AUTOMATED EKG ANNOTATION WITH NEURAL NETWORKS

By

MESFIN B. TAYE

Master of Science in Mathematics
Addis Ababa University
Addis Ababa, Ethiopia
2012

Submitted to the Faculty of the
Graduate College of
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
December, 2016

AUTOMATED EKG ANNOTATION WITH NEURAL NETWORKS

Thesis Approved:

Dr. Martin T. Hagan

Thesis Advisor

Dr. Keith A. Teague

Dr. Carl D. Latino

Dr. Sheryl A. Tucker

Dean of the Graduate College

TABLE OF CONTENTS

Chapter Page

1 INTRODUCTION 1

1.1 Problem Description 1

2 BASIC NEURAL NETWORK CONCEPTS AND NOTATION 4

2.1 Single-Input Neuron 4

2.2 Multiple-Input Neuron 4

2.3 A Layer of Multiple Neurons 5

2.4 Multilayer Network 6

2.5 Transfer Function 7

2.6 Focused Time Delay Network 7

2.6.1 Example 8

2.7 Scaling the Network Input 10

3 TRAINING NEURAL NETWORKS 11

3.1 Supervised Learning 11

3.2 Performance Index 11

3.3 Training Algorithm 12

3.4 Gradient Calculation 13

3.5 Stopping Criteria 16

4 DATA FOR QRS DETECTION 17

4.1 Description of the Data 17

4.2 Training and Testing Data 18

4.2.1 Group Formulation 18

5 LOCATION OF QRS COMPLEX 21

5.1 Committee of Networks 22

5.1.1 Committee Formulation Algorithm 22

5.2 Definition of Errors 25

5.3 Setting Algorithm Parameters 26

5.3.1 Number of Neurons, Detection Threshold and Error Ratio . . 26

5.3.2 Number of Iterations per Cycle 28

5.3.3 Detection Threshold 29

5.4 Summary 29

6 RESULTS FOR QRS DETECTION 33

6.1 Error Analysis 35

6.1.1 Inconsistent Physician Annotations 36

6.1.2 Noisy and Irregular Sequences 39

6.1.3 Small Amplitude Signals 40

6.2 Comparison with the Pan-Tompkins Algorithm 41

7 DATA FOR BEAT CLASSIFICATION 45

7.1 Selection of Beats 45

7.2 Data Description 46

8 NETWORK AND TRAINING FOR BEAT CLASSIFICATION 48

8.1 Network Architecture 48

8.2 Performance Index 49

8.3 Training Algorithm and Stopping Criteria 49

8.4 Committee of Networks 50

8.4.1 Committee Using Full Data 50

8.4.2 Committee Using Balanced Data 50
8.5 Definition of Errors 51

9 BEAT CLASSIFICATION RESULTS 53

9.1 Errors for Balanced Data Committee 54
9.2 Errors for Full Data Committee 54
9.2.1 Using Voting Threshold 56
9.2.2 Using Detection Threshold 59
9.3 Error Analysis 59
9.3.1 Difficult to Classify Beats 59
9.3.2 Types of Errors 65

10 CONCLUSIONS 72

10.1 Future Work 74

BIBLIOGRAPHY 75

LIST OF TABLES

Table Page

4.1 Iteration Times vs. Number of Sequences 19

4.2 Time (sec) Required for Two Iterations with 700 Sequences 19

5.1 Symbol Definitions 25

5.2 Performance Measures QRS Detection 26

5.3 Optimization of d_t, S^1 and ρ 27

5.4 Optimization of S^1 28

5.5 Comparison for Number of Iterations 30

6.1 Training Error Summary for the Three Networks and the Committee 33

6.2 Testing Error Summary for the Three Networks and the Committee . 35

6.3 Comparison of Neural Network and Pan-Tompkins Methods 43

6.4 Test Set Comparison of Neural Network and Pan-Tompkins Methods 44

7.1 Positive and Negative Beats in Normal and Abnormal Sets 47

9.1 Errors for Balanced Data with Varying v_t 55

9.2 Error Weighting Ratio Selection 57

9.3 Error Calculation Using Voting for Full Data, $\rho = 5$ 58

9.4 Error Calculation Using Voting for Full Data with $\rho = 39$ 60

9.5 Error Calculation Using Threshold with $\rho = 5$ 61

9.6 Error Calculation Using Threshold with $\rho = 39$ 62

9.7 Error Calculation with $\rho = 60$ 63

9.8 Types of Neighbors for FNs and FPs 64

9.9 Minimum Balanced Errors 65
9.10 Neighbors Classification for $\rho = 39$ 65

LIST OF FIGURES

Figure Page

1.1 Schematic Diagram of Normal Sinus Rhythm[12] 2

2.1 Single Neuron 4

2.2 Multiple Input Neuron 5

2.3 A Layer of Multiple Neurons 6

2.4 Three Layers Network 7

2.5 A Delay Block 8

2.6 Tap Delay Line 8

2.7 Focused Time Delay Network 9

5.1 EKG signal segment and network output 21

5.2 Detection Threshold (d_t) Optimization 31

6.1 RER Versus Iteration Number on One Network with $d_t = 3$ 34

6.2 RER Versus Iteration Number on One Network with $d_t = 50$ 35

6.3 Number of Sequences Versus Number of Errors 36

6.4 Negative and Positive Sequences with no Error 36

6.5 Intra-sequence Inconsistency of Annotation 37

6.6 Intra-sequence Inconsistency of Annotation (Expanded) 38

6.7 Inter-sequence Inconsistency of Annotation 38

6.8 Multiple Targets per Cardiac Cycle 39

6.9 Three Targets in a Cardiac Cycle 39

6.10 Sequence with Multiple R Wave Indications 40

6.11 Noisy Sequence 41
6.12 Irregular Sequence 42
6.13 A Low Amplitude Sequence 42
6.14 NN and Pan-Tompkins Methods on a Positive Sequence 43
6.15 NN and Pan-Tompkins Methods on a Negative Sequence 44

7.1 Abnormal Beat of Type V 46
7.2 Normal Beat 46
7.3 Negative Beats 47

8.1 Beat Classification Network 48
8.2 ROC 52

9.1 ROC curve for Balanced Data 54
9.2 ROC Curve for Full Data Trained with $\rho = 5$ 57
9.3 True Negative Neighbors of a False Negative Beat 67
9.4 True Positive Neighbors of a Missclassified False Positive 68
9.5 Exactly the Same Sequences Annotated Differently 2 69
9.6 Exactly the Same Sequences Annotated Differently 1 70
9.7 Exactly the Same Beats Annotated Differently 71

CHAPTER 1

INTRODUCTION

This report describes the opening phase of a project to develop an automated Electrocardiogram (EKG) annotation system. The electrical activity of the heart is measured by the EKG, as shown in Figure 1.1. At rest, the interior of myocytes (heart muscle cells) is negatively polarized. When these cells depolarize, they become positive and contract. The depolarization wave of positive charges flows outward from the Sinus Node, which is the main pacemaker of the heart. It is this electrical phenomena of polarization and depolarization that is recorded on the EKG.

The depolarization wave on both atria produces a wave called the P wave. The ventricular depolarization, on the other hand, produces the QRS complex. The Q wave is the part of this complex that is directed downward. It may or may not appear in the complex. However, if it appears, it must appear before the R wave, the upward wave. The downward wave of the complex is the S wave. After the QRS complex, there is a flat baseline called the ST segment, which is the initial part of ventricular re-polarization. This re-polarization produces the T wave. A cardiac cycle is a cycle that contains the P wave, QRS complex and the T wave. A single EKG sequence may contain several cycles.

1.1 Problem Description

An electrode is the skin sensor that is used to measure the heart signals. Two electrodes are required to record a lead. Though the standard EKG recording has twelve leads, several heart arrhythmias can be detected using a single lead EKG. Based on a one lead recording method, devices have been developed to record and store electric signals of individuals. Wireless ambulatory electrocardiograms have changed the way we collect EKG signals. Now we can monitor our heart from anywhere. These monitoring devices use smart phone apps to detect potential problems and to transfer the signals to professional interpreters. In addition, these applications send alarms through email or other communication channels when they need attention.

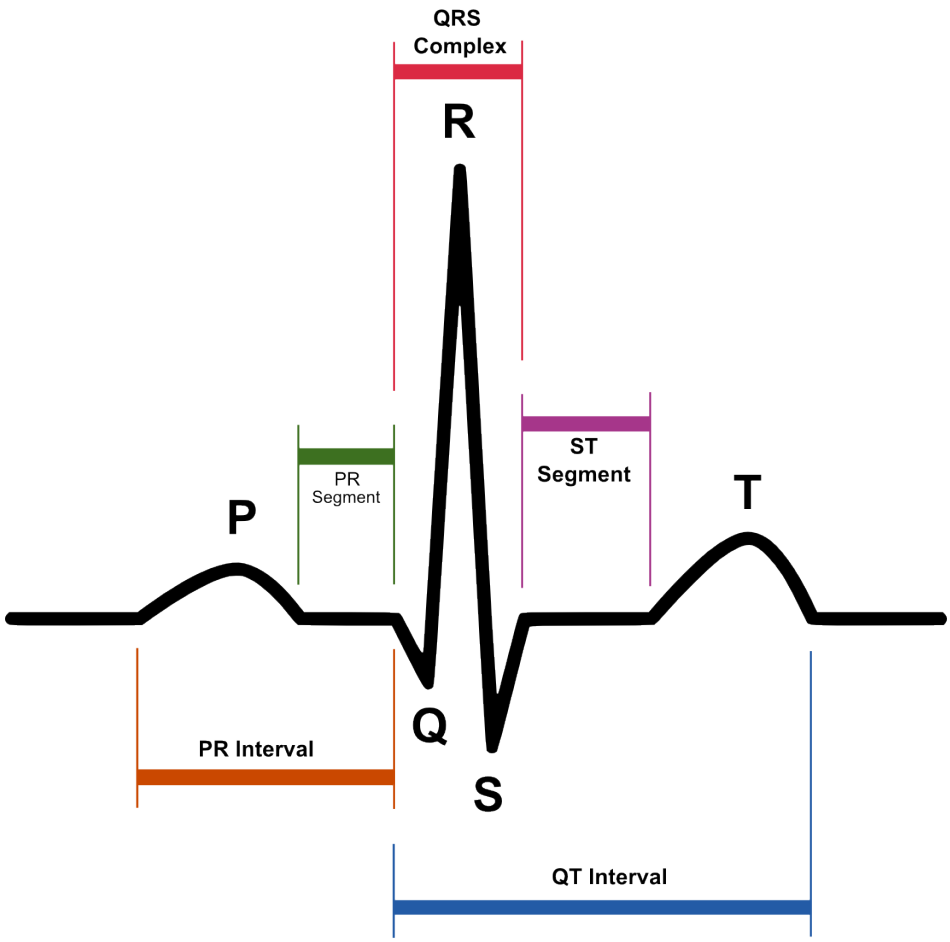

Figure 1.1: Schematic Diagram of Normal Sinus Rhythm[12]

The more people use these devices, and the more frequently the measurements are taken, the more manpower is required to interpret the signals. Due to the complexity of the signals, a well trained professional is required to interpret them. This is expensive. As a result, several signal processing techniques have been introduced to automatically analyze and categorize EKG signals. However, the noisy nature of these signals makes it difficult to identify the locations of the P, QRS, and T waves. Since the QRS complex contains substantial information regarding the functions and structures of the heart, detection of this complex in a signal is very crucial. The objective of this project is to develop an automated annotation system for EKG signals. This involves three subsystems:

1. Detecting the QRS complexes in a signal.

2. Categorizing the normal and abnormal cycles in a signal.

3. Specifying the types of arrhythmia of the abnormal cycles.

Many algorithms have been developed so as to detect the QRS complex as well as to detect arrhythmia. Some of the algorithms involve artificial neural networks, genetics algorithms, wavelet transforms, filter banks and heuristic methods. However, only a few developers made their source code available to the public for corroboration [11]. For those whose source code is available, an analysis has been made so as to compare their performances using the MIT/BIH database [3]. In this analysis, three algorithms are tested. The two algorithms are based on digital filters, Pan and Tompkins algorithm, and Hamilton and Tompkins algorithm, and the third one is based on phasor transform. According to this analysis, the Pan and Tompkins algorithm in [2] outperformed the rest of the algorithms. For this reason, we will compare our algorithm with the Pan and Tompkins algorithm stated in [2] using our data.

This report will focus on the two subsystems - QRS detection and beat classification. In Chapter 2 we will present some neural network background and notation. This will be followed by a discussion of neural network training procedures in Chapter 3. Chapter 4 will describe the data sets that will be used to train and test the automated QRS detector. Chapter 5 will describe the architecture of the QRS detector, and Chapter 6 will present the results of training and testing. In chapter 7, we describe the data for beat classification. Chapter 8 discusses the network and training for beat classification. Then, following the result of the beat classification system in chapter 9, the conclusion and a brief summary of future works will be give in chapter 10.

CHAPTER 2

BASIC NEURAL NETWORK CONCEPTS AND NOTATION

We will be using neural networks as part of our automated EKG annotation system. In this chapter we introduce some neural network concepts and notation that will be needed for the description of the annotation system. We will begin with the simplest building block, the neuron, and will build up to the multilayer network, which is the architecture we will use for the automated annotation system.

2.1 Single-Input Neuron

The basic building block of a neural network is the neuron, as shown in Figure 2.1. Let w, p and b denote the scalar weight, input and bias of a network, respectively. The weight multiplies the inputs, and the bias is added to the result. The net input, $n = wp + b$, then passes through the transfer function, f. Hence, the output of the neuron is given by $a = f(n)$.

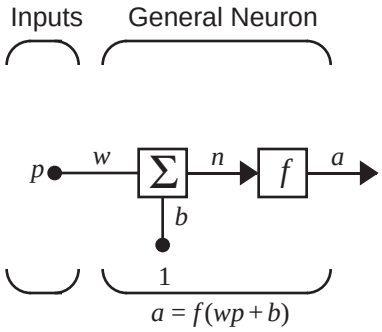

Figure 2.1: Single Neuron

2.2 Multiple-Input Neuron

We often have more than one input. Figure 2.2 depicts a neuron with R inputs. For each input p_i, there is a corresponding weight element, $w_{1,i}$, such that $n = w_{1,1} * p_1 +$

$w_{1,2} * p_2 + w_{1,3} * p_3 ... + w_{1,R} * p_R + b$. Here, the output is $a = f(\mathbf{W}\mathbf{p} + b)$, where

$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & . & . & . & w_{1,R} \end{bmatrix} \tag{2.1}$$

and

$$\mathbf{p} = \begin{bmatrix} p_1 \\ p_2 \\ . \\ . \\ . \\ p_R \end{bmatrix} \tag{2.2}$$

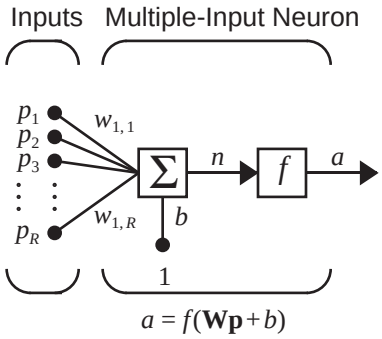

Figure 2.2: Multiple Input Neuron

2.3 A Layer of Multiple Neurons

Figure 2.3 shows a layer that contains S neurons and R inputs. Each input is weighted and connected to each neuron. The matrix notation of the layer operation is given by

$$\mathbf{a} = \mathbf{f}(\mathbf{W}\mathbf{p} + \mathbf{b}) \tag{2.3}$$

where,

$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & . & . & .w_{1,R} \\ w_{2,1} & w_{2,2} & . & . & .w_{2,R} \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ w_{S,1} & w_{S,2} & . & . & .w_{S,R} \end{bmatrix}, \mathbf{p} = \begin{bmatrix} p_1 \\ p_2 \\ . \\ . \\ . \\ p_R \end{bmatrix}, \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ . \\ . \\ . \\ b_S \end{bmatrix} \tag{2.4}$$

The following figure shows a layer of S neurons and R inputs. Notice that the transfer functions for each neuron are the same.

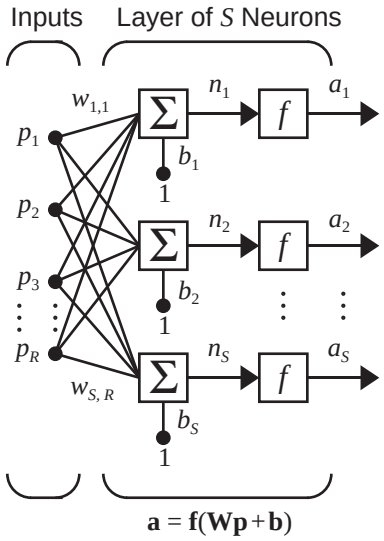

Figure 2.3: A Layer of Multiple Neurons

2.4 Multilayer Network

A three layer network is shown in Figure 2.4. In this network, there are S^1, S^2, and S^3 neurons in the first, second and third layer, respectively. In general, we denote the number of neurons in the M^{th} layer by S^M. In each of the layers, there is only one transfer function. For the first layer in this figure, the transfer function is f^1, the weight matrix is $\mathbf{W}^1$, and there are R inputs and S^1 neurons. The outputs of the first layer are inputs for the second layer, the outputs of the second layer are inputs for the third layer, and so forth. All but the last layer of the network are called Hidden Layers. The last layer is known as the Output Layer. A neural network that contains only forward connections is known as a multilayer network. In general, for a network of M layers, we have

$$\mathbf{a}^{m+1} = f^{m+1}(\mathbf{W}^{m+1} * \mathbf{a}^m + \mathbf{b}^{m+1}), m = 1, 2, 3, ...M-1 \tag{2.5}$$

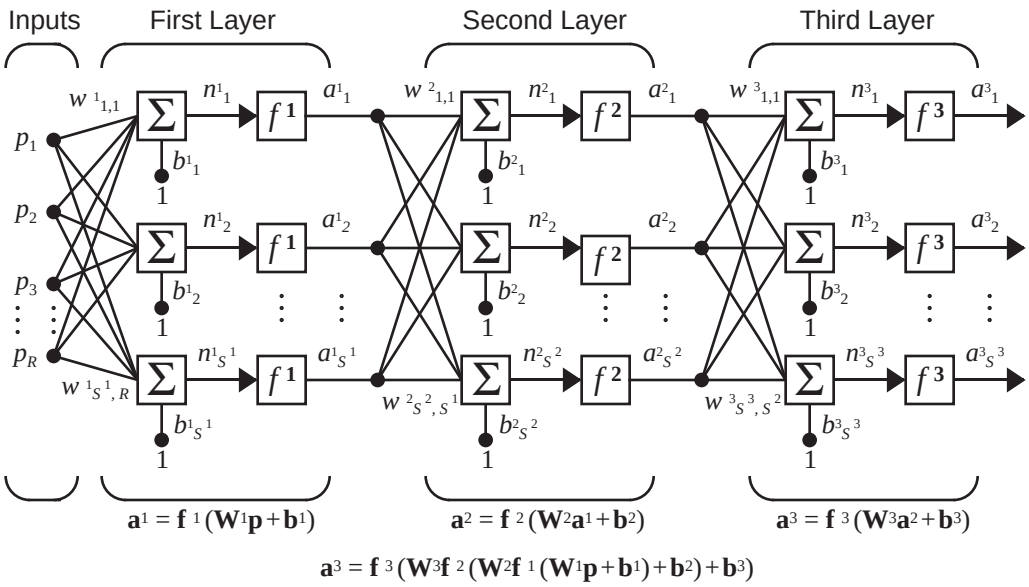

Figure 2.4: Three Layers Network

2.5 Transfer Function

In pattern recognition problems, the transfer function of the output layer is normally either a sigmoid function or a softmax function. The outputs of the softmax function range from 0 to 1 and sum to one. The form of softmax is given by

$$a_i = f(n_i) = e^{n_i} \div \sum_{j=1}^{S} e^{n_j} \tag{2.6}$$

In the hidden layers of multilayer networks, tansigmoid functions are common. The tansigmoid transfer function is given by:

$$f(n) = \frac{e^n - e^{-n}}{e^n + e^{-n}} \tag{2.7}$$

2.6 Focused Time Delay Network

When annotating the EKG, it is useful to consider a moving window of the signal. This requires a network that has memory. This can be done by adding delays to a multilayer network. The output of a delay is the input delayed by one time step, as shown in Figure 2.5.

