amount of time per iteration. We found that for RPROP, the algorithm is more robust. A total of 10 runs were done for each of the Heart1, Heart2, Heart3 database. (The symbol *10 at the top right corner of each entry indicates that the averaged values were computed from 10 runs).

For Delta-Bar-Delta, the algorithm is less robust because we need to choose appropriate Delta-Bar-Delta parameters \( (\alpha_0, \theta, \phi, \kappa) \) where \( \alpha_0 \) is the initial learning rate, and \( \theta, \phi, \kappa \)
### Table 2: Results for Heart Database with RPROP

<table>
<thead>
<tr>
<th>Data File</th>
<th>(H1,H2)</th>
<th>(3,3)</th>
<th>(5,5)</th>
<th>(10,10)</th>
<th>Averaged</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heart 1</td>
<td>Iteration</td>
<td>674.5 *10</td>
<td>1060.5 *10</td>
<td>1017.0 *10</td>
<td>80.0% *30</td>
</tr>
<tr>
<td></td>
<td>Accuracy</td>
<td>80.5% *10</td>
<td>79.4% *10</td>
<td>80.2% *10</td>
<td>83.4% *30</td>
</tr>
<tr>
<td></td>
<td>Time/Iter</td>
<td>38.0 ms *10</td>
<td>61.1 ms *10</td>
<td>121.6 ms *10</td>
<td>77.1% *30</td>
</tr>
<tr>
<td>Heart 2</td>
<td>Iteration</td>
<td>857.5 *10</td>
<td>1315.5 *10</td>
<td>1169.0 *10</td>
<td>83.3% *30</td>
</tr>
<tr>
<td></td>
<td>Accuracy</td>
<td>83.4% *10</td>
<td>83.1% *10</td>
<td>83.1% *10</td>
<td>80.2% *10</td>
</tr>
<tr>
<td></td>
<td>Time/Iter</td>
<td>39.0 ms *10</td>
<td>60.5 ms *10</td>
<td>127.6 ms *10</td>
<td>83.3% *30</td>
</tr>
<tr>
<td>Heart 3</td>
<td>Iteration</td>
<td>558.5 *10</td>
<td>1038.0 *10</td>
<td>767.5 *10</td>
<td>76.6% *30</td>
</tr>
<tr>
<td></td>
<td>Accuracy</td>
<td>75.4% *10</td>
<td>77.1% *10</td>
<td>77.2% *10</td>
<td>80.0% *10</td>
</tr>
<tr>
<td></td>
<td>Time/Iter</td>
<td>40.5 ms *10</td>
<td>57.3 ms *10</td>
<td>116.7 ms *10</td>
<td>80.0% *10</td>
</tr>
</tbody>
</table>

### Table 3: Results for Heart Database with Delta-Bar-Delta

<table>
<thead>
<tr>
<th>Data File</th>
<th>(H1,H2)</th>
<th>(3,3)</th>
<th>(5,5)</th>
<th>(10,10)</th>
<th>Averaged</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heart 1</td>
<td>Iteration</td>
<td>2207.5 *4</td>
<td>2215.3 *15</td>
<td>2813.0 *10</td>
<td>79.9% *29</td>
</tr>
<tr>
<td></td>
<td>Accuracy</td>
<td>78.1% *4</td>
<td>80.3% *15</td>
<td>79.9% *10</td>
<td>79.9% *29</td>
</tr>
<tr>
<td></td>
<td>Time/Iter</td>
<td>30.9 ms *4</td>
<td>46.7 ms *15</td>
<td>94.5 ms *10</td>
<td>79.9% *29</td>
</tr>
<tr>
<td>Heart 2</td>
<td>Iteration</td>
<td>1700.0 *7</td>
<td>3011.5 *13</td>
<td>3761.8 *11</td>
<td>83.4% *31</td>
</tr>
<tr>
<td></td>
<td>Accuracy</td>
<td>84.5% *7</td>
<td>82.4% *13</td>
<td>83.8% *11</td>
<td>83.4% *31</td>
</tr>
<tr>
<td></td>
<td>Time/Iter</td>
<td>31.1 ms *7</td>
<td>46.6 ms *13</td>
<td>93.5 ms *11</td>
<td>83.4% *31</td>
</tr>
<tr>
<td>Heart 3</td>
<td>Iteration</td>
<td>1446.2 *13</td>
<td>1153.1 *16</td>
<td>1084.7 *17</td>
<td>64.6% *46</td>
</tr>
<tr>
<td></td>
<td>Accuracy</td>
<td>63.7% *13</td>
<td>63.2% *16</td>
<td>66.6% *17</td>
<td>64.6% *46</td>
</tr>
<tr>
<td></td>
<td>Time/Iter</td>
<td>31.2 ms *13</td>
<td>47.3 ms *16</td>
<td>94.8 ms *17</td>
<td>64.6% *46</td>
</tr>
</tbody>
</table>
### Table 4: Results for Cleveland Database with RPROP