A tapped delay line (TDL) is a set of delays cascaded together, as shown in Figure 2.6. In problems that require analysis of time-varying patterns, TDLs operate so that a history of the data can be presented to the network. This is illustrated in

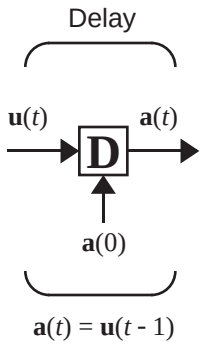

Figure 2.5: A Delay Block

Figure 2.7, which is the focused time delay network architecture we will use for QRS detection, with a TDL of length N_w at the input of the network.

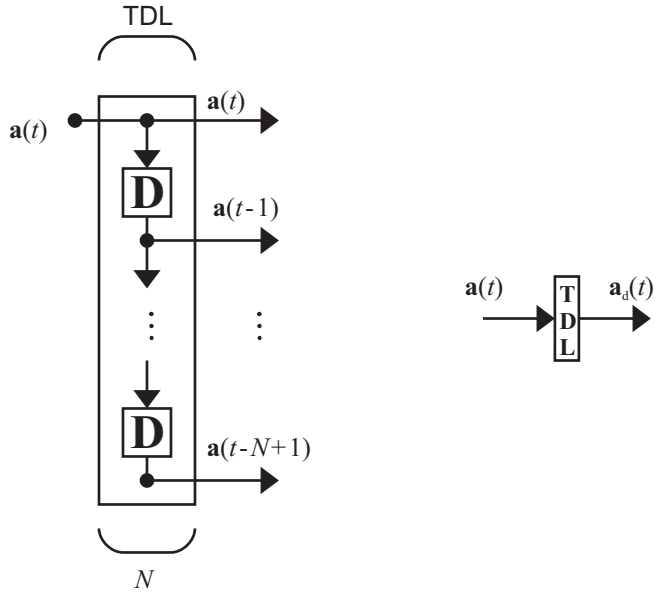

Figure 2.6: Tap Delay Line

2.6.1 Example

Suppose that we have one EKG sequence:

$$\{z(1), z(2), ..., z(Q)\} \tag{2.8}$$

and corresponding target sequence

$$\mathbf{t}(k) = \left\{ \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, ... \right\} \tag{2.9}$$

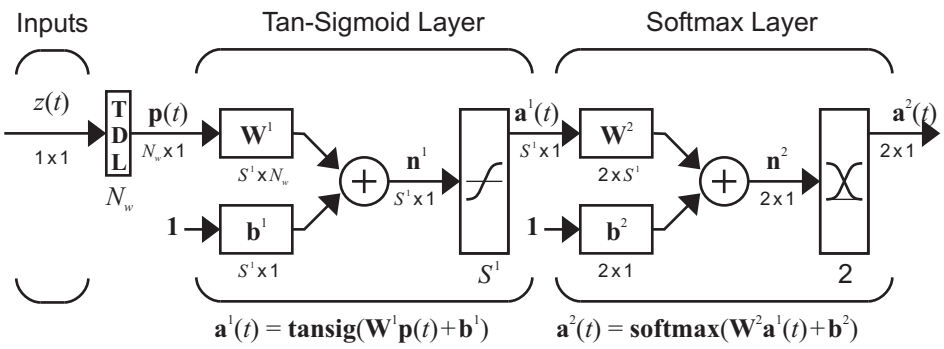

Figure 2.7: Focused Time Delay Network

where $\begin{bmatrix} 1 & 0 \end{bmatrix}^T$ in the target sequence represents the locations of an R wave. For simplicity, assume an R wave occurs every three time steps. Since, in this example, each cycle of the EKG would consist of 3 points, we would use a TDL of length 3.

It is possible to convert the dynamic network of Figure 2.7 into a static network by subdividing and stacking the input sequence. The TDL of Figure 2.7 creates a three dimensional input vector $\mathbf{p}$ from the scalar z. We can create a matrix $\mathbf{P}$ whose columns are the $\mathbf{p}$ vectors as follows:

$$\mathbf{P} = \begin{bmatrix} z(1) & z(2) & . & . & . & z(Q-2) \\ z(2) & z(3) & . & . & . & z(Q-1) \\ z(3) & z(4) & . & . & . & z(Q) \end{bmatrix} \tag{2.10}$$

We would select the corresponding targets to indicate when the R wave is in the center of the TDL, as follows:

$$\mathbf{T} = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 & . & . & . & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & . & . & . & 0 & 1 & 1 \end{bmatrix}$$

where $\begin{bmatrix} 1 & 0 \end{bmatrix}^T$ in the column of $\mathbf{T}$ would occur whenever an R wave is indicated at the time point $z(k)$ that is located in the center of the corresponding column of $\mathbf{P}$.

To state this in a more general way, let $Q_1 = Q - N_w + 1$, where Q is the length of a sequence and N_w is the length of the tapped delay line. Each column of the input is given by

$$\mathbf{P}(:, j) = z(j : j + N_w - 1)^T, \qquad \forall j = 1, 2, 3, ..., Q_1 \tag{2.11}$$

The corresponding target columns are given by

$$\mathbf{T}(:, j) = \mathbf{t}(j + \lfloor \frac{N_w}{2} \rfloor), \qquad \forall j = 1, 2, 3, ..., Q_1 \tag{2.12}$$

In order to determine an appropriate length for the tapped delay line in our problem, we scanned the entire data set looking for the largest interval containing a QRS complex. It was found that the largest number of data points between two consecutive QRS complexes is 175. Hence, we used a tapped delay line of length $N_w = 351$. The sequence length is $Q = 9000$, and we trained using 625 sequences at a time. The resulting input matrix size is 351×5406250 and target size is 2×5406250, where $625 \times (9000 - 351 + 1) = 5406250$. A discussion of why we used 625 sequences at a time is given later in this report.

2.7 Scaling the Network Input

In order for the Neural Network to extract the necessary features of the training data, data preprocessing is essential. Normalization is one part of the preprocessing. Here, we discuss the two common normalization techniques.

- **mapstd** is a transforms the data so that the mean and standard deviation are 0 and 1, respectively. The transformation is defined by

$$\mathbf{p}^n = (\mathbf{p} - \mathbf{p}^{mean})./\mathbf{p}^{std} \tag{2.13}$$

  where $\mathbf{p}^{mean}$ is the average of the input vectors $\mathbf{p}$, and $\mathbf{p}^{std}$ is the standard deviation of the input vector $\mathbf{p}$.

- **mapminmax** transforms the data so that the normalized inputs fall in the interval $[-1, 1]$. The transformation is defined by

$$\mathbf{p}^n = 2 * (\mathbf{p} - \mathbf{p}^{min})./(\mathbf{p}^{max} - \mathbf{p}^{min}) - 1 \tag{2.14}$$

  where $\mathbf{p}^{min}$ and $\mathbf{p}^{max}$ are the minimum and maximum of the input vector $\mathbf{p}$, respectively.

For the work presented in this report, we used the mapminmax preprocessing step.

Now that we have described the network architecture, and associated data preprocessing, we will discuss how the network is trained.

CHAPTER 3

TRAINING NEURAL NETWORKS

Changing the weights and biases of the network changes the network performance. We want to modify these parameters so that the network outputs match the target outputs as closely as possible. The techniques we use to modify these parameters are called Training Algorithms or Learning Rules. Depending on the nature of the problem, there are several learning rules to choose from. Section 3.3 discusses one of the most common algorithms, the steepest descent learning algorithm.

3.1 Supervised Learning

For supervised learning algorithms, the training data contains both network inputs and target network outputs. The network is trained so that the network outputs are close to the targets. A performance index is used to measure how close the network outputs are to the targets. During training, the weights and biases of the networks are adjusted to optimize the performance index.

3.2 Performance Index

A performance index is a quantitative measure of the network's performance. This performance index is small whenever the outputs match the targets, and large otherwise. Before we talk about the methods for optimizing the performance index, we will discuss the common performance indexes. The most common performance index in multilayer networks is sum square error. Suppose we have sets of inputs and their corresponding targets:

$$\{\mathbf{p}_1, \mathbf{t}_1\}, \{\mathbf{p}_2, \mathbf{t}_2\}, \ \ldots, \{\mathbf{p}_q, \mathbf{t}_q\} \tag{3.1}$$

The weighted sum square error performance index is given by

$$F(\mathbf{x}) = \sum_{j=1}^{Q} \sum_{i=1}^{S^M} w_{i,j}^{e} (t_{i,j} - a_{i,j})^2 \tag{3.2}$$

where $w^e_{i,j}$ is the error weighting, $\mathbf{x}$ is the vector of weights and biases, $t_{i,j}$ is the i^{th} element of the j^{th} target vector and $a_{i,j}$ is the i^{th} network output for the input $\mathbf{p}_j$. The error weighting is needed because the number of time points at which an R wave is located is a small percentage of the total number of samples in the EKG signal. It is possible to make the error very small by simply classifying every point as not an R wave. This would increase Type II errors, but the overall error rate would be small. To prevent this problem, we can weight Type II errors more than Type I errors. For example, in one of the target sets only 22,168 time steps out of 5,406,250 are R locations, which is only 0.41%. For this research we will use an error weighting of one for all data points where there is no R wave, and an error weighting of ρ for data points where an R wave is indicated. In a later section we will test to determine an optimal value for ρ, which we will refer to as the error weighting ratio.

Another common performance index for pattern recognition is cross entropy. The weighted cross entropy performance index is given by

$$F(\mathbf{x}) = -\sum_{j=1}^{Q}\sum_{i=1}^{S^M} w^e_{i,j}\mathbf{t}_{i,j} ln(\mathbf{a}_{i,j}/\mathbf{t}_{i,j}) \tag{3.3}$$

This is the performance index we will use in the remainder of this report.

3.3 Training Algorithm

After defining the performance index, the next task is to determine those weights and biases that minimize the performance index. There are several algorithms that can be used for performance optimization. These algorithms are iterative. In each iteration, we would like to move in a direction that reduces the performance index. For every k, we want to have $F(\mathbf{x}_{k+1}) \leq F(\mathbf{x}_k)$. A general iteration would be

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha\mathbf{p}_k \tag{3.4}$$

where α is the learning rate, and $\mathbf{p}_k$ is the search direction. One of the most common training algorithms is the steepest descent algorithm. In this algorithm, the search direction is the negative gradient direction. The learning rate, α, can be fixed, or it can be determined by minimizing the performance with respect to α by searching along the line $\mathbf{x}_k - \alpha\mathbf{g}_k$, where $\mathbf{g}_k$ is the gradient, which is given by

$$\nabla F = \left[\frac{\partial F}{\partial x_1} \quad \frac{\partial F}{\partial x_2} \quad . \quad . \quad . \quad \frac{\partial F}{\partial x_n}\right]^T \tag{3.5}$$

An alternative to the steepest descent, or gradient descent, algorithm is the conjugate gradient algorithm. Conjugate gradient algorithms use the gradient, but then make small adjustments to the search direction. The Scaled Conjugate Gradient (SCG) algorithm [1], in particular, has been very successful in training neural networks for pattern recognition problems. We will use the SCG algorithm for this work.

3.4 Gradient Calculation

In the case of a single layer linear network, the error is a linear function of the parameters. We can easily determine the derivative of the error with respect to the parameters. However, multi-layer networks with nonlinear transfer functions require an application of the chain rule, which is referred to as backpropagation. In this pattern recognition problem, we used two layers, with a tansigmoid transfer function (Eq. 2.7) in the first layer and a softmax (Eq. 2.6) transfer function in the second layer. The performance index is cross entropy, given in (Eq. 3.3).

The following steps describe how the gradient is computed. In the first step the input is propagated forward through the network and the network error is computed. In the second step sensitivities are backpropagated through the network. In the final step the gradient is computed by multiplying sensitivities times the layer inputs.

1. Propagate the input forward though the network

$$\mathbf{a}^0 = \mathbf{p} \tag{3.6}$$

$$\mathbf{a}^{m+1} = f^m(\mathbf{W}^{m+1}\mathbf{a}^m + \mathbf{b}^{m+1}) \qquad m = 0, 1, 2, ..., M, \tag{3.7}$$

where M is the total number of layers. Since we have only two layers, Eq. 3.7 becomes

$$\begin{aligned} \mathbf{a}^1 &= f^1(\mathbf{W}^1\mathbf{a}^0 + \mathbf{b}^1) \\ a_i^1 &= \frac{e^{n_i^1} - e^{-n_i^1}}{e^{n_i^1} + e^{-n_i^1}} \end{aligned} \tag{3.8}$$

$$\begin{aligned} \mathbf{a} = \mathbf{a}^2 &= f^2(\mathbf{W}^2\mathbf{a}^1 + \mathbf{b}^2) \\ a_i^2 &= f^2(n_i^2) = \frac{e^{n_i^2}}{\sum_{j=1}^{S^2} e^{n_j^2}}, \end{aligned} \tag{3.9}$$

2. Propagate the sensitivity backward through the network. We start from the last layer M.

$$\mathbf{s}^M = \frac{\partial F}{\partial \boldsymbol{n}^M} \tag{3.10}$$

Then the sensitivities for the hidden layers can be calculated by

$$\mathbf{s}^m = \dot{\boldsymbol{F}}^m(\boldsymbol{n}^m)(\boldsymbol{W}^{m+1})^T \boldsymbol{s}^{m+1} \tag{3.11}$$

where

$$\dot{\boldsymbol{F}}^m = \begin{bmatrix} \dot{f}^m(n_1^m) & 0 & \cdots & 0 \\ 0 & \dot{f}^m(n_2^m) & \cdots & 0 \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ 0 & 0 & \cdots & \dot{f}^m(n_{s^m}^m) \end{bmatrix} \tag{3.12}$$

and

$$\dot{f}^m(n_1^m) = \frac{\partial f^m(n_j^m)}{\partial n_j^m} \tag{3.13}$$

Since the elements in the target vector and the output vector have only two elements, and always sum to 1, Eq. 3.3 can be simplified to

$$F(x) = -\sum_{j=1}^{q}(w_j^e(t_j ln(a_j^2) + (1-t_j)ln(1-a_j^2))) \tag{3.14}$$

The sensitivity in the second layer is

$$\begin{aligned} \mathbf{s}_i^2 &= \frac{\partial F}{\partial \boldsymbol{n}_i^2} \\ &= w_i^e(\frac{-t_i}{a_i} + \frac{t_i}{1-a_i})\frac{\partial a_i}{\partial n_i^2} \\ &= w_i^e(\frac{-t_i}{a_i} + \frac{t_i}{1-a_i})\dot{f}^2(n_i^2) \end{aligned} \tag{3.15}$$

where

$$\dot{f}^2(n_i^2) = \frac{e^{n_i^2}\sum_{j=1}^{q}(e^{n_j^2}) - e^{2n_i^2}}{\sum_{k=1}^{q}\sum_{j=1}^{q} e^{n_j^2+n_k^2}} \tag{3.16}$$

The sensitivity in the first layer is

$$\mathbf{s}^1 = \dot{\boldsymbol{F}}^1(\boldsymbol{n}^1)(\boldsymbol{W}^2)^T \boldsymbol{s}^2 \tag{3.17}$$

where

$$\dot{\boldsymbol{F}}^1 = \begin{bmatrix} \dot{f^m}(n_1^1) & 0 & \cdots & 0 \\ 0 & \dot{f^m}(n_2^1) & \cdots & 0 \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ 0 & 0 & \cdots & \dot{f}^1(n_{s^1}^1) \end{bmatrix} \tag{3.18}$$

and

$$\dot{f}^1(n_i^1) = \frac{\partial f^1(n_i^1)}{\partial n_i^1} = 1 - (a_i^1)^2 \tag{3.19}$$

Using these sensitivities, the portions of the gradient associated with each weight and bias are

$$\mathbf{gW^2} = \mathbf{s}^2(\mathbf{a}^1)^T \tag{3.20}$$

$$\mathbf{gb^2} = \mathbf{s}^2 \tag{3.21}$$

$$\mathbf{gW^2} = \mathbf{s}^1\mathbf{p}^T \tag{3.22}$$

$$\mathbf{gW^2} = \mathbf{s}^1 \tag{3.23}$$

The steps above produce the gradient corresponding to a single input-target pair. To compute the total gradient for the entire training set, the individual gradients for each pair must be summed together. If weights are updated after each individual input-target pair, this is referred to as incremental training. If the weights are updated after the total gradient has been computed, this is referred to as batch training. In this work, we divided the data into several groups of sequences. The gradient was computed on a single group, and the weights were updated after each group was presented. The reason for this compromise between batch and incremental training was that the data set was too large to be loaded into memory at the same time. This will be discussed in a later section.

3.5 Stopping Criteria

Part of the training algorithm is determining when to stop. There are several techniques that can be used. One is to check the magnitude of the gradient. When a minimum of the performance is reached, the gradient will be zero. We will stop the training if the magnitude of the gradient is less than some predetermined level.

An additional method is called early stopping. By increasing the number of iterations of training, we are increasing the complexity of the resulting network. If training is stopped before the minimum is reached, then the network will be less likely to overfit the data. One early stopping method is called cross-validation, which uses a validation set to decide when to stop. The available data (after removing the test set, as will be described later) is divided into a training set and a validation set. The training set is used to compute gradients and to determine the weight update at each iteration. The validation set is an indicator of what is happening to the network function in between the training points, and its error is monitored during the training process. When the error on the validation set goes up for several iterations, the training is stopped, and the weights that produced the minimum error on the validation set are used as the final trained network weights.

We also set a maximum number of iterations, after which the training is stopped. If the weights do not converge after the maximum number of iterations, then we restart the training by taking the weights saved from the last training as the initial condition.

CHAPTER 4

DATA FOR QRS DETECTION

In this chapter we will describe the data set and will explain how the data was selected for training, validation and testing sets. Also, because of the size of the data set, we needed to divide it into subgroups for processing.

4.1 Description of the Data

The nominal EKG signal was recorded for 30 seconds at 300 samples/second, which gives us a sequence length of $Q = 30 \times 300 = 9000$ data points. There were 16,000 sequences presented for analysis. The sequences did not all have the same length. For example, there were 14,488 sequences with length greater than 7,000, and only 4,156 sequences with a length greater than 9,500. For training purposes, it was convenient to have sequences of equal length. There were 13,971 of these sequences with a length of at least 9000 points. Therefore, we selected those sequences for analysis, and called them the usable sequences.

The EKG signals in the data set have been annotated by physicians. These annotations will form the targets to be used when training neural networks to perform automated annotations. In the annotated signals, the waveforms did not always follow the classic pattern shown in Figure 1.1. In some cases, the R wave was a negative pulse instead of a positive pulse. We found that it was important to include both types of signals in any data set that we used to train the neural networks.

On average, there are 32 cardiac cycles in a 30 second sequence, and each cycle contains only one R-location. If at least 16 of the R waves of a sequence are negative, then we will label the sequence a Negative Sequence, otherwise it is said to be a Positive Sequence. Twenty percent of the usable data were negative sequences and the other eighty percent were positive sequences.

4.2 Training and Testing Data

It is important to hold aside a certain subset of the data during the training process. After the network has been trained, we will compute the errors that the trained network makes on this test set. The test set errors will then give us an indication of how the network will perform in the future; they are a measure of the generalization capability of the network. We divided the usable EKG data into two partitions - training/validation data and testing data. Both partitions must be representative of the entire data set. If the training/validation set does not cover all regions of the input space, it will be difficult for the neural network to generalize when tested on new sequences. The training/validation set usually contains 85% of the usable data. The remaining 15% will be used to test the network. We selected and isolated the testing data set so that it will not be used for any purpose other than testing the network after all the training is done. In order to cover all of the input space with both data sets, we performed some preliminary tests to determine the types of sequences we have.

Roughly 20% of the usable data are negative sequences (a total of 2,373 sequences) and the other 80% are positive (a total of 9502 sequences). From the usable sequences, we selected 11,875 sequences for training/validation (85% of the usable data) and 2,096 sequences for testing (15%of the usable data). The data were selected randomly, with the restriction that 20% of the training/validation data were negative and the remaining 80% were positive. Similarly, 20% of the testing data (2,096 sequences) were negative and the remaining 80% were positive.

4.2.1 Group Formulation

One of the challenges in dealing with big data is memory. It was not possible to fit the entire data set in memory at the same time, so we needed to divide the training set into groups. In order to decide the number of sequences to be included in a group, it was important to test the number of sequences, N_s, that can be accommodated in memory. In MATLAB, if an array becomes too large to fit in memory, the system will automatically use virtual memory, which significantly slows down computations. In our tests, we increased the size of the data sets, and noted when computation times began to increase dramatically.

Table 4.1 shows training times for two iterations as the number of sequences in the data set is varied in the interval $N_s = [300, 760]$. We see that there is a linear

relationship between the number of sequences and the training time until $N_s = 700$. However, the iteration time for more than 700 sequences increases dramatically. This would imply that 700 sequences would fit adequately in the available memory. A further test was made to verify the consistency of this conclusion, and the results are shown in Table 4.2. We can see that the times are consistent for 700 sequences.

N_s	First Iteration Time (sec)	Second Iteration Time (sec)
300	32.199	33.633
350	57.954	52.572
400	56.175	62.276
450	55.037	46.457
500	62.868	50.466
550	55.754	65.645
600	70.293	59.468
650	77.563	65.848
700	76.455	74.615
750	243.08	264.9
750	106.064	45.474
760	83.647	86.018

Table 4.1: Iteration Times vs. Number of Sequences

Trails	Time for the First Iteration	Time for the Second Iteration
1	76.455	74.615
2	77.258	73.621
3	76.516	75.091
4	76.404	74.484
5	74.523	73.546

Table 4.2: Time (sec) Required for Two Iterations with 700 Sequences

Based on this result, we can determine the number of groups (N_g) and the number of sequences in a group (N_s). If we use exactly 700 sequences per group, this will reduce the total number of sequences used from the available 11,875 to 11,200. If we want to use all of the sequences, we could choose 625 sequences per group ($N_s = 625$), which will produce 19 groups ($N_g = 19$).

It was also important to test whether a completely random grouping or a proportional grouping produces better training. In a completely random grouping, all the 625 sequences in a group are selected randomly from the 11,875 training sets. The distribution of positive and negative sequences would be different from one group to the other. On the other hand, with proportional grouping, we randomly select 20% of the 625 sequences in a group from the 2,373 negative sequences and randomly select the rest from the 9,502 positive sequences. In this way, each group has the same proportions of positive and negative sequences as in the total data set. We found that the proportional grouping produced the best training results.

CHAPTER 5

LOCATION OF QRS COMPLEX

After the focused time delay neural network of Figure 2.7 has been trained, we need to do some processing on the network output to determine the identified R wave locations. The network is supposed to produce an output of $\begin{bmatrix} 1 & 0 \end{bmatrix}^T$ at the location of an R wave, but this will not be exactly true. Figure 5.1 shows a small segment of an EKG signal, and the corresponding network outputs after training. We can see that the outputs do not go exactly to 1 and 0. The first output reaches approximately 0.8 in this case. We have to make a decision as to how close to 1 the first network output should be in order to be considered as an R wave indication. We will set a detection threshold d_t, and wherever the first network output is larger than d_t will be considered an R wave indication. (A later section of this report will describe tests to determine an appropriate value for d_t.) In addition, we only want to have one indication in each cardiac cycle. To ensure this, if more than one indication occurred within 50 samples (167 ms), we selected only the point with the highest network output to be the single indication.

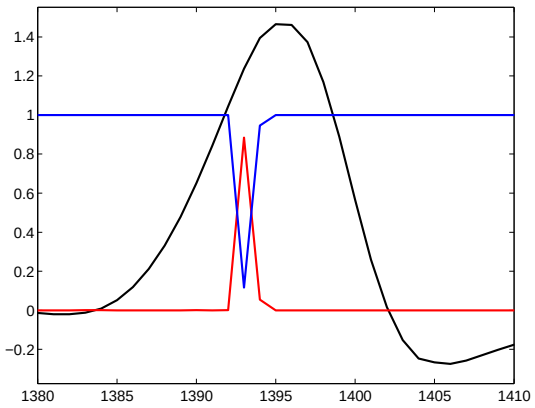

Figure 5.1: EKG signal segment and network output

5.1 Committee of Networks

Each time a network is trained, the training starts from a different initial set of weights and biases. In addition, the data is divided into different random training and validation sets each time a network is trained. In this way, it is possible to train a number of different networks on the same overall data set. After training, these networks can be combined together to form a committee of networks, which can produce a joint decision. This can be done by averaging the outputs of the committee members before applying the threshold test, or by taking a vote of the members after the threshold has been applied. If we have N_n networks, and if at least $N_n/2$ of them indicate an R wave, then we accept that indication. Since it is possible that different committee members might indicate slightly different locations for the R wave, an algorithm needs to be designed to combine votes when the indicated R wave locations are close to each other. Such an algorithm is described below.

5.1.1 Committee Formulation Algorithm

The idea of the following algorithm is that whenever any committee member indicates an R wave, the other committee members are checked to see if they have an indication within a detection width d_w of the original indication. If any of them do, then again the remaining members are checked for indications within d_w of the last indication. This continues until no other indications are found within the detection width. At this point, if at least $N_n/2$ have indications that are close, the committee indicates an R wave at a location that is an average of the member locations.

Algorithm for Committee Vote:

We denote the output of the second layer of the k^{th} network by ${}_k a_1^2(t)$. Hence, we can define the indicator output by

$$y^{(k)}(t) = \begin{cases} 1 & if\ {}_k a_1^2(t) \geq d_t \\ 0 & else \end{cases}, \tag{5.1}$$

The sum of the indicator outputs of all the networks is given by

$$y^{tot}(t) = \sum_{i=1}^{k} y^{(i)}(t), \tag{5.2}$$

Let Γ_y be an ordered set of indexes such that

$\Gamma_y = \underset{(t)}{find}\ \{y^{tot}(t) \neq 0\}$

$j = \Gamma_y(1)$.
while $j \in \Gamma_y$
    $\Gamma^{tot} = \{j\}$
    $k = j$
    $go = True$
    while go
        $\Gamma = [k+1 : k+d_w]$
        $\Gamma' = \Gamma \cap \Gamma_y$
        If $\Gamma_k \neq \emptyset$
            $\Gamma^{tot} = \Gamma^{tot} \cup \Gamma'$
            $k = max\{\Gamma'\}$
        else
            $go = False$
        end if
    end while

    $N_n^{'} = \sum_{l \in \Gamma^{tot}} y^{tot}(l)$
    If $N_n^{'} \geq \frac{N_n}{2}$
        $w_{ave} = \frac{\sum_{l \epsilon (\Gamma^{tot})} y^{tot}(l) * l}{N_n^{'}}$
        $y_c^{tot}(round(w_{ave})) = 1$
        $j =$ next value in Γ_y greater than k
    end if
end while

It is easier to explain this concept using an example. Suppose that we have three networks. Each network might suggest existence of a pattern at a specific location. For simplicity, we assign patterns on the interval $t \in [1, 10]$. Let the output from the three networks be given as follows.

$$ {}_1a_1^2(t) = \begin{bmatrix} 0.01 & 0.02 & 0.25 & 0.06 & 0.01 & 0.02 & 0.02 & 0.12 & 0.11 & 0.01 \end{bmatrix} \tag{5.3} $$

$$ {}_2a_1^2(t) = \begin{bmatrix} 0.01 & 0.02 & 0.05 & 0.6 & 0.01 & 0.02 & 0.02 & 0.12 & 0.11 & 0.01 \end{bmatrix} \tag{5.4} $$

$$ {}_3a_1^2(t) = \begin{bmatrix} 0.01 & 0.02 & 0.05 & 0.06 & 0.01 & 0.60 & 0.02 & 0.01 & 0.11 & 1.00 \end{bmatrix} \tag{5.5} $$

The indicator output selects those elements of the output that are above the given detection threshold d_t. The indicator output with a threshold of $d_t = 0.15$ is

$$y^{(1)}(t) = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.6}$$

$$y^{(2)}(t) = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.7}$$

$$y^{(3)}(t) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.8}$$

Now we have

$$y^{tot}(t) = \sum_{i=1}^{k} y^{(i)}(t) = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.9}$$

and the ordered set of indexes is

$$\Gamma_y = \underset{(t)}{find} \{y^{tot}(t) \neq 0\} = \begin{bmatrix} 3 & 4 & 6 & 10 \end{bmatrix} \tag{5.10}$$

Now we start with $\Gamma_y(1) = 3$. We search for R location within the detection width distance of, say $d_w = 2$. On the first search we will find the fourth and the sixth data points. Hence $\Gamma^{'} = \{4, 6\}$ and $\Gamma^{tot} = \{3, 4, 6\}$. Then we take the maximum element which is $max\{3, 4, 6\} = 6$ and we start another search beginning at 6. Since there is no R-location within the detection width distance, we take the sum of R-locations in the set Γ^{tot}. We count the total number of votes, which is 3, so we can have an R-location suggested by the majority.

$$N_n^{'} = \sum_{l \in \Gamma^{tot}} y^{tot}(l) = 3 \geq \frac{N_n}{2} = \frac{3}{2} \tag{5.11}$$

The average location is calculated by

$$w_{ave} = \frac{\sum\limits_{l \epsilon (\Gamma^{tot})} y^{tot}(l) * l}{N_n^{'}} = \frac{3 + 4 + 6}{3} = 4.33 \approx 4 \tag{5.12}$$

Therefore we have an indicated R wave at 4. That is

$$y_c^{tot}(round(w_{ave})) = y_c^{tot}(4) = 1 \tag{5.13}$$

We then restart the next search at the next element of Γ_y greater than 6, which is 10. The process proceeds until we cover all the elements of Γ_y.

5.2 Definition of Errors

In order to assess the performance of the automated annotation system, we need to define appropriate measures of quality. A standard measure would be error rate. The difficulty with using this metric is that errors in R wave location can occur at every time point in the EKG sequence. However, on average, only one point in 250 will actually be an R wave location. (For example, there are 22,168 points indicated as R-locations from 5,406,250 data points in one of the 19 training groups.) An algorithm could be 99.7% accurate by simply saying that there are no R waves in a sequence. We will report error rate (ER), but we will also use the following performance measures: False Discovery Rate (FDR), which is a measure of how often the network detects an R wave when it is not there, Miss Rate (MR), which measures how often the network does not detect an R wave when it is there, and Relative Error Rate (RER), which is a ratio of the number of errors to the number of R wave locations and false detections of R waves. The RER will be much larger than the ER, which divides by the total number of time points. Before defining these measures precisely, we first define some key variables in Table 5.1.

Symbol	Definition
P	# of R wave locations in the data set
N	# of time points with no R wave
FP	# of false positives, Type I errors (false indications of R wave)
FN	# of false negatives, Type II errors (R wave without indication)
TN	# of true negatives (no R wave without indication)
TP	# of true positives (correct indications of R wave)

Table 5.1: Symbol Definitions

We can now define ER, FDR, MR and RER in Table 5.2.

There is one other aspect of error calculation to be addressed, and that is precision. It may be that the network indication of an R location is only one time step (3.3 miliseconds) away from the physician indication of the location. On the other hand, the network may have no R wave indication over a compete cardiac cycle (beat). These two situations should be treated differently. If the network indicates an R wave within a detection width d_w, then we will say that the network indication is correct. If it is greater than d_w, we will say that it is wrong. We will report errors

Symbol	Definition	Equation
ER	Error Rate	$\frac{FP+FN}{P+N}$
FDR	False Discovery Rate	$\frac{FP}{FP+TP}$
MR	Miss Rate	$\frac{FN}{P}$
RER	Relative Error Rate	$\frac{FP+FN}{P+FP}$

Table 5.2: Performance Measures QRS Detection

using two different detection widths - 3 and 50. Correct results with $d_w = 3$, which is one hundredth of a second, will indicate that the network is producing precise R wave locations. Correct results with $d_w = 50$ will indicate that at least the network is able to detect the beat, although the precise location of the R wave within the beat may be off. Because a number of the EKG sequences in the database have inconsistent physician indications of R wave locations, as will be described later, using both 3 and 50 for d_w will give us a better understanding of the operation of the network.

5.3 Setting Algorithm Parameters

5.3.1 Number of Neurons, Detection Threshold and Error Ratio

Before testing the R wave location system, we need to determine appropriate settings for the following parameters: 1) error weighting on FN errors, ρ; 2) the detection threshold, d_t; and 3) the number of neurons in the hidden layer, S^1. To find these values, we ran some preliminary tests with a subset of the data (20 sequences). The results are shown in Table 5.3. The best results were obtained with $d_t = 0.1$, $\rho = 5$ and $S^1 = 30$. The light gray row depicts the combination that causes the smallest relative error rate.

We performed a further refinement on the number of hidden neurons. We set $d_t = 0.1$ and $\rho = 5$ and then adjusted S^1 from 10 to 65. We also increased the number of test sequences from 20 to 200, to be sure that we were not overfitting. The results are shown Table 5.4. In this preliminary test, reasonable results are obtained when $S^1 = 30$ or more. A network with $S^1 = 40$ has a total of 14,162 parameters (weights and biases), and the full data set (19 groups) we will use to train the final network has 421,192 target points, so we will not overfit during the full training. We

ρ	d_t	S^1	FN	FP	Sum	FNR	FDR	ER	RER
5	0.1	10	43	41	84	0.0537	0.0513	4.76e-4	0.0998
5	0.1	20	31	28	59	0.0393	0.0356	3.34e-4	0.0722
5	0.1	30	16	23	39	0.0207	0.0294	2.21e-4	0.0489
5	0.05	10	65	33	98	0.0790	0.0417	5.55e-4	0.1145
5	0.05	20	71	17	88	0.0856	0.0219	4.98e-4	0.1040
5	0.05	30	41	15	56	0.0513	0.0194	3.17e-4	0.0688
5	0.025	10	94	25	119	0.1103	0.0319	6.74e-4	0.1360
5	0.025	20	152	13	165	0.1670	0.0169	9.35e-4	0.1788
5	0.025	30	90	12	102	0.1061	0.0156	5.78e-4	0.1186
10	0.1	10	43	23	66	0.0537	0.0294	3.74e-4	0.0801
10	0.1	20	37	27	64	0.0465	0.0344	3.62e-4	0.0779
10	0.1	30	24	16	40	0.0307	0.0207	2.27e-4	0.0501
10	0.05	10	76	17	93	0.0911	0.0219	5.27e-4	0.1093
10	0.05	20	61	18	79	0.0745	0.0232	4.47e-4	0.0944
10	0.05	30	41	12	53	0.0513	0.0156	3.00e-4	0.0654
10	0.025	10	123	11	134	0.1396	0.0143	7.59e-4	0.1502
10	0.025	20	134	11	145	0.1502	0.0143	8.21e-4	0.1606
10	0.025	30	72	8	80	0.0867	0.0104	4.53e-4	0.0955

Table 5.3: Optimization of d_t, S^1 and ρ

will use $S^1 = 40$ for the full test. After analyzing the results, we will determine if the data set can be better fit with more neurons.