<table>
<thead>
<tr>
<th>Data File</th>
<th>(H1,H2)</th>
<th>(3,3)</th>
<th>(5,5)</th>
<th>(10,10)</th>
<th>Averaged</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cleveland 1</td>
<td>Iteration</td>
<td>327.5</td>
<td>244.5</td>
<td>124.0</td>
<td>79.1%</td>
</tr>
<tr>
<td></td>
<td>Accuracy</td>
<td>79.4%</td>
<td>78.7%</td>
<td>79.3%</td>
<td>79.1%</td>
</tr>
<tr>
<td></td>
<td>Time/Iter</td>
<td>17.5 ms</td>
<td>30.7 ms</td>
<td>58.8 ms</td>
<td>19.6 ms</td>
</tr>
<tr>
<td>Cleveland 2</td>
<td>Iteration</td>
<td>436.0</td>
<td>501.0</td>
<td>338.0</td>
<td>90.1%</td>
</tr>
<tr>
<td></td>
<td>Accuracy</td>
<td>86.9%</td>
<td>92.2%</td>
<td>92.4%</td>
<td>90.1%</td>
</tr>
<tr>
<td></td>
<td>Time/Iter</td>
<td>19.8 ms</td>
<td>27.4 ms</td>
<td>51.8 ms</td>
<td>19.8 ms</td>
</tr>
<tr>
<td>Cleveland 3</td>
<td>Iteration</td>
<td>340.5</td>
<td>390.5</td>
<td>232.0</td>
<td>84.2%</td>
</tr>
<tr>
<td></td>
<td>Accuracy</td>
<td>84.3%</td>
<td>84.4%</td>
<td>83.5%</td>
<td>84.2%</td>
</tr>
<tr>
<td></td>
<td>Time/Iter</td>
<td>19.6 ms</td>
<td>25.6 ms</td>
<td>50.3 ms</td>
<td>19.6 ms</td>
</tr>
</tbody>
</table>