S^1	FN	FP	Sum	FNR	FDR	ER	RER
10	529	277	806	0.124383	0.069233	0.000913	0.177925
15	531	202	733	0.124794	0.051452	0.00083	0.16446
20	565	205	770	0.131732	0.052176	0.000872	0.17134
25	499	201	700	0.118162	0.051210	0.000793	0.158228
30	451	223	674	0.108024	0.056499	0.000763	0.153251
35	420	262	682	0.101351	0.06573	0.000773	0.154788
40	486	258	744	0.115439	0.064792	0.000843	0.166517
45	479	183	662	0.113966	0.046839	0.00075	0.150935
50	475	185	660	0.113122	0.047327	0.000748	0.150547
55	473	168	641	0.1127	0.043165	0.000726	0.14685
60	462	215	677	0.110368	0.054582	0.000767	0.153829
65	405	248	653	0.098087	0.062437	0.00074	0.1491889

Table 5.4: Optimization of S^1

5.3.2 Number of Iterations per Cycle

Because the full data set is too large to fit in memory, we divided the data into 19 groups, as described earlier. We will train on each group for a certain number of iterations, and then switch to the next group, until we have completed the cycle of all groups. After that, we will repeat the cycle. In the complete training/validation set, the number of groups is $N_g = 19$, and the number of sequences per group is $N_s = 625$. The number of iterations in cycle k will be indicated as $N_i(k)$, and the number of cycles will be indicated as N_c. The choice of the number of iterations per cycle is a trade-off between efficiency of computation and potential for overfitting and slowed training. It is more efficient to leave a group in memory for more iterations, rather than switching groups in and out of memory every few iterations. On the other hand, if we train too long on one group, the network may overtrain on the specific characteristics of that group, and have to relearn general characteristics when the next group is presented, causing some oscillation in training and slower convergence.

As a preliminary experiment, we used $N_g = 2$, $N_s = 300$ and $S^1 = 40$. We tested for $N_i = \{2, 3, 4, ..., 39\}$ assuming that $N_i * N_c \approx 100$, so that the total number of

iterations in each test is approximately the same (with the restriction that N_c must be an integer). The results of this experiment are shown in Table 5.5. The smallest error occurred for $N_i = 39$. This means it is possible to perform 39 iterations on each group before going to the next group and still achieve a better overall error. The total number of iterations in each case shown in Table 5.5 is approximately the same, but we get a bigger reduction in error if we perform 39 iterations on each group before switching.

We have tested the overhead associated with switching groups into memory. We found that it is a very small percentage of the time required to perform 40 iterations on a group with 625 sequences. For that reason, we decided to use $N_i = 40$ for the initial training cycles. After the first 5 cycles, after which the training error is quite small, we reduced N_i to 10.

5.3.3 Detection Threshold

We made a further refinement of the detection threshold after the network with $S^1 = 40$ had been completely trained. As described earlier, after the network is trained, the first network output is passed through a detection threshold d_t to determine if the network output is large enough to indicate the existence of an R wave. Ideally, the first output should equal 1, if an R wave exists. However, in practice an appropriate threshold might be much lower. If we make d_t too large, we will miss some R waves, causing more False Negative, or Type II errors. If we make d_t too small, we will indicate R waves where they do not exist, causing more False Positive, or Type I errors. We want to set the threshold to minimize the total number of errors. Figure 5.2 shows a plot of the number of Type I, Type II and total errors as d_t is adjusted over the interval $[0.05, 0.4]$ for a trained network. The value that minimizes the total error is approximately $d_t = 0.15$. This is the value that was used for the remaining tests.

5.4 Summary

To summarize, the training/validation data set of 11,875 sequences was normalized using the mapstd preprocessing function, reformed with a tapped delay line of length $N_w = 351$, and divided into $N_g = 19$ groups of $N_s = 625$ sequences each. Each group has a $351 \times 5,406,250$ input matrix and a $2 \times 5,406,250$ target matrix. A two-layer focused time delay network, as in Figure 2.7, with $S^1 = 40$ tansigmoid neurons in the hidden layer and $S^2 = 2$ softmax neurons in the output layer was trained using the

N_i	$N_i * N_c$	**Performance**
20	100	0.009521
21	105	0.009579
22	110	0.009329
23	92	0.009247
24	96	0.009366
25	100	0.008946
26	104	0.008945
27	108	0.008964
28	112	0.00845
29	87	0.009299
30	90	0.009155
31	93	0.008772
32	96	0.008739
33	99	0.008863
34	102	0.008608
35	105	0.008572
36	108	0.008361
37	111	0.008197
38	114	0.008062
39	117	0.007929

Table 5.5: Comparison for Number of Iterations

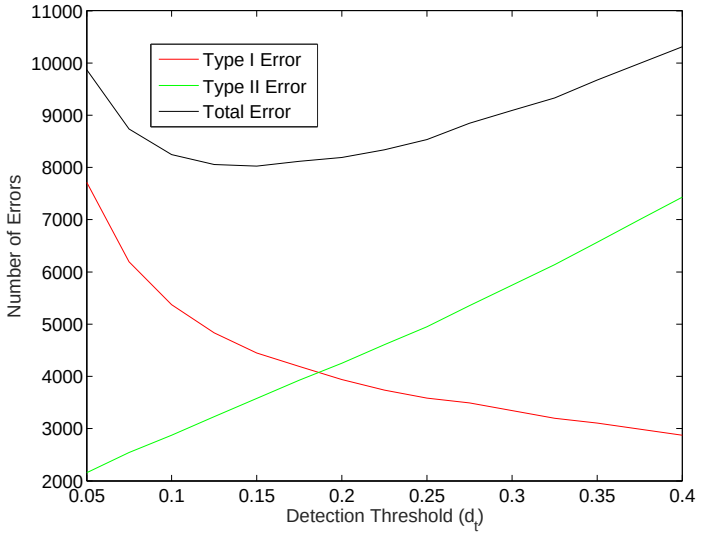

Figure 5.2: Detection Threshold (d_t) Optimization

scaled conjugate gradient training algorithm to minimize the weighted cross entropy performance function. The performance weighting for the $\begin{bmatrix} 1 & 0 \end{bmatrix}^T$ targets (indicating an R wave) was $\rho = 5$, and it was equal to 1 for the remaining cases.

The training was performed for $N_i = 40$ iterations per cycle for the first 5 cycles. This means there were a total of $N_i \times N_g \times 10 = 40 \times 19 \times 5 = 3,800$ iterations in the first 5 cycles. During these first 5 cycles, 15% of the data were randomly selected for a validation set to be used for early stopping. After the first 5 cycles of training, the training continued with $N_i = 10$ iterations per cycle for 135 additional cycles. The total number of iterations in the training was $3,800 + 10 \times 19 \times 135 = 29,450$ iterations. After the first 5 cycles, all of the training/validation data was used for training. After training was completed, the first element of the network output was checked against a detection threshold of $d_t = 0.15$ to produce an R wave indication.

We currently have trained three different networks using the above procedures. Each network was trained using a different set of initial weights, and with different random selections of training and validation sets during the first 5 training cycles. The three networks were combined to test the committee concept, although this is a small number of networks. After an analysis of the results and a check of the validity of the data set, additional networks will be trained.

Each of the networks was trained using the Neural Network Toolbox for MATLAB on a Linux server with 24 Intel Xenon processors using 2.80 GHz clocks. The server had a 47.1 GB RAM with 32 GB swap RAM. To perform the total of 29,450 iterations

on a single network required 423.28 hours.

CHAPTER 6

RESULTS FOR QRS DETECTION

Table 6.1 summarizes the training/validation set errors for the three networks and the committee when the detection width, d_w, is set to 3 and 50. As mentioned earlier, correct results with $d_w = 3$, which is one hundredth of a second, indicate that the network is producing precise R wave locations (relative to the physician annotations). Correct results with $d_w = 50$ indicate that at least the network is able to detect the beat, although the precise location of the R wave within the beat may be off. As we will see later, this seems to be caused mainly by inconsistencies in the physician annotations.

There are a few things we can notice from the summary. First, the False Discovery Rate and the Miss Rate are almost equal in all three individual networks. This means that the networks are not biased in their decisions. By using the error ratio of $\rho = 5$, we are weighting the false negative errors enough to balance the results, even though there are many more opportunities for false positive errors. Second, with $d_w = 50$, we remove approximately 60% of the errors, when compared to $d_w = 3$. (RER goes from 11% to 4.2%.) This means that most of the errors occurred when the network annotation was offset from the physician annotation, although they both came within the same beat. We will analyze these errors in a later section. It is our impression that the physicians did not always indicate the R wave location at the same location on the signal.

Nets	$d_w = 50$				$d_w = 3$			
	FDR	MR	RER	ER	FDR	MR	RER	ER
Net1	0.0215	0.0236	0.0442	1.91e-4	0.0559	0.0569	0.1068	4.95e-4
Net2	0.0204	0.0224	0.0419	1.81e-4	0.0553	0.0562	0.1057	4.89e-4
Net3	0.0207	0.0227	0.0424	1.83e-4	0.0549	0.0559	0.1049	4.85e-4
Com	0.0201	0.0220	0.0412	1.78e-4	0.0421	0.0787	0.1146	5.36e-4

Table 6.1: Training Error Summary for the Three Networks and the Committee

Figure 6.1 shows the convergence in training for one of the networks, with $d_w = 3$. The first point represents the first 5 cycles of training (with 40 iterations and 19 groups per cycle), and the remaining points represent the final 135 cycles (with 10 iterations and 19 groups per cycle). Figure 6.2 represents the same training process, but with $d_w = 50$. Even though we are switching between groups during each cycle, we can see that the overall error (over the entire training/validation set) is trending lower. It appears that we can continue to improve the performance with additional training cycles. We will continue to train longer, and to potentially use more neurons and more layers, but first we want to analyze the current errors, to see if they are caused by a failure of the network to correctly identify patterns, or by problems in the data set. We will also train additional networks, as three networks is a very small size for the committee. At this point, we have merely verified the committee algorithm, but do not have enough networks to determine if the committee can significantly improve performance.

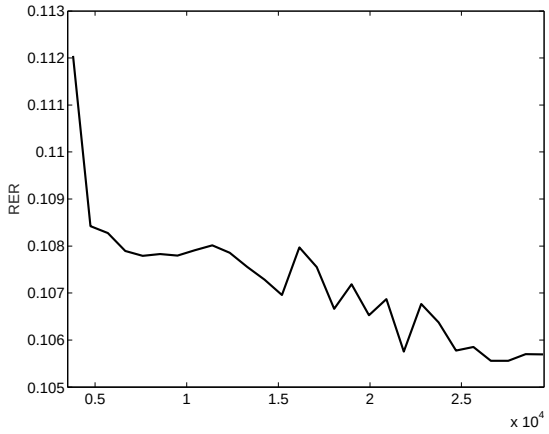

Figure 6.1: RER Versus Iteration Number on One Network with $d_t = 3$

Table 6.2 shows the performance of the networks on the test set. Recall that the test set contained 2,096 sequences that were randomly selected from the original 13,971 usable sequences in the full data set (15%), with the requirement that 20% of the sequences be negative. The test set was not used in any way to train the neural networks or to set any parameters (e.g., ρ, d_t, S^1, etc.). The errors on the test set are very similar to the training set errors. This means that the errors we obtained on the training set are reliable, and we would expect to see similar errors on any new data that would be collected in the future.

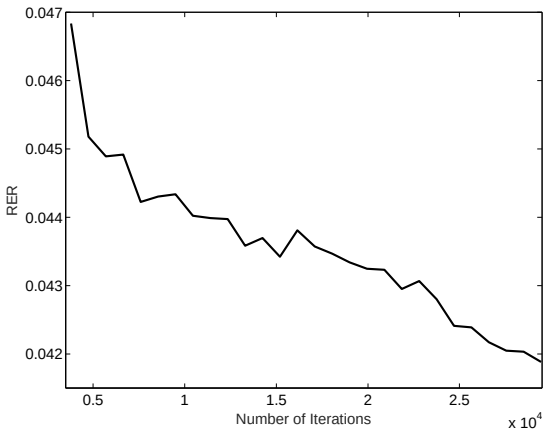

Figure 6.2: RER Versus Iteration Number on One Network with $d_t = 50$

Method	$d_w = 50$				$d_w = 3$			
	FDR	MR	RER	ER	FDR	MR	RER	ER
Net1	0.0206	0.0262	0.0458	1.97e-4	0.0570	0.0615	0.1119	5.17e-4
Net2	0.0220	0.0264	0.0473	2.04e-4	0.0575	0.0610	0.1119	5.17e-4
Net3	0.0220	0.0259	0.0468	2.01e-4	0.0580	0.0608	0.1121	5.18e-4
Com	0.0225	0.0232	0.0447	1.92e-4	0.0574	0.0572	0.1084	4.99e-4

Table 6.2: Testing Error Summary for the Three Networks and the Committee

6.1 Error Analysis

Most of the errors occur in only a few of the sequences in the data set. Of the total of 11,875 sequences in the training/validation data set, 6,135 sequences had no errors at all. Only 10% of the sequences contribute 68% of the total error. This is illustrated in Figure 6.3, which shows the number of sequences that produce a specific number of errors. We can see that over 6,000 sequences have no error, and that the majority of the errors are concentrated in a few sequences.

In general, the network performed better on positive sequences than on negative sequences (see the definition of positive and negative sequences in Section 4.1. The positive sequences, which represent 80% of the data, contribute only 41.48% of the errors, while the negative sequences, which represent only 20% of the data, contribute 58.52% of the errors. The network located the R wave with no error in 5,149 of the positive sequences but only in 986 of the negative sequences. Figure 6.4 shows examples of positive and negative sequences in which there were no errors in the

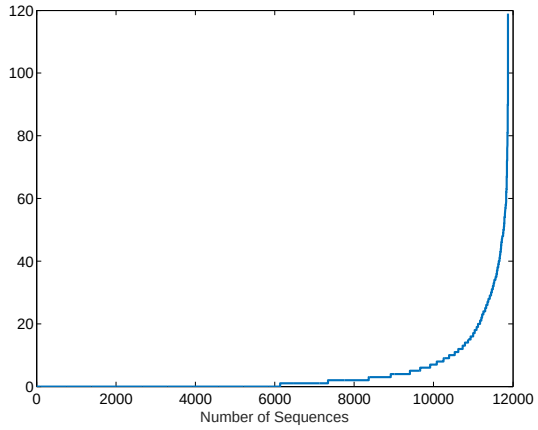

Figure 6.3: Number of Sequences Versus Number of Errors

location of the R waves. (These are short segments of two of the 6,135 sequences with no errors.) The black spikes indicate the physician annotation of the R wave location, which for these sequences corresponded exactly to the network annotation.

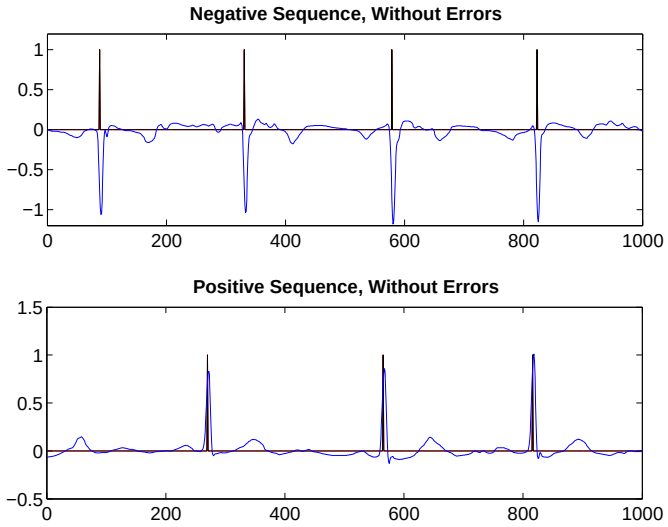

Figure 6.4: Negative and Positive Sequences with no Error

In the following sections we will investigate some of the types of errors.

6.1.1 Inconsistent Physician Annotations

As mentioned above, 60% of the errors are removed by changing the detection width from $d_w = 3$ to $d_w = 50$. This means that, in 60% of the errors, the network is able to detect the cardiac cycle, but is not able to detect the location of the R wave within

10 ms (three samples) of the physician annotation. After analyzing many sequences in which there were errors for $d_w = 3$ but not for $d_w = 50$, we found that the main cause was a seeming inconsistency in the physician annotations. An example of inconsistency within the same sequence is shown in Figure 6.5. The red line represents the physicians annotated R wave location (target). The green line represents the raw output of the neural network (first neuron). The black line represents the annotated R wave location determined by the network (output larger than the detection threshold d_t).

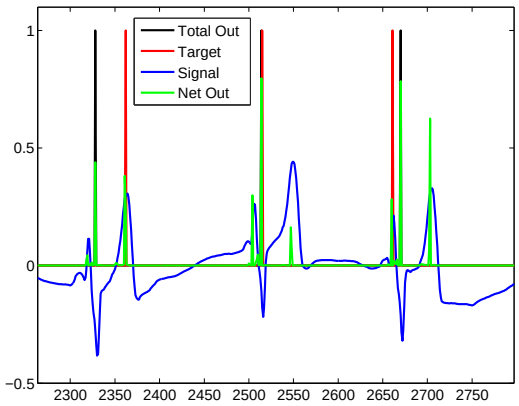

Figure 6.5: Intra-sequence Inconsistency of Annotation

Three different cardiac cycles are shown. In the first cycle, the physician indicates the R wave as located near the peak of the final rising wave in the cycle. In the second cycle, the physician annotation of the R wave occurs at the first negative wave of the cycle. (The total network output matches the target in this case, producing no error.) In the third cycle, the physician annotation of the R wave occurs near the peak of the first positive wave of the cycle. To the untrained observer, these three annotations do not seem to be consistent, and the network is confused, as a result.

Notice that the neural network produces some output (green line) at all of the physician-annotated locations. However, if a network has more than one output over threshold within 50 samples (0.167 sec), then the largest output is selected as the indicated R wave location. This is to eliminate multiple indications within a cardiac cycle. Figure 6.6 is an expanded view of the last cardiac cycle in Figure 6.5. From this expanded figure, we can see that the network produces some output at each location where the physician might have given an indication. (Note that, overall, the total network annotation is consistent. It always indicates the R wave at the first negative

wave of the cycle.)

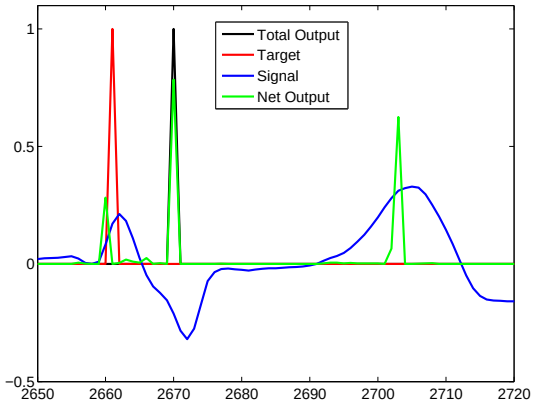

Figure 6.6: Intra-sequence Inconsistency of Annotation (Expanded)

This effect is also illustrated in Figure 6.7. This figure shows cardiac cycles in two different sequences. In the top figure, the physician indicated the R wave on the falling edge of the first negative wave. In the bottom figure, the physician indicated the R wave on the rising edge of the first positive wave. Note that there is a network output at both locations on both cycles, but only the largest one was selected.

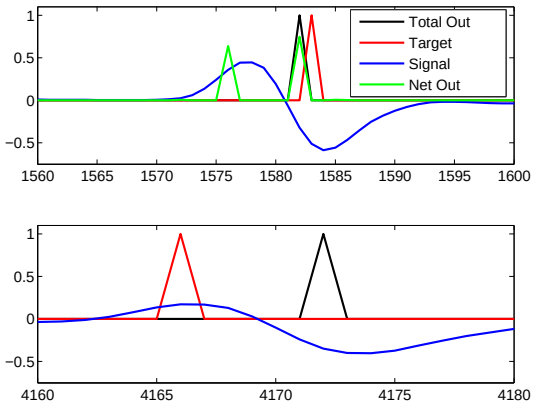

Figure 6.7: Inter-sequence Inconsistency of Annotation

There are also cases where the physician indicated multiple R waves within one cardiac cycle, as shown in Figure 6.8.

In other sequences there are even more indicated R waves within one cycle, as in Figure 6.9 and Figure 6.10.

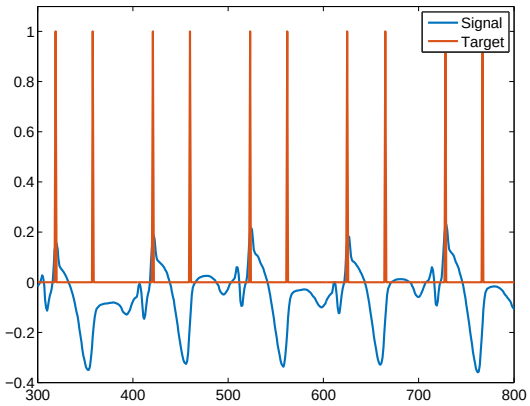

Figure 6.8: Multiple Targets per Cardiac Cycle

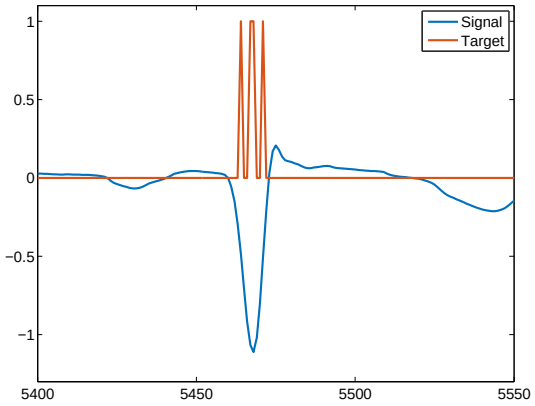

Figure 6.9: Three Targets in a Cardiac Cycle

To summarize, it seems that the number of errors (for the 10 ms precision) can be reduced by up to 60%, if the physician annotations can be made more consistent.

6.1.2 Noisy and Irregular Sequences

A certain percentage of the sequences have a high level of noise, which makes it difficult to determine the R locations. An example is shown in Figure 6.11.

There are also sequences with irregular shapes, which are often combined with physician annotations that do not seem to be consistent, such as the sequence shown in Figure 6.12.

Overall, we estimate that approximately 32% of the errors produced by the neural network, for a detection width of $d_w = 3$, are caused by signals with a large amount of noise. These errors are very unlikely to be improved by larger networks and additional

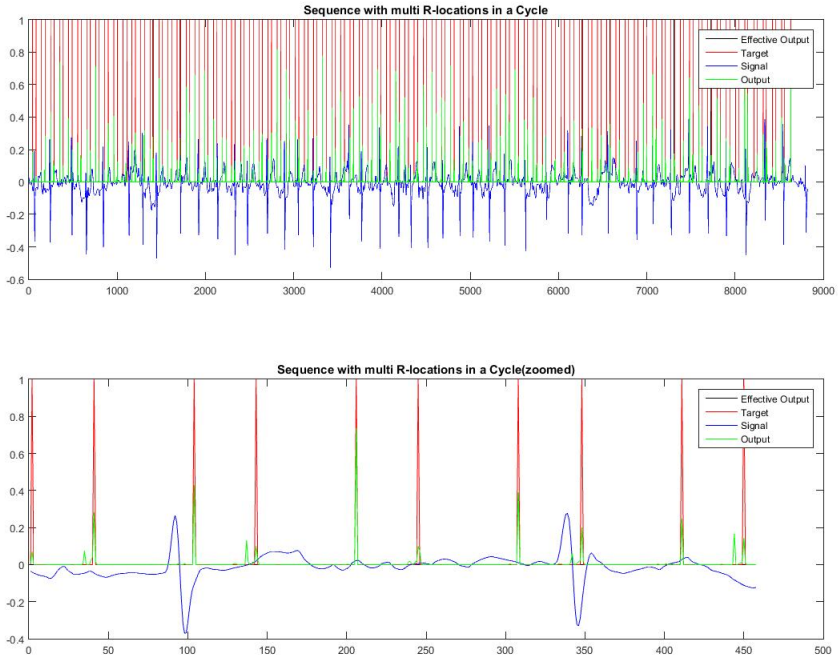

Figure 6.10: Sequence with Multiple R Wave Indications

training, since the noise level is so high that even a close visual inspection of the signal by a trained physician could only produce a very rough guess of where the QRS complex was located. In these cases, it probably would make sense to either remove these sequences from the training set, or have no R wave indications.

6.1.3 Small Amplitude Signals

In addition to the inconsistencies in signal annotation and the high noise levels in some signals, there was a third issue that we uncovered in analyzing the network errors. There are some EKG sequences in the data base that have a much lower signal level than the large majority of sequences. This is illustrated in Figure 6.13. Because there were so few of these types of sequences, relative to the total number of sequences, the network did not see enough examples to consistently recognize the patterns. There were also issues with consistent physician annotations with these sequences, as can be seen in Figure 6.13.

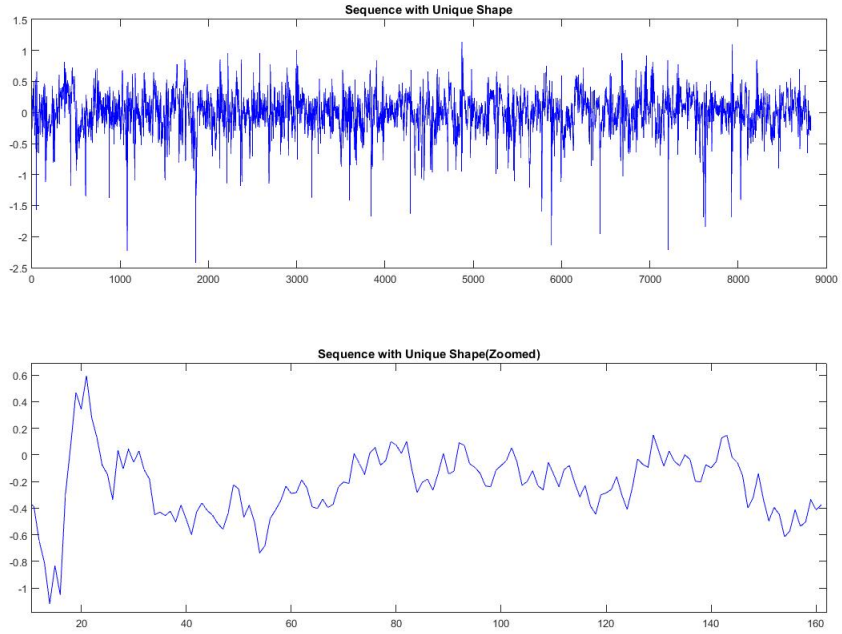

Figure 6.11: Noisy Sequence

6.2 Comparison with the Pan-Tompkins Algorithm

A common algorithm used for QRS complex detection is the Pan-Tompkins algorithm [2]. According to its authors, "the algorithm detects QRS complexes using slope, amplitude, and width information. A bandpass filter preprocesses the signal to reduce interference, permitting the use of low amplitude thresholds in order to get high detection sensitivity. In the algorithm, they used a dual-thresholds technique and search back for missed beats. The algorithm periodically adapts each threshold and RR interval limit automatically. This adaptive approach provides for accurate use on EKG signals having many diverse signal characteristics, QRS morphologies, and heart rate changes."

Since Pan-Tompkins is considered a standard, we will use it to provide a baseline to compare with our neural network algorithm. It was recently tested [3] against two other popular methods and was found to be the most accurate. We used an implementation of the Pan-Tompkins algorithm in MATLAB by Hooman Sedghamiz [4]. The results of applying this algorithm to the same data set that we used to train the neural network is shown in Table 6.3. This is a repeat of Table 6.1, with the added first row that contains the Pan-Tompkins statistics. We can see that the

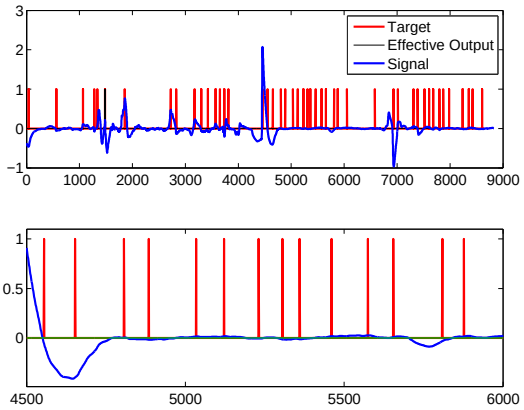

Figure 6.12: Irregular Sequence

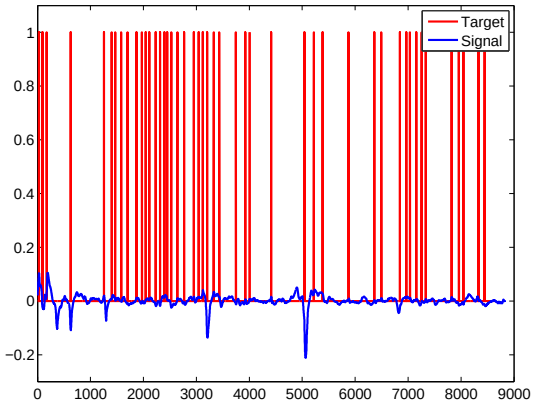

Figure 6.13: A Low Amplitude Sequence

neural network method is consistently better than Pan-Tompkins in all cases. For $d_w = 3$, the RER is reduced from 33% to 10%. For $d_w = 50$, it is reduced from about 7% to about 4%.

For the high precision case ($d_w = 3$), Pan-Tompkins has particular problems, especially with negative sequences. This can be seen when comparing Figure 6.14, which shows results for a segment of a positive sequence, and Figure 6.15, which shows results for a segment of a negative sequence. For these two full sequences, the neural network method made no errors. The Pan-Tompkins method did well on the positive sequence, but it made 45 errors on the negative sequence, which means that it did not correctly identify any R wave location with a precision of $d_w = 3$.

Table 6.4 shows the test set errors of the Pan-Tompkins algorithm, as well as the neural network errors. The test set errors for Pan-Tompkins are similar to the training

Method	$d_w = 50$				$d_w = 3$			
	FDR	MR	RER	ER	FDR	MR	RER	ER
Pan-Tom	0.0427	0.0346	0.0744	3.33e-4	0.2027	0.1978	0.3337	2.08e-3
Net1	0.0215	0.0236	0.0442	1.91e-4	0.0559	0.0569	0.1068	4.95e-4
Net2	0.0204	0.0224	0.0419	1.81e-4	0.0553	0.0562	0.1057	4.89e-4
Net3	0.0207	0.0227	0.0424	1.83e-4	0.0549	0.0559	0.1049	4.85e-4
Com	0.0201	0.0220	0.0412	1.78e-4	0.0421	0.0787	0.1146	5.36e-4

Table 6.3: Comparison of Neural Network and Pan-Tompkins Methods

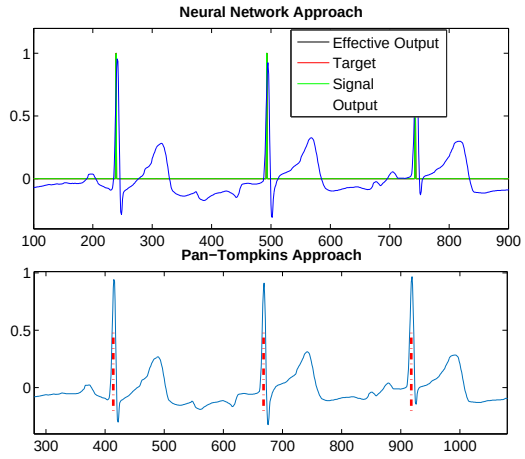

Figure 6.14: NN and Pan-Tompkins Methods on a Positive Sequence

set errors. The high precision, $d_w = 3$, RER is unchanged, and the low precision, $d_w = 50$, RER is only slightly higher. The neural network performs consistently better than Pan-Tompkins in both cases. Because the test set was not used at all for training the network, or setting any algorithm parameters, and because testing and training errors are consistent for the neural network and Pan-Tompkins results, we can expect the neural network method to outperform Pan-Tompkins significantly on sequences that would be collected in the future.

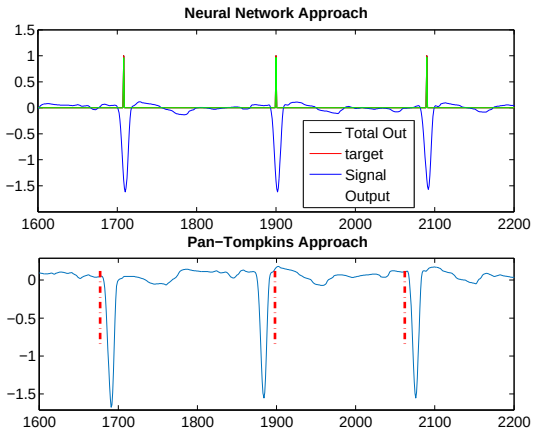

Figure 6.15: NN and Pan-Tompkins Methods on a Negative Sequence

Method	$d_w = 50$				$d_w = 3$			
	FDR	MR	RER	ER	FDR	MR	RER	ER
Pan-Tom	0.0437	0.0399	0.0802	3.59e-4	0.2000	0.1980	0.3321	2.04e-3
Net1	0.0206	0.0262	0.0458	1.97e-4	0.0570	0.0615	0.1119	5.17e-4
Net2	0.0220	0.0264	0.0473	2.04e-4	0.0575	0.0610	0.1119	5.17e-4
Net3	0.0220	0.0259	0.0468	2.01e-4	0.0580	0.0608	0.1121	5.18e-4
Com	0.0225	0.0232	0.0447	1.92e-4	0.0574	0.0572	0.1084	4.99e-4

Table 6.4: Test Set Comparison of Neural Network and Pan-Tompkins Methods

CHAPTER 7

DATA FOR BEAT CLASSIFICATION

In the previous chapters, we have covered the automated QRS detection system. This chapter and the chapters that follow will cover the classification of cardiac cycles into normal and abnormal categories. After detecting the beat locations within the sequences, we need to identify whether those beats are normal or abnormal. Albeit the focus of QRS detection is on the entire sequence, the focus of beat classification is on individual beats, which we need to extract from the sequences. In this chapter, we describe the mechanisms used to select the beats to be used to train the beat classifier and the characteristics of the final data set.

7.1 Selection of Beats

We used the QRS detection system to identify each cycle or beat in a sequence. Wherever a cycle is detected, and the identified R location agrees with a physician annotation, the contents of the 351 element tapped delay line at the input to the QRS detection network are saved. At the same time, the physician indication of normal or abnormal for that beat are also saved. This was done for all EKG sequences. The physicians' indication of a beat identifies one of these five categories: Atrial A, Junctional J, Normal N, Ventricular V, or Unidentified X. Any A, J, V, or X represents an abnormal beat, whereas N represents a normal beat.

In order to conveniently plot annotations and beats, we assigned numbers one through five, alphabetically, to represent the annotations. Therefore A, J, N, V, and X are represented by 1, 2, 3, 4, and 5 respectively. All but N (3) represent abnormal beats. (The binary representation of the target data for this beat classification system is discussed in Section 7.2.) An abnormal beat of type V, is given in Figure 7.1. Here the amplitude of the target is 4, which corresponds to the alphabetical representation of arrhythmia type V. A normal heart beat is shown in Figure 7.2. Note that the amplitude of the target is 3, representing the normal beat, which is type N.