### Table 5: Results for Cleveland Database with Delta-Bar-Delta

<table>
<thead>
<tr>
<th>Data File</th>
<th>(H1,H2)</th>
<th>(3,3)</th>
<th>(5,5)</th>
<th>(10,10)</th>
<th>Averaged</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cleveland 1</td>
<td>Iteration</td>
<td>683.3</td>
<td>638.9</td>
<td>450.5</td>
<td>76.5%</td>
</tr>
<tr>
<td></td>
<td>Accuracy</td>
<td>70.4%</td>
<td>78.7%</td>
<td>80.0%</td>
<td>76.5%</td>
</tr>
<tr>
<td></td>
<td>Time/Iter</td>
<td>15.8 ms</td>
<td>22.0 ms</td>
<td>41.7 ms</td>
<td>15.8 ms</td>
</tr>
<tr>
<td>Cleveland 2</td>
<td>Iteration</td>
<td>1005.0</td>
<td>1020.0</td>
<td>735.0</td>
<td>94.6%</td>
</tr>
<tr>
<td></td>
<td>Accuracy</td>
<td>94.0%</td>
<td>93.3%</td>
<td>95.3%</td>
<td>94.6%</td>
</tr>
<tr>
<td></td>
<td>Time/Iter</td>
<td>14.7 ms</td>
<td>20.7 ms</td>
<td>40.5 ms</td>
<td>14.7 ms</td>
</tr>
<tr>
<td>Cleveland 3</td>
<td>Iteration</td>
<td>1076.6</td>
<td>700.0</td>
<td>612.6</td>
<td>84.9%</td>
</tr>
<tr>
<td></td>
<td>Accuracy</td>
<td>85.5%</td>
<td>84.7%</td>
<td>84.6%</td>
<td>84.9%</td>
</tr>
<tr>
<td></td>
<td>Time/Iter</td>
<td>15.2 ms</td>
<td>21.8 ms</td>
<td>40.8 ms</td>
<td>15.2 ms</td>
</tr>
</tbody>
</table>
were defined in Equations 11 and 12. Our preliminary study [TC94] indicates that network performance was mostly affected by $\alpha_0, \kappa$. Consequently, we fixed $\theta = 0.7$ and $\phi = 0.55$ and $\alpha_0, \kappa$ are randomly generated in the range $\alpha_0 \in [0.1, 0.9]$ and $\kappa \in [0.01, 0.09]$. For some of these combinations, the algorithm terminated because of one of the three stopping criteria were satisfied but the best error for the validation set ($E_{val}$) in Figure 6) is still very large. As mentioned earlier, the validation set is an indication of the generalization ability of the neural network. Large validation set error, in most cases, corresponds to large testing error. We, therefore, discarded these results. This practice is acceptable because during the training phase, the validation set is being presented to the network and the validation error is a known quantity. A total of 20 runs were done for Delta-Bar-Delta but only those with best validation error $< 17$ were accepted in calculation of the averaged iteration, accuracy, and time per iteration. The $*n$ symbol of each entry of Table 3 indicates that entry were computed using $n$ runs.

From Tables 2 and 3 we have the following observations. In general, the time per iteration for Delta-Bar-Delta is less but the number of iteration required for the latter is also larger. As for test accuracy, the averaged accuracy (over all runs and architectures) for Heart1, Heart2 and Heart3 are 80.0%, 83.3%, 76.6% for RPROP; and are 79.9%, 83.4%, 64.6% for Delta-Bar-Delta. We attribute the inferior results for Heart3 to its particular permutation. (This observation is consistent with [Pre94] which also reported a larger classification error for Heart3.) However, we concede that Delta-Bar-Delta has a more pronounced disparity among Heart3 and the other two databases. We attribute this phenomenon to the fact that in Delta-Bar-Delta, the actual value of the weight gradient is used in weight adaptation (Equation 10). On the other hand, for RPROP, only the sign of the weight gradient is used for weight adaptation (Equation 15). Consequently, RPROP is more robust and less affected by the order of presentation of the database.

Comparing the different network architectures, we observed little difference in test accuracy for both algorithms. As expected, larger network does take more time per iteration. Also, for Delta-Bar-Delta, the smallest network $(H_1, H_2 = 3, 3)$ has the smallest number of acceptable runs (only 4 out of 20 runs have validation error $< 17$). This indicates that for smaller network, the choice of the Delta-Bar-Delta parameter is more critical. Non-optimal choices of the parameters render training unsuccessful.

Tables 4 and 5 present the results for the Cleveland Foundation results. Similar observation can be drawn. The averaged test accuracy over the three network architectures $(H_1 = H_2 = 3, 5, 10)$ for Cleveland1, Cleveland2, Cleveland3 are 79.1%, 90.1%, 84.2% for
RPROP; and are 76.5%, 94.6%, 84.9% for Delta-Bar-Delta. There is not much difference between the two algorithms. But the Delta-Bar-Delta rule has a smaller number of runs with best validation error < 17.

In conclusion, in this section, we compared the two algorithms (RPROP and Delta-Bar-Delta) using three network architectures. We found that in general, RPROP is more robust and less affected by the order of presentation of the database. This is because, for RPROP, weight adaptation involves only the sign rather than the actual value of the weight gradient. There is little effect in test accuracy among the three architectures for the same weight adaptation algorithm. However, for Delta-Bar-Delta, smaller network is more parsimonious in choosing its learning parameters.

5.3 Issue 3: Missing Values

In previous section, we have presented results for both RPROP and Delta-Bar-Delta using different network architectures and the original database of [Pre94]. As described in Section 2, these databases were obtained from real data collected by international medical institutions. As for all real data, there are missing values. In [Pre94], these missing values are replaced by 0. In general, the zero value is not in the range of the regular values. Since the handling of missing values is an important issue in practical pattern recognition problem, in this section, we investigate an alternate method of handling the missing values. In particular, we use the averaged value of the attribute to replace the missing value.