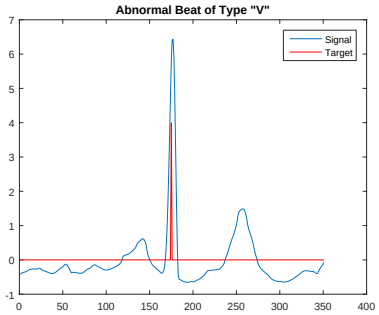

Figure 7.1: Abnormal Beat of Type V

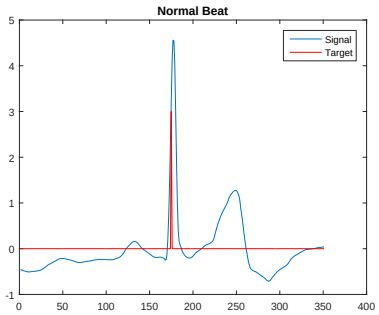

Figure 7.2: Normal Beat

7.2 Data Description

From the QRS detection system, those detected cycles within a detection width of 3 of the physicians annotation were selected for inclusion in the beat classification data set. There were 400,544 beats that met this criterion. The 351x1 tapped delay line input for each of these beats then formed the inputs for the beat classification data set. If the physician indicated that a beat was normal, N, then the corresponding target for that beat was assigned to be $\begin{bmatrix} 0 & 1 \end{bmatrix}^T$. If the physician indicated that the beat was abnormal (A, J, N, V, or X), the corresponding target was assigned to be $\begin{bmatrix} 1 & 0 \end{bmatrix}^T$.

When dividing the data into training, validation and test sets, it is important that each subset be representative of the full data set. For this reason, we need to investigate the characteristics of the data. One characteristic that was important for the QRS detection system was whether the R wave was a positive or negative pulse. For the beat detection data, if the R wave of a beat is a negative pulse, then we will call it a negative beat; if it is a positive pulse, we will call it a positive beat. As in the QRS detection system, we need to consider the distribution of the positive

and negative beats as we form training, testing and validation sets. An example of a positive normal beat is given in Figure 7.2, and an example of a positive abnormal beat is given in Figure 7.1. Two negative beats, one normal and one abnormal, are shown in Figure 7.3. Notice that the amplitude of the targets in the beats indicates whether they are normal or abnormal.

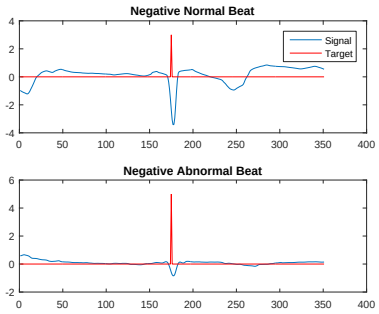

Figure 7.3: Negative Beats

In the beat classification data set, we have a total of 11,810 abnormal and 388,734 normal beats. The percentages of positive and negative beat distributions within the normal and abnormal cases is given in Table 7.1. It is interesting that the ratio of positive and negative beats in both the normal and the abnormal cases is very similar, and is close to the ratio of positive and negative sequences in the QRS detection system, which was 80% positive and 20 % negative.

Table 7.1: Positive and Negative Beats in Normal and Abnormal Sets

	Normal	Abnormal
Positive Beats	81	79.2
Negative Beats	19	20.8

CHAPTER 8

NETWORK AND TRAINING FOR BEAT CLASSIFICATION

As with QRS detection, beat classification is a pattern recognition problem. In pattern recognition problems, there are commonly used transfer functions, training algorithms and performance index functions. These features are known to be effective in classification problems. For that reason, for the beat classification system, we used some of the same network and training features that we used in the QRS detection system.

8.1 Network Architecture

In the QRS detection system, we needed a moving window of the EKG signal, and for that we used a network with memory. The general network architecture was a focused time delay network, as shown in Figure 2.7. For the beat detection system, a two layer feed forward neural network architecture with no memory is used, as shown in Figure 8.1. There are $R = 351$ elements of the input vector, which corresponds to the size of the tapped delay line in the QRS detection network. The first layer of this network has $S^1 = 40$ neurons and a tansigmoid transfer function (see Eq. 2.7). Since using forty neurons in the hidden layer resulted in a satisfactory result in the QRS detection system, we continued using it for beat classification. The second layer of the network has two neurons and a softmax transfer function, Eq. 2.6.

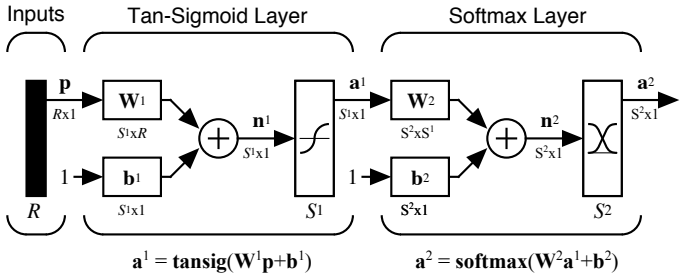

Figure 8.1: Beat Classification Network

8.2 Performance Index

As in the QRS detection system, we will use the weighted Cross Entropy performance index, Eq. 3.3. The beat classification problem also has an unbalanced data set; as was described in Section 7.2, only 3% of the beats are abnormal. As discussed in the paragraph before Eq. 3.3, it is possible to make the error very small by simply classifying every point as normal. This would increase Type II errors, but the overall error rate would be small. To prevent this problem, we can weight Type II errors more than Type I errors. In a similar strategy to the QRS detection problem, we will use an error weighting of one for all normal beats, and an error weighting of ρ for all abnormal beats. The selection of ρ will be discussed in a later section.

8.3 Training Algorithm and Stopping Criteria

In order to train the beat classification neural network, as in the QRS detection network, we used the Scaled Conjugate Gradient (SCG) training algorithm. Unlike the QRS detection system, the input data was small enough to fit in computer memory, so we did not have to divide the data into mini-batches. Moreover, we were able to use a GPU for training. In comparison to training only with the CPU, training using the GPU was five time faster. Relative to the QRS detection system, the computational burden is reduced, because of the smaller data set.

As in the training of the QRS detection network, we used early stopping to prevent overfitting. The available data is divided into a training set and a validation set. (As described earlier, a separate testing set was already put aside.) The training set is used to compute gradients and to determine the weight update at each iteration. The validation set is an indicator of what is happening to the network function in between the training points, and its error is monitored during the training process. When the error on the validation set goes up for ten iterations, the training is stopped, and the weights that produced the minimum error on the validation set are used as the final trained network weights. We randomly selected 15% of the data for the validation set, with the remaining 85% used for training. Training was also stopped, if the validation error did not go up after 500 iterations, but this rarely happened.

8.4 Committee of Networks

In the QRS detection system, we used a committee of networks to improve performance. Each network in the committee was trained with different initial weights and different training and validation sets. The method for combining the committee outputs in that case was somewhat complex, because the R wave indications of each network could be slightly offset from each other. For the beat classification case, the committee formation is much simpler, since the individual beats are aligned.

We considered two approaches for combining the network outputs to provide a single committee classification. In the first approach – the voting committee – each network is given a vote. For example, if there are ten networks in the committee, and five or more networks have their first output above the detection threshold (typically $d_t = 0.5$), then the committee indicates an abnormal beat. In the second approach – the averaged committee – the first outputs from all the networks are averaged together, and if the average is above the detection threshold, then the committee indicates an abnormal beat.

8.4.1 Committee Using Full Data

We used two different methods to compensate for the unbalanced data set. For one method, we used the full data set, but we tested different error weighting ratios (ρ) to determine a value that balanced the percentage of Type I and Type II errors. For each ρ, one hundred networks were trained.

8.4.2 Committee Using Balanced Data

Another way of compensating for the unbalanced data set is to produce new training sets that contain an equal number of normal and abnormal beats. In each such training set, the proportion of negative and positive beats needs to be as indicated in Table 7.1. In addition, for each network the balanced data sets should contain some different beats. As mentioned in the previous chapter, the total data set contains 11,810 abnormal and 388,734 normal beats. For the balanced data sets, we selected 10,628 beats (90% of 11,810) at random from both the normal and abnormal sets, for a total of 21,256 total beats. Each of these balanced sets was then used to train one network. For a committee with 100 networks, we selected 100 balanced data sets. Since each data set was balanced, the error weighting ratio was one.

8.5 Definition of Errors

The beat classifier considers an abnormal beat to be positive and a normal beat to be negative. We assessed the performance of the system using false positive rate (FPR), false negative rate (FNR), and error rate (ER). FPR indicates the percentage of normal beats that are incorrectly classified as abnormal. On the other hand, FNR (also called miss rate, MR) indicates the percentage of abnormal beats that are miss-classified as normal. We also used the standard measure, ER, to determine how often the classifier made wrong decisions of any type. These measures of performance (except for FPR) are also defined in Table 5.2, and the definitions of the variables that define the measures are given in Table 5.1.

Two other measures of performance of a pattern recognition system are sensitivity and specificity. The probability that the classifier correctly detects an abnormal beat as abnormal is the sensitivity, or true positive rate (TPR), which is given by $\frac{TP}{P}$. On the other hand, the probability that the classifier correctly detects a normal beat as normal is the specificity, or true negative rate (TNR), which is given by $\frac{TN}{N}$.

A curve that illustrates the relation between TPR and FPR, as the detection threshold is adjusted, is known as the Receiver Operating Characteristic (ROC) curve [6]. For measuring the performance of the classifier, we used the area under the ROC curve (AUROC). AUROC measures the discriminating performance of the classifier. An area of 1 means that the system discriminates the positives and negatives perfectly. In contrast, an area of 0.5 means that the system discriminates the positives and negatives by chance. Any AUROC close to 1 is good and anything close to 0.5 is bad. An example of an ROC curve whose AUROC is 0.8521 is given in Figure 8.2.

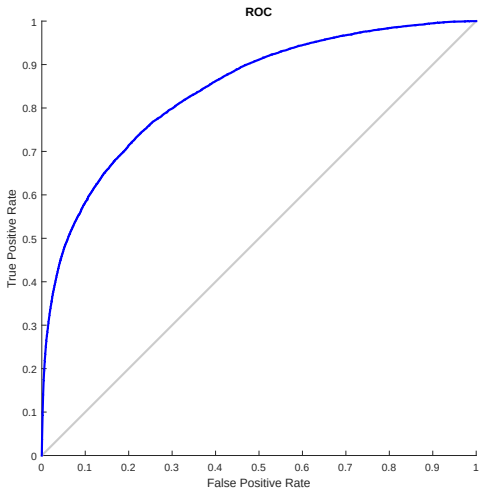

Figure 8.2: ROC

CHAPTER 9

BEAT CLASSIFICATION RESULTS

In Section 8.4, we described two methods for forming a committee output. For the first method – the voting committee – each network is given a vote, and if the number of positive votes is greater than or equal to the voting threshold v_t (typically half of the total number of networks), then the committee vote is considered positive. (An individual network vote is considered positive, if the first network output is greater than or equal to the detection threshold d_t [typically 0.5].) For the second method – the averaged committee – the outputs of the individual networks are averaged, and the committee result is considered positive, if the average of the first output is greater than or equal to d_t.

In Section 8.4, we also described two ways of forming the training data set. In one method, balanced data sets (equal numbers of normal and abnormal beats) are obtained to train committee members by randomly selecting subsets of the full data set. In the other method, the full data set is used. However, when the full data set is used, something must be done to compensate for the fact that there are fewer abnormal beats than normal beats in the overall data set. There are three ways to compensate – we can adjust the error weighting ρ, the detection threshold d_t or the voting threshold (v_t). Each of these parameters influences FPR and FNR, which we would like to keep relatively equal. If ρ is increased (or v_t or d_t is decreased), then FPR will increase and FNR will decrease. The idea will be to hold two of these parameters fixed, and adjust the other two until FPR and FNR are approximately balanced.

In Section 9.1, we show results for the balanced data set and the voting committee. In this case, $\rho = 1$ and $d_t = 0.5$, but v_t is adjusted to make FPR and FNR approximately equal.

In Section 9.2, we use the full data set to train a voting committee of 100 networks, and we begin by adjusting ρ, with $d_t = 0.5$ and $v_t = 50$. We select two values of ρ, and then use these two values in the remainder of the section. In Subsection 9.2.1, we set $\rho = 5$ or $\rho = 39$ and $d_t = 0.5$, and then we adjust v_t to balance TPR and TNR

for a voting committee. In Subsection 9.2.2, we set $\rho = 5$ or $\rho = 39$ and $v_t = 50$, and then we adjust d_t to balance TPR and TNR for an averaged committee. Finally, in Section 9.3, we analyze the beat classification errors.

9.1 Errors for Balanced Data Committee

Based on the balanced data formation described in Subsection 8.4.2, 100 different training data sets were formed and were used to train 100 networks. The voting committee output is considered positive if at least v_t networks have their first output above $d_t = 0.5$. For this committee, FP, FN, total error, FPR, FNR (MR) and ER are calculated. According to Table 9.1, FPR and FNR are approximately balanced when $v_t = 70$. At this voting threshold, FNR, FPR and ER are all near 29%. The ROC curve for the averaged committee is given in Figure 9.1. The corresponding AUROC is 0.7915. This indicates that the averaged committee of beat classifiers formed using balanced data discriminates normal and abnormal beats with a probability of 0.7915. This result will be compared with the result of a committee formed using the full data set in Section 9.2.

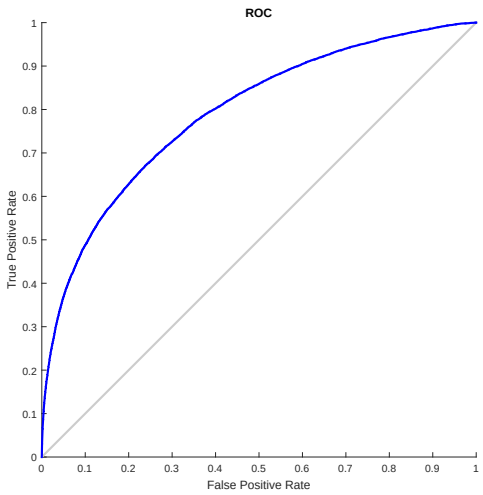

Figure 9.1: ROC curve for Balanced Data

9.2 Errors for Full Data Committee

When using the full data set, we selected values of ρ to produce approximately equal FPR and FNR, and also to balance the number of false negatives and false positives.

Table 9.1: Errors for Balanced Data with Varying v_t

v_t	**FN**	**FP**	**Sum**	**FNR**	**ER**	**FPR**
5	113	358356	358469	0.009568	0.894955	0.921854
10	250	332604	332854	0.021169	0.831005	0.855608
15	404	310143	310547	0.034208	0.775313	0.797828
20	549	289966	290515	0.046486	0.725301	0.745924
25	723	271474	272197	0.061219	0.679568	0.698354
30	887	254136	255023	0.075106	0.636692	0.653753
35	1079	237393	238472	0.091363	0.59537	0.610682
40	1282	220563	221845	0.108552	0.553859	0.567388
45	1515	203736	205251	0.128281	0.512431	0.524101
50	1788	186839	188627	0.151397	0.470927	0.480635
55	2103	169533	171636	0.178069	0.428507	0.436116
60	2453	151548	154001	0.207705	0.38448	0.38985
65	2868	133090	135958	0.242845	0.339433	0.342368
70	3403	113950	117353	0.288146	0.292984	0.293131
75	4020	94342	98362	0.34039	0.245571	0.24269
80	4761	74633	79394	0.403133	0.198215	0.19199
85	5646	54856	60502	0.478069	0.15105	0.141114
90	6856	35106	41962	0.580525	0.104763	0.090309
95	8408	16761	25169	0.711939	0.062837	0.043117
100	10819	1975	12794	0.916088	0.031942	0.005081

Table 9.2 compares results for several error weighting ratios. The FN and FP values are close when $\rho = 5$, and FPR and FNR are close when $\rho = 39$. For each of these two values for ρ, we trained 100 networks with different random initial weights, and different training/validation divisions. The resulting networks were used to form committees using the voting and averaging approaches described in Section 8.4.

9.2.1 Using Voting Threshold

This section discusses the results for the voting committee, trained on the full data set, which was formed by counting the number of networks whose first output was above the detection threshold $d_t = 0.5$. For error weightings of $\rho = 5$ and $\rho = 39$, we tested twenty different v_t values. The committee output is considered positive (indicating an abnormal beat), if at least v_t networks out of the hundred have their first output above 0.5. If not, the given beat will be considered normal. The summary of the error calculations when $\rho = 5$ and $\rho = 39$ are given in Table 9.3 and Table 9.4, respectively.

The ROC curve of the averaged committee output using $\rho = 5$ is given in Figure 9.2.1. The corresponding AUROC is 0.8376. Notice that the discriminating performance of this committee is better than the committee formed using the balanced data set. In this case, the probability that the system will discriminate normal and abnormal beats is about 84%, as compared to 79% when balanced data sets are used.

(Observe in Table 9.3 that as v_t increases, the FP error decreases. When $v_t = 100$, we were able to classify most of the normal beats with an error rate of only 2.89%. Those remaining 119 normal beats that were not classified correctly – the false positives in Table 9.3 – must have been very difficult for the networks to classify. A detailed analysis of these beats is given in Subsection 9.3.1, which will identify problems in the data set.)

With $\rho = 5$, it is not possible to balance FPR and FNR. For $\rho = 39$, as shown in Table 9.4, FPR and FNR are approximately equal when $v_t = 52$. At this value, FPR, FNR and ER are all approximately 25%. This is less than the 29% rate achieved using the balanced data sets. In Section 9.3.2, we will go through the details of the beats that are misclassified using $v_t = 52$.