Table 6 is the results for RPROP on the Heart1, Heart2, Heart3 databases. When compared with Table 2, we found that the test accuracy is slightly smaller and time is marginally higher. Otherwise, there is not much difference. Table 7 corresponds the results for Delta-Bar-Delta with Heart1, Heart2, Heart3. With the missing values filled with the attribute’s averaged value, the disparity between Heart3 and the other two databases is greatly reduced. We believe that this is because the averaged missing values have smoothed the data. This smoothing effect made learning with Delta-Bar-Delta rule easier because, in this case, the numerical value of the weight gradient is used for weight adaptation.

For the Cleveland database, Tables 8 and 9 (compared with Tables 4 and 5) show the two approaches of handling missing values produce very similar results. This is because the Cleveland database has only 2 missing values, and therefore, we do not expect much difference in how the missing values are treated.
### Table 6: Results for Heart Database with RPROP and Averaged Missing Values

<table>
<thead>
<tr>
<th>Data File</th>
<th>(H1,H2)</th>
<th>Iteration</th>
<th>Accuracy</th>
<th>Time/Iter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heart 1</td>
<td>(3,3)</td>
<td>1545.0</td>
<td>79.1%</td>
<td>39.8 ms</td>
</tr>
<tr>
<td></td>
<td>(5,5)</td>
<td>837.5</td>
<td>80.2%</td>
<td>59.0 ms</td>
</tr>
<tr>
<td></td>
<td>(10,10)</td>
<td>1722.0</td>
<td>80.6%</td>
<td>142.4 ms</td>
</tr>
<tr>
<td></td>
<td>Averaged</td>
<td>80.1%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Heart 2</td>
<td>(3,3)</td>
<td>692.1</td>
<td>73.2%</td>
<td>43.8 ms</td>
</tr>
<tr>
<td></td>
<td>(5,5)</td>
<td>1427.5</td>
<td>79.9%</td>
<td>60.7 ms</td>
</tr>
<tr>
<td></td>
<td>(10,10)</td>
<td>1388.0</td>
<td>77.9%</td>
<td>124.4 ms</td>
</tr>
<tr>
<td></td>
<td>Averaged</td>
<td>77.4%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Heart 3</td>
<td>(3,3)</td>
<td>661.7</td>
<td>73.6%</td>
<td>42.1 ms</td>
</tr>
<tr>
<td></td>
<td>(5,5)</td>
<td>1199.5</td>
<td>72.2%</td>
<td>55.4 ms</td>
</tr>
<tr>
<td></td>
<td>(10,10)</td>
<td>1188.0</td>
<td>73.0%</td>
<td>124.3 ms</td>
</tr>
<tr>
<td></td>
<td>Averaged</td>
<td>72.9%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 7: Results for Heart Database with Delta-Bar-Delta and Averaged Missing Values

<table>
<thead>
<tr>
<th>Data File</th>
<th>(H1,H2)</th>
<th>Iteration</th>
<th>Accuracy</th>
<th>Time/Iter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heart 1</td>
<td>(3,3)</td>
<td>2456.0</td>
<td>75.4%</td>
<td>30.7 ms</td>
</tr>
<tr>
<td></td>
<td>(5,5)</td>
<td>2608.3</td>
<td>81.6%</td>
<td>46.9 ms</td>
</tr>
<tr>
<td></td>
<td>(10,10)</td>
<td>3172.5</td>
<td>81.8%</td>
<td>93.63 ms</td>
</tr>
<tr>
<td></td>
<td>Averaged</td>
<td>80.4%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Heart 2</td>
<td>(3,3)</td>
<td>2682.5</td>
<td>81.8%</td>
<td>31.0 ms</td>
</tr>
<tr>
<td></td>
<td>(5,5)</td>
<td>2708.0</td>
<td>79.3%</td>
<td>46.6 ms</td>
</tr>
<tr>
<td></td>
<td>(10,10)</td>
<td>4573.3</td>
<td>80.2%</td>
<td>93.5 ms</td>
</tr>
<tr>
<td></td>
<td>Averaged</td>
<td>80.7%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Heart 3</td>
<td>(3,3)</td>
<td>1640.0</td>
<td>74.1%</td>
<td>31.3 ms</td>
</tr>
<tr>
<td></td>
<td>(5,5)</td>
<td>2076.7</td>
<td>77.1%</td>
<td>46.8 ms</td>
</tr>
<tr>
<td></td>
<td>(10,10)</td>
<td>2165.8</td>
<td>76.6%</td>
<td>94.1 ms</td>
</tr>
<tr>
<td></td>
<td>Averaged</td>
<td>76.6%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 8: Results for Cleveland Database with RPROP and Averaged Missing Values