The ROC curve for the averaged committee, trained with $\rho = 39$, is given in Figure 8.2. The corresponding AUROC is 0.8521, which is larger than the 0.8376

Table 9.2: Error Weighting Ratio Selection

ρ	**FN**	**FP**	**Sum**	**FNR**	**ER**	**FPR**
1	10380	904	11284	0.878916	0.028172	0.002325
2	9913	2244	12157	0.839373	0.030351	0.005773
3	8912	4087	12999	0.754615	0.032453	0.010514
4	8739	5626	14365	0.739966	0.035864	0.014473
5	8104	7859	15963	0.686198	0.039853	0.020217
10	6754	20543	27297	0.571888	0.06815	0.052846
15	6006	33815	39821	0.508552	0.099417	0.086988
20	5508	44100	49608	0.466384	0.123852	0.113445
25	4772	61410	66182	0.404064	0.16523	0.157974
30	4190	75079	79269	0.354784	0.197903	0.193137
35	3309	89465	92774	0.280186	0.23162	0.230145
40	3137	106958	110095	0.265622	0.274864	0.275144
45	3246	133845	137091	0.274852	0.342262	0.34431
50	2694	138970	141664	0.228112	0.353679	0.357494
55	2011	139073	141084	0.170279	0.352231	0.357759
60	2381	161968	164349	0.201609	0.410314	0.416655

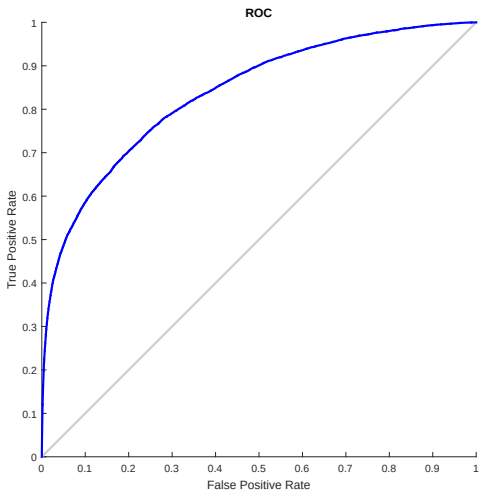

Figure 9.2: ROC Curve for Full Data Trained with $\rho = 5$

Table 9.3: Error Calculation Using Voting for Full Data, $\rho = 5$

v_t	**FN**	**FP**	**Sum**	**FNR**	**ER**	**FPR**
5	5474	30699	36173	0.463506	0.09031	0.078972
10	6033	21492	27525	0.510838	0.068719	0.055287
15	6447	16448	22895	0.545893	0.05716	0.042312
20	6809	13086	19895	0.576545	0.04967	0.033663
25	7085	10565	17650	0.599915	0.044065	0.027178
30	7369	8592	15961	0.623963	0.039848	0.022103
35	7641	6983	14624	0.646994	0.03651	0.017963
40	7920	5735	13655	0.670618	0.034091	0.014753
45	8143	4724	12867	0.6895	0.032124	0.012152
50	8394	3941	12335	0.710754	0.030796	0.010138
55	8651	3244	11895	0.732515	0.029697	0.008345
60	8879	2726	11605	0.75182	0.028973	0.007013
65	9081	2259	11340	0.768925	0.028311	0.005811
70	9337	1858	11195	0.790601	0.027949	0.00478
75	9571	1529	11100	0.810415	0.027712	0.003933
80	9807	1268	11075	0.830398	0.02765	0.003262
85	10070	1033	11103	0.852667	0.02772	0.002657
90	10358	798	11156	0.877053	0.027852	0.002053
95	10722	520	11242	0.907875	0.028067	0.001338
100	11427	119	11546	0.96757	0.028826	0.000306

value for $\rho = 5$ and the 0.7915 value for the balanced data set method.

To summarize, the best result is observed when we use the voting committee, trained with the full data set, and with $\rho = 39$, $d_t = 0.5$ and $v_t = 52$. The resulting FPR, FNR and ER are somewhat less than 25%.

It would seem that with more than 400,000 sample beats in the data set, we should be able to achieve smaller error rates. In the analysis section, Section 9.3, we discuss the types of errors that occur and their causes.

9.2.2 Using Detection Threshold

Another way of compensating for the unbalanced data set is to use an averaged committee and vary the detection threshold d_t. For 100 networks trained using $\rho = 5$ and $\rho = 39$, we formed averaged committees. We calculated the error rates for 20 different values of d_t. The summaries of the errors are given in Table 9.5 for $\rho = 5$ and Table 9.6 for $\rho = 39$.

Notice that FNR and FPR are approximately equal at $d_t = 0.125$ and at $d_t = 0.5$ for $\rho = 5$ and $\rho = 39$, respectively. However, these error rates are larger than those for the voting committee with $v_t = 52$. Hence, for further error analysis and application we use the network that produced the gray row in Table 9.4, which is the voting committee, trained with the full data set, and with $\rho = 39$, $d_t = 0.5$ and $v_t = 52$.

9.3 Error Analysis

In order to have a better understanding of why the errors occur, it is important to look at those beats that are difficult to classify. In this section, we will identify and analyze those beats carefully. Later in this section, we will categorize the types of errors that occur for the best committee.

9.3.1 Difficult to Classify Beats

The first step in this section is to identify those beats that are most difficult to classify correctly. If we reduce ρ, the number of false positives will be reduced, as there is a smaller penalty for the false negatives. Also, increasing v_t leads to a reduction of the number of false positives. This is because a larger number of networks must agree on the existence of an abnormal beat. Therefore, if we use a small ρ and a large v_t, we will have a small number of false positives, and these will be the normal beats

Table 9.4: Error Calculation Using Voting for Full Data with $\rho = 39$

$v_t(\rho = 39)$	FN	FP	Sum	FNR	ER	FPR
1	548	258123	258671	0.046401	0.645799	0.664009
2	710	237770	238480	0.060119	0.59539	0.611652
3	824	225638	226462	0.069771	0.565386	0.580443
4	913	216971	217884	0.077307	0.54397	0.558148
5	972	209771	210743	0.082303	0.526142	0.539626
10	1300	185817	187117	0.110076	0.467157	0.478006
15	1527	169708	171235	0.129297	0.427506	0.436566
20	1738	156914	158652	0.147163	0.396091	0.403654
25	1924	145653	147577	0.162913	0.368441	0.374686
30	2082	135718	137800	0.176291	0.344032	0.349128
35	2266	126136	128402	0.191871	0.320569	0.324479
40	2437	117049	119486	0.206351	0.298309	0.301103
45	2603	108347	110950	0.220406	0.276998	0.278718
50	2783	99479	102262	0.235648	0.255308	0.255905
52	2858	96063	98921	0.241998	0.246967	0.247118
55	2968	90940	93908	0.251312	0.234451	0.233939
60	3203	82080	85283	0.271211	0.212918	0.211147
65	3454	73298	76752	0.292464	0.191619	0.188556
70	3762	64586	68348	0.318544	0.170638	0.166144
75	4131	55511	59642	0.349788	0.148902	0.142799
80	4495	46066	50561	0.38061	0.126231	0.118503
85	5013	36339	41352	0.424471	0.10324	0.09348
90	5697	26243	31940	0.482388	0.079742	0.067509
95	6707	15915	22622	0.567909	0.056478	0.040941
100	8943	4227	13170	0.75724	0.03288	0.010874

Table 9.5: Error Calculation Using Threshold with $\rho = 5$

$d_t(\rho = 5)$	**FN**	**FP**	**Sum**	**FNR**	**ER**	**FPR**
0.05	517	262924	263441	0.043776	0.657708	0.67636
0.1	2362	121958	124320	0.2	0.310378	0.313731
0.125	3232	87334	90566	0.273666	0.226107	0.224663
0.15	3881	65513	69394	0.32862	0.173249	0.168529
0.2	4822	40314	45136	0.408298	0.112687	0.103706
0.25	5578	26603	32181	0.472312	0.080343	0.068435
0.3	6183	18025	24208	0.523539	0.060438	0.046368
0.35	6775	12317	19092	0.573666	0.047665	0.031685
0.4	7316	8356	15672	0.619475	0.039127	0.021495
0.45	7896	5548	13444	0.668586	0.033564	0.014272
0.5	8415	3834	12249	0.712532	0.030581	0.009863
0.55	8936	2610	11546	0.756647	0.028826	0.006714
0.6	9424	1736	11160	0.797968	0.027862	0.004466
0.65	9879	1196	11075	0.836494	0.02765	0.003077
0.7	10250	819	11069	0.867909	0.027635	0.002107
0.75	10632	538	11170	0.900254	0.027887	0.001384
0.8	11046	325	11371	0.935309	0.028389	0.000836
0.85	11449	133	11582	0.969433	0.028916	0.000342
0.9	11721	24	11745	0.992464	0.029323	6.17E-05
0.95	11809	0	11809	0.999915	0.029482	0
1	11810	0	11810	1	0.029485	0

Table 9.6: Error Calculation Using Threshold with $\rho = 39$

$d_t(\rho = 39)$	**FN**	**FP**	**Sum**	**FNR**	**ER**	**FPR**
0.05	0	388594	388594	0	0.970166	0.99964
0.1	9	377635	377644	0.000762	0.942828	0.971448
0.15	59	350280	350339	0.004996	0.874658	0.901079
0.2	168	315827	315995	0.014225	0.788915	0.81245
0.25	351	278323	278674	0.029721	0.695739	0.715973
0.3	611	238784	239395	0.051736	0.597675	0.614261
0.35	996	198208	199204	0.084335	0.497334	0.509881
0.4	1560	159995	161555	0.132091	0.403339	0.41158
0.45	2160	126779	128939	0.182896	0.32191	0.326133
0.5	2755	98761	101516	0.233277	0.253445	0.254058
0.55	3485	75255	78740	0.295089	0.196583	0.19359
0.6	4141	55666	59807	0.350635	0.149314	0.143198
0.65	4900	39593	44493	0.414903	0.111081	0.101851
0.7	5672	26551	32223	0.480271	0.080448	0.068301
0.75	6517	16523	23040	0.55182	0.057522	0.042505
0.8	7535	9249	16784	0.638019	0.041903	0.023793
0.85	8645	4281	12926	0.732007	0.032271	0.011013
0.9	9893	1488	11381	0.83768	0.028414	0.003828
0.95	11195	350	11545	0.947925	0.028823	0.0009
1	11810	0	11810	1	0.029485	0

that are most difficult to classify. In contrast, a large ρ and a small v_t will give us a small number of false negatives, and these will be the abnormal beats that are most difficult to classify.

Table 9.3 shows errors for a small ρ (5). If we check the row for the largest v_t (100), we find that there are only 119 false positive beats. These are the most difficult normal beats to classify correctly. Table 9.7 shows errors for a large ρ (60). If we check the row for a small v_t (2), we find that there are only 201 false negative beats. These are the most difficult abnormal beats to classify.

Table 9.7: Error Calculation with $\rho = 60$

v_t	**FN**	**FP**	**Sum**	**FNR**	**ER**	**FPR**
1	0	388734	388734	0	0.970515	1
2	201	317975	318176	0.017019	0.79436	0.817976
3	262	301633	301895	0.022185	0.753712	0.775937
4	321	291318	291639	0.02718	0.728107	0.749402
5	365	283260	283625	0.030906	0.708099	0.728673

In order to understand why these 119 FP and 201 FN beats are difficult to identify, we will investigate other beats with similar shapes that are classified correctly. To measure the similarity between two beats, we will use Euclidean distance. If the distance between beats is small, we will call them neighbors, as defined below.

1. The Euclidean Distance d between two vectors, $\mathbf{p}$ and $\mathbf{q}$, is given by

$$d(\mathbf{p}, \mathbf{q}) = (\sum_{j=1}^{length(\mathbf{p})} (p_j - q_j)^2)^{0.5} \tag{9.1}$$

   where p_j and q_j are elements of the vectors $\mathbf{p}$ and $\mathbf{q}$, respectively.

2. Two vectors $\mathbf{p}$ and $\mathbf{q}$ are said to be neighbors, if $d(\mathbf{p}, \mathbf{q}) <= n_d$, where $n_d \in \mathbf{R}^+$ is called the neighbor distance. Through experimentation, we found that $n_d = 0.25$ produced beats that looked similar to each other, so we used this distance in the remaining research described below.

For each misclassified beat, we found up to 100 neighbors for analysis. Of the 119 false positives, 41 of them had at least one neighbor. Of the 201 false negatives, 169 of them had at least one neighbor. Then, for those misclassified beats with neighbors,

Table 9.8: Types of Neighbors for FNs and FPs

Type of Neighbor	# Neighbors of FN	# Neighbors of FP
FN	3	19
FP	669	17
TN	4954	16
TP	90	39

we identified the neighbors as TP, TN, FP, or FN. Table 9.8 shows the number of FP, FN, TP and TN neighbors of the false negative and false positive beats.

Notice that the number of TN neighbors of the false negatives is much higher than the other types of neighbors. There are 4,654 TN neighbors of the false negatives. A TN beat is a normal beat that is classified as normal by the network, whereas a false negative beat is an abnormal beat that is misclassified as normal. Because the neighbors of the FN beats are mainly TN beats, this indicates that there may be inconsistencies in the physician annotations (as we found for the QRS detection problem). If the beats are neighbors, then they must have a very similar shape. Since the FN beats look like many of the TN beats, this means that the physicians classified beats with the same shape in different ways. (This suggests that many FN beats are actually normal beats that some physicians classified as abnormal.) The network will make the choice that is consistent with the majority of the physician annotations, but if the physician annotations are not consistent, the network will make *mistakes* on the minority of annotations that disagree.

For the false positives, most of the neighbors are TP. As in the FN case, this suggests inconsistency in the physician annotations. There are FP and TP beats that look very similar. However, the physicians annotated some as normal and others as abnormal. The network will go with the majority of annotations, since this will optimize the overall performance. If two beats are almost identical in shape, the network must assign them to the same class. The physicians do not have the same constraint.

This effect is illustrated in Figure 9.3. The top left axis is a false negative beat, and the rest are its neighbors. Similarly, in Figure 9.4, the top left figure is a false positive beat, and the rest are its neighbors. Recall that the amplitude of a target indicates whether the beat is normal or abnormal. In Figure 9.3, the amplitude of the target for the FN beat is 4, which indicates a ventricular abnormality. But the rest of the

TN beats have a target of amplitude 3, which indicates a normal beat. Visually, all of the beats in each figure look very similar. However, they were annotated differently by the physicians. There was a sufficient amount of this inconsistency to significantly degrade network performance. We will quantify this effect in Subsection 9.3.2.

9.3.2 Types of Errors

When $\rho = 39$ and $v_t = 52$, we were able to obtain the smallest balanced values for FPR, FNR and ER. This is represented by the gray row in Table 9.4, which is reproduced in Table 9.9. Here, there are a total of 98,921 (2,858 FN and 96063 FP) misclassified beats. For each of these beats, we searched for neighbors within a distance of 0.25. Out of the 2,858 false negatives, 1,840 of them have neighbors, and out of the 96,063 false positives, 30,964 of them have neighbors. (Note that the number of neighbors of each misclassified beat might vary from 1 to 500.) There are 60,566 and 85,243 neighbors of false negatives and false positives, respectively. These neighbors are then classified as TP, TN, FP, or FN. The summary is given in Table 9.10.

Table 9.9: Minimum Balanced Errors

FN	**FP**	**Sum**	**FNR**	**ER**	**FPR**	**TP**	**TN**
2858	96063	98921	0.241998	0.246998	0.247118	8952	292671

Table 9.10: Neighbors Classification for $\rho = 39$

Type of Neighbor	# Neighbors of FNs	# Neighbors of FPs
FN	1050—0.88%	748—1.7%
FP	2313—31.7%	27041—3.8%
TN	56927—65.9%	56185—94%
TP	276—1.5%	1269—0.5%

For the false negatives, we see that the majority of the neighbors are TN. This is consistent with the discussion in Section 9.3.1 for the most difficult to classify beats. When the FN neighbors are TN, there are beats that look the same, but some are annotated as normal, and others are annotated as abnormal. This is going to cause difficulties for the neural network.

The pattern for the false positives in this case is different than it was for the most difficult to classify beats in the previous section. Here the largest group of neighbors for the false positives are TN. This is most likely caused by the unbalanced data set. Because we are trying to balance FNR and FPR, we are going to classify some normal beats as abnormal.

To clarify these ideas, we provide some specific examples next.

1. True Negative Neighbors of a False Negative Beat

   In Figure 9.3, a false negative beat is given at the left top corner; the rest are true negative neighbors. The type of neighbor, the vote and the distance to the neighbor are also shown at the top of each axis. Notice that the vote for each of the cases in the figure was zero. This indicates that all 100 networks agreed that all four beats shown in the figure have the same pattern, which is normal.

   Moreover, the relative distances of the false negative from the true negatives is 0.1, 0.1, and 0.13, which indicates that the beats are very close to each other (which can also be seen visually), although the physician annotation for the top left beat is different. There are 1,184 such beats that are annotated as abnormal while their neighbors are normal. This suggests that 41.4% of the false negative errors were caused by inconsistent annotations.

2. True Positive Neighbors of a False Positive Beat

   As with the false negative beat example, there are inconsistent annotations among the false positive beats. In Figure 9.4, all but the top left beat are normal. The number of votes in all of these four beats is 100, which indicates that all the 100 networks agreed that the given beat is indeed abnormal. There are 29,307 such beats that are annotated as normal while their neighbors are abnormal. If the annotation was consistent, we would have reduced 30.5% of the false positive errors caused by such inconsistent annotations.

3. Exactly the Same Sequences and Beats Annotated Differently

   Some sequences were annotated multiple times by different physicians. This has resulted in different annotations for exactly the same beats. According to Figure 9.5, there is no abnormal beat in the sequence, since the amplitude of the target in red is 3. When this same sequence was annotated by a different physician, in Figure 9.6, five abnormal beats are indicated. Note that the amplitude of the target in red is 1, wherever these abnormal beats occur.