<table>
<thead>
<tr>
<th>Data File</th>
<th>(H1,H2)</th>
<th>Iteration</th>
<th>Accuracy</th>
<th>Time/Iter</th>
<th>Averaged</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cleveland 1</td>
<td>(3,3)</td>
<td>413.5 *20</td>
<td>80.0% *20</td>
<td>20.3 ms *20</td>
<td>80.2% *48</td>
</tr>
<tr>
<td></td>
<td>(5,5)</td>
<td>237.2 *18</td>
<td>80.4% *18</td>
<td>30.9 ms *18</td>
<td>80.6% *10</td>
</tr>
<tr>
<td></td>
<td>(10,10)</td>
<td>187.0 *10</td>
<td>80.4% *10</td>
<td>55.4 ms *10</td>
<td>80.4% *10</td>
</tr>
<tr>
<td></td>
<td>Averaged</td>
<td></td>
<td></td>
<td></td>
<td>80.2% *48</td>
</tr>
<tr>
<td>Cleveland 2</td>
<td>(3,3)</td>
<td>513.0 *20</td>
<td>91.1% *20</td>
<td>20.0 ms *20</td>
<td>91.9% *48</td>
</tr>
<tr>
<td></td>
<td>(5,5)</td>
<td>423.5 *20</td>
<td>92.1% *20</td>
<td>28.0 ms *20</td>
<td>93.2% *20</td>
</tr>
<tr>
<td></td>
<td>(10,10)</td>
<td>341.3 *8</td>
<td>93.2% *8</td>
<td>53.5 ms *8</td>
<td>93.2% *8</td>
</tr>
<tr>
<td></td>
<td>Averaged</td>
<td></td>
<td></td>
<td></td>
<td>91.9% *48</td>
</tr>
<tr>
<td>Cleveland 3</td>
<td>(3,3)</td>
<td>392.1 *20</td>
<td>83.1% *20</td>
<td>19.0 ms *20</td>
<td>83.9% *49</td>
</tr>
<tr>
<td></td>
<td>(5,5)</td>
<td>376.8 *19</td>
<td>84.3% *19</td>
<td>26.5 ms *19</td>
<td>84.4% *19</td>
</tr>
<tr>
<td></td>
<td>(10,10)</td>
<td>261.0 *10</td>
<td>84.7% *10</td>
<td>48.2 ms *10</td>
<td>84.7% *10</td>
</tr>
<tr>
<td></td>
<td>Averaged</td>
<td></td>
<td></td>
<td></td>
<td>83.9% *49</td>
</tr>
</tbody>
</table>

### Table 9: Results for Cleveland Database with Delta-Bar-Delta and Averaged Missing Values

<table>
<thead>
<tr>
<th>Data File</th>
<th>(H1,H2)</th>
<th>Iteration</th>
<th>Accuracy</th>
<th>Time/Iter</th>
<th>Averaged</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cleveland 1</td>
<td>(3,3)</td>
<td>727.6 *13</td>
<td>74.0% *13</td>
<td>16.4 ms *13</td>
<td>79.0% *51</td>
</tr>
<tr>
<td></td>
<td>(5,5)</td>
<td>499.4 *18</td>
<td>79.6% *18</td>
<td>23.4 ms *18</td>
<td>81.6% *20</td>
</tr>
<tr>
<td></td>
<td>(10,10)</td>
<td>508.0 *20</td>
<td>81.6% *20</td>
<td>42.9 ms *20</td>
<td>81.6% *20</td>
</tr>
<tr>
<td></td>
<td>Averaged</td>
<td></td>
<td></td>
<td></td>
<td>79.0% *51</td>
</tr>
<tr>
<td>Cleveland 2</td>
<td>(3,3)</td>
<td>2095.0 *2</td>
<td>92.0% *2</td>
<td>15.2 ms *2</td>
<td>93.7% *11</td>
</tr>
<tr>
<td></td>
<td>(5,5)</td>
<td>930.0 *1</td>
<td>94.7% *1</td>
<td>22.2 ms *1</td>
<td>94.0% *8</td>
</tr>
<tr>
<td></td>
<td>(10,10)</td>
<td>766.3 *8</td>
<td>94.0% *8</td>
<td>40.9 ms *8</td>
<td>94.0% *8</td>
</tr>
<tr>
<td></td>
<td>Averaged</td>
<td></td>
<td></td>
<td></td>
<td>93.7% *11</td>
</tr>
<tr>
<td>Cleveland 3</td>
<td>(3,3)</td>
<td>1000.0 *6</td>
<td>84.1% *6</td>
<td>15.7 ms *6</td>
<td>84.5% *29</td>
</tr>
<tr>
<td></td>
<td>(5,5)</td>
<td>632.5 *12</td>
<td>84.4% *12</td>
<td>23.4 ms *12</td>
<td>84.7% *11</td>
</tr>
<tr>
<td></td>
<td>(10,10)</td>
<td>731.8 *11</td>
<td>84.7% *11</td>
<td>41.5 ms *11</td>
<td>84.7% *11</td>
</tr>
<tr>
<td></td>
<td>Averaged</td>
<td></td>
<td></td>
<td></td>
<td>84.5% *29</td>
</tr>
</tbody>
</table>
In conclusion, in this section, we investigated the effect of how missing values are treated. We found that for RPROP, the effect is minimal whereas for Delta-Bar-Delta, using the averaged values to replace the missing value helps to smooth the data and made learning easier.

6 CONCLUSIONS

In this paper, we investigated the use of backpropagation ANNs for the diagnosis of coronary disease based on a patient’s personal data such as age, gender, subjective patient pain descriptions; and results of various medical examinations such as blood pressure and electrocardiogram 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 [D+F89] and 78.9% for the Cleveland Foundation data [GLF89].

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. To further reduce the training time, we use a parallel computer, the Intel Paragon computer, with pattern-partitioning implementation. We found that, in general, the RPROP algorithm is more robust and tends to give similar results regardless of network architectures, order of presentation of database, or how the missing values of data are treated.

We have tested the algorithm on two kinds of datafiles, one is the combined datafile from all four medical organizations; and the second one consists of data from the Cleveland Foundation only. We used the combined dataset mainly to be consistent with Prechelt’s approach but also because the number of data samples available from individual institution is small. With the exception of the Cleveland Foundation, the other three institutions have less than 300 samples and all have missing values (Table 1). The data set from the Cleveland Foundation, on the other hand, has 303 samples and only two missing values.
For the Heart1,2,3 datafiles (the combined data set) and using RPROP, the averaged testing accuracy is between 76.6% to 83.3%. For the Cleveland1,2,3 datafiles (data from Cleveland Foundation only), the averaged accuracy is between 79.1% to 90.1%. For Delta-Bar-Delta, it is more parsimonious in choosing its learning parameter. The best results are from using the averaged value of an attribute to replace the missing values. In this case, the averaged accuracy for the combined dataset ranges from 76.6% to 86.7% and for the Cleveland database, from 79.0% to 93.7%. These results are consistent with that of [Pre94] which uses only the RPROP algorithm and no parallel implementation.

Comparing these results with earlier results from probabilistic and statistical techniques, the ANN prediction accuracy is comparable. More importantly, for computation time, the ANN algorithm shows a bigger advantage over classical statistical techniques. Exploiting parallel computation with 5 processors, we are able to achieve training in < 5 minutes for the combined data file and < 1 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.

ACKNOWLEDGEMENT

The authors would like to thank the Center for Scientific Computing of the Applied Mathematics Department in SUNY Stony Brook for the use of the Intel Paragon parallel computer. The authors also like to thank the anonymous reviewers for many insightful comments and suggestions to improve the article.

References