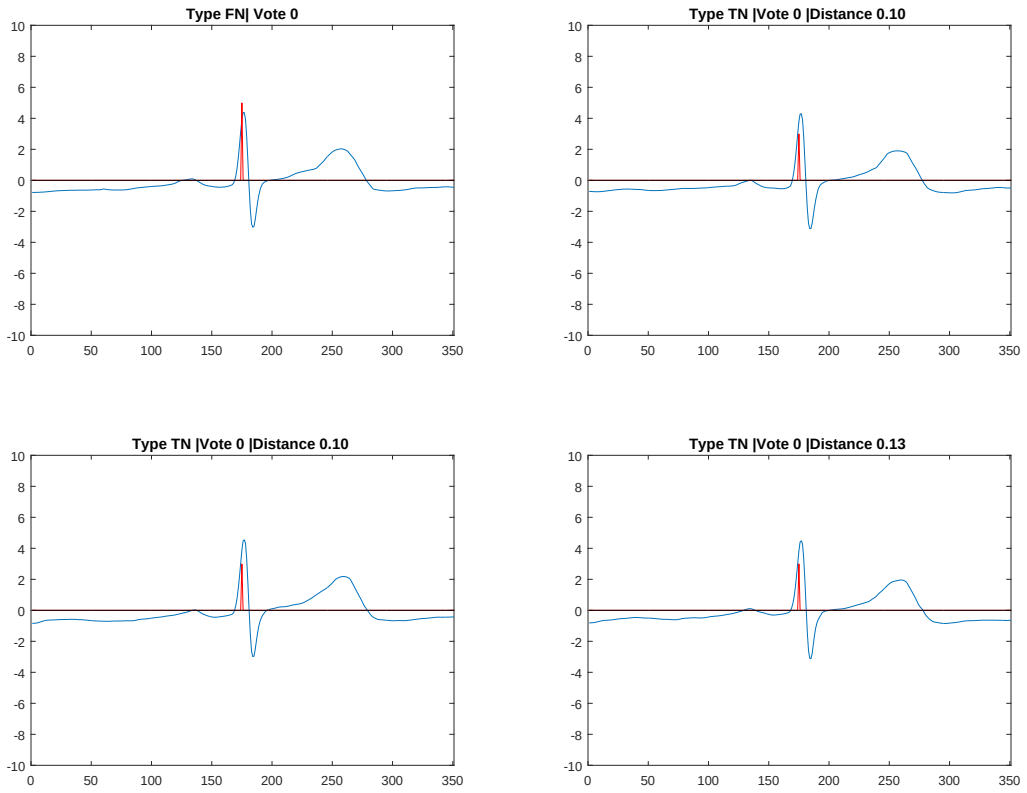

Figure 9.3: True Negative Neighbors of a False Negative Beat

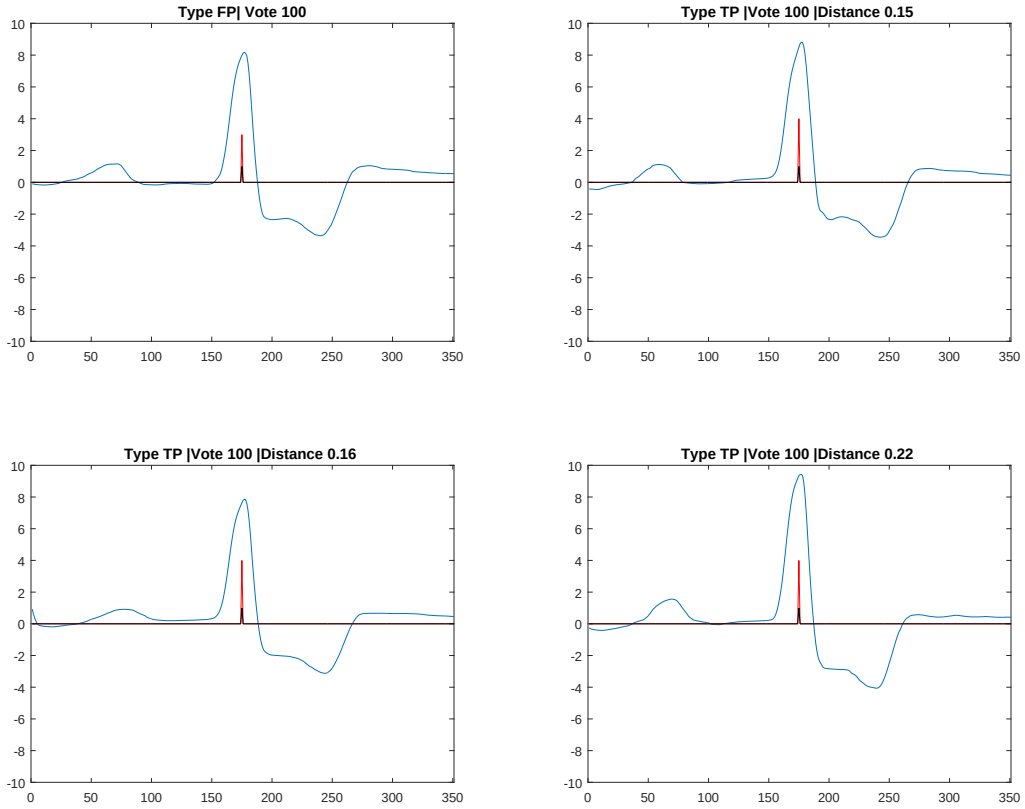

Figure 9.4: True Positive Neighbors of a Missclassified False Positive

On the other hand, Figure 9.7 shows that all of the three TN beats are exactly the same as the FN beat at the left top corner. Though they are exactly the same beats, as indicated by their relative distance of 0, the FN beat is annotated as 1, and the rest as 3. Clearly, these four beats are exactly the same. The network classified them as normal.

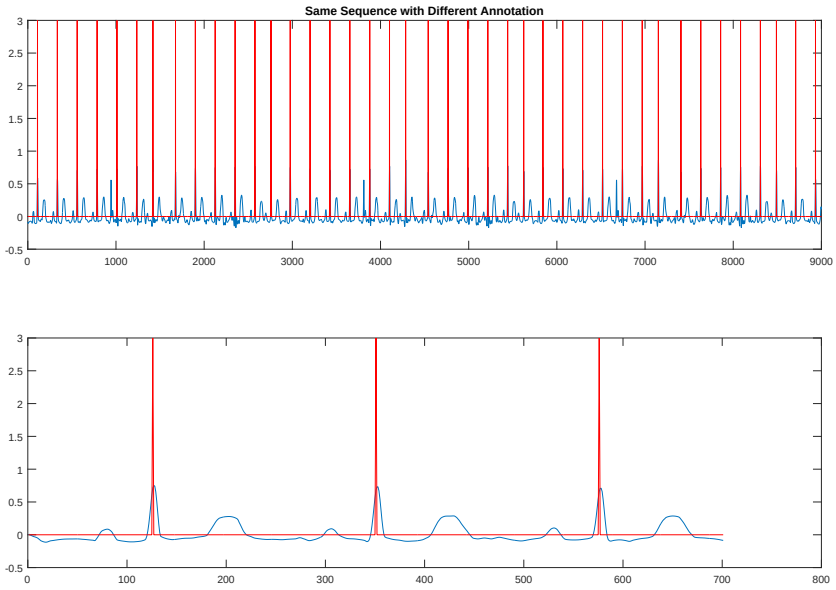

Figure 9.5: Exactly the Same Sequences Annotated Differently 2

Summary

There are many fewer abnormal beats than normal beats in the data set. If the error weighting were equal for all cases, the network could make the error very small by classifying all beats as normal. We have used an unbalanced error weighting in order to achieve a balance between false negative and false positive error rates. As we increase the error weighting on the false negatives, we can cause more abnormal beats to be correctly classified, but more normal beats will then be misclassified (false positives). For the abnormal beats that continue to be misclassified (false negatives), even as the error weighting increases, we find that approximately 94% of their neighbors are normal (true negative). This implies that the physicians are not consistent in their classifications.

There are certain types of beats that the physicians sometimes classify as normal, and other times classify as abnormal. The network classifies them as normal, because the majority of the physicians classify them as normal. But the physicians are not

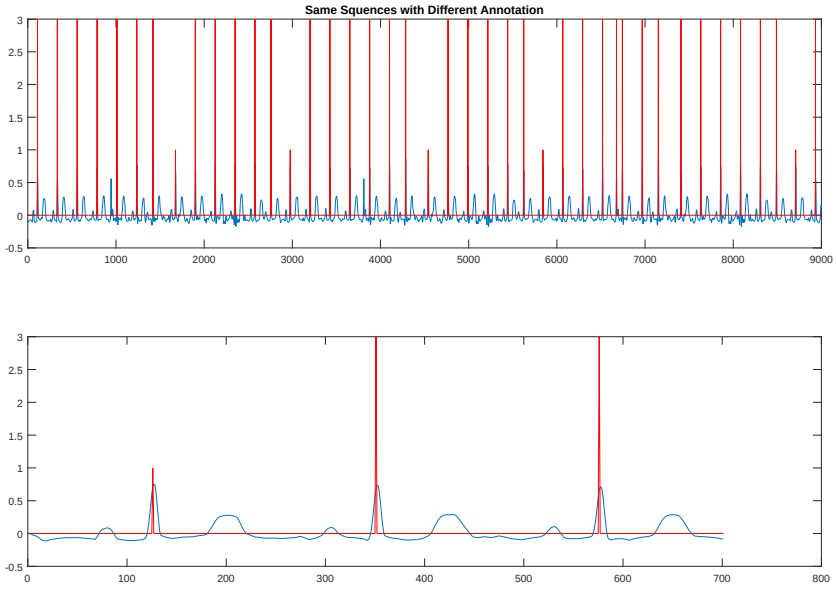

Figure 9.6: Exactly the Same Sequences Annotated Differently 1

consistent in these classifications. For the normal beats that are misclassified (false positives), some neighbors are false positives, but some neighbors are true negatives. The number of true negative neighbors increases as the error weighting increases, because the network is penalized more for false negatives than for false positives. It appears that a large proportion of the errors can be reduced by obtaining a more consistent physician classification. We estimate that we could have reduced 41.4% of the false negative errors and 30.5% of the false positive errors errors if there was no inconsistency.

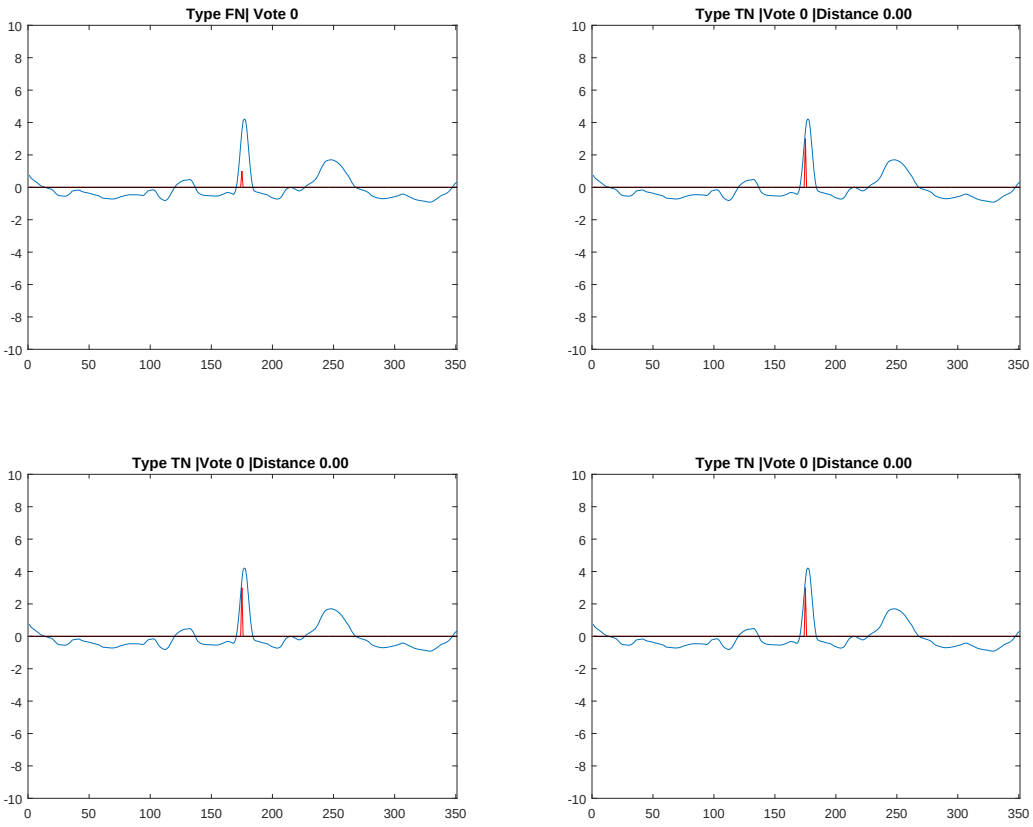

Figure 9.7: Exactly the Same Beats Annotated Differently

CHAPTER 10

CONCLUSIONS

This report has described the development of an automated system for EKG signals. The system involves two subsystems:

1. Detecting the QRS complexes in a signal.

2. Categorizing the normal and abnormal cardiac cycles in a signal.

This first subsystem – the detection of QRS complexes uses a focused time delay neural network (or a committee of such networks), followed by a threshold detection module, to provide an indication of the location of the R wave. The network was trained using 11,875 different physician-annotated sequences of EKG signals, each consisting of 9,000 time points at a 300 Hz sampling rate. To incorporate the 351 time delays at the input of the neural network, so that a window in time of the EKG signal can be used to determine the R wave location, the data used to train the network consisted of 19 groups of $351 \times 5,406,250$ input matrices and $2 \times 5,406,250$ target matrices. Only one group could be resident in memory at the same time, so training proceeded in stages.

The final trained QRS detection system was tested for precisions of 10 ms and 167 ms on the R wave location. On the training/validation data set, for the 10 ms precision, the system achieved a Miss Rate of approximately 5.6% and a False Detection Rate of approximately 5.5%, with a Relative Error Rate of approximately 11% and a total Error Rate of 0.049%. For the 167 ms precision, the system achieved a Miss Rate of approximately 2.3% and a False Detection Rate of approximately 2.1%, with a Relative Error Rate of approximately 4.2% and a total Error Rate of 0.018%.

On the test set, which was not used at all for network training, or for setting any algorithm parameters, the error rates were almost the same as the training set rates. Because the test set was randomly selected from the large original data set, and because it was not used in any way to develop the neural network QRS detection

system, we can expect that error rates obtained on future data sets will be similar to those described in this report.

An analysis of the errors in the automated QRS detection system found that approximately 60% of the training/validation set errors for the 10 ms precision occurred because of inconsistencies in the physician annotation. An additional 32% of the errors occurred because of very high noise on the EKG signal. A further 3.7% of the errors occurred because of very low signal levels. The remaining 4.3% of the errors occurred for miscellaneous reasons.

To provide a reference point with which to gauge the neural network annotation system, we also used the Pan-Tompkins QRS detection algorithm, which is the most cited EKG algorithm in the literature. Pan-Tompkins was recently tested [3] against two other popular methods and was found to be the most accurate. The results of our tests showed that the Pan-Tompkins error was a 200% increase (0.11 to 0.33 RER) over the neural network error for the 10 ms precision, and approximately a 75% increase (0.046 to 0.08 RER) for the 167 ms precision. The neural network significantly and consistently outperformed Pan-Tompkins throughout the training/validation data set. The neural network also outperformed Pan-Tompkins by a similar amount on the test set, which was not used in any way in developing the neural network QRS detection system.

The conclusion of this subsystem is that the neural network QRS detection system is very accurate. It can be made more accurate by refining the data set. In particular, if the inconsistency in physician annotations can be adjusted, the errors should drop significantly. In addition, with the more accurate data set, it should be possible to use larger (perhaps deeper) networks, a larger committee, and longer training times, which should further enhance the performance.

Although the QRS detection system can be improved, it is accurate enough to develop the beat classification system. In this second subsystem, an attempt has been made to classify cardiac cycles as normal and abnormal. First, we selected those correctly detected beats from the QRS detection system. We have 400,544 such beats, where each beat contains the contents of the 351 tapped delay line that are at the input to the QRS detection system. Along with this beats, the corresponding physician annotation for normal or abnormal is used to form the target set.

For this system we used a committee of 100 networks and a voting of 52. Each of the 100 networks that form the committee are trained using an error weighting of $\rho = 39$. The training and validation error of this system is 24.7% of ER, 24% of FPR

and FNR. However, 41.4% of the false negative and 30.5% of the false positives occur due to inconsistent annotations by the physicians.

10.1 Future Work

The final stage of the development of a full flagged system includes a system that specifies the arrhythmia of the abnormal cycles. This system can use all abnormal cycles from the beat classification data set. Again, a multilayer network can be used to refine the classification into types of arrhythmia. Starting from a two layer networks, one can use deeper networks with more neurons. This will depend on the accuracy of the QRS detection system and the beat classification system. Moreover, the refinement of the data set will be very crucial as the process proceeds.

BIBLIOGRAPHY

[1] M. Moller, "A scaled conjugate gradient algorithm for fast supervised learning," *Neural networks*, vol. 6, no. 4, pp. 525–533, 1993.

[2] J. Pan and W. J. Tompkins, "A real-time QRS detection algorithm," *IEEE Transactions on Biomedical Engineering*, vol. 32, pp. 230–236, 1985.

[3] R. A. Alvarez, A. J. M. Penin, and X. A. V. Sobrino, "A comparison of three qrs detection algorithms over a public database," *Procedia Technology*, vol. 9, pp. 1159–1165, 2013.

[4] H. Sedghamiz, "Pan-Tompkins Algorithm matlab code." `http://www.mathworks.com/matlabcentral/fileexchange/45840-complete-pan-tompkins-implementation-ecg-qrs-detector`. Accessed: 2016-04-06.

[5] M. T. Hagan and H. B. Demuth and M. H. Beale, *Neural Network Design*. Boston, MA: PWS, 1996.

[6] K. Woods, *Generating ROC Curves for Artificial Neural Networks*. IEEE Transactions on Medical Imaging, vol. 16, No. 3, 1997.

[7] E. S.Winokur and M. K. Delano and C. G. Sodini, "A wearable cardiac monitor for long-term data acquisition and analysis," *IEEE Transactions on Biomedical Engineering*, vol. 60, pp. 189–192, 2013.

[8] J. Pan and W. J. Tompkins, "A procedure for training recurrent networks," in *Proc. Int. Joint Conf. Neural Netw.*, pp. 1–8, Aug. 2013.

[9] Dale Dubin, *Rapid Interpretation of EKG's*. Hong Kong: ISBN, 2000.

[10] P. S. Hamilton and W. J. Tompkins, "Quantitative Investigation of QRS Detection Rules Using the MIT/BIH Arrhythmia Database," *IEEE Engineering in Medical and Biology Magazin,* , vol. BME-33, NO. 12, 1986.

[11] B. U. Kohler, C.Henning, and R. Olgmeister, "The Principles of Software QRS Detection," *IEEE Transactions on Biomedical Engineering,* , vol. 21, NO. 1, 2002.

[12] Anthony Atkielski, "Schematic Diagram of Normal Sinus Rhythm for a Human Heart as Seen on ECG(with English Labels," *en:Image:SinusRhythmLabels.png,* , vol. , NO. , 2007.

Name: Mesfin B. Taye Date of Degree: December, 2016

Institution: Oklahoma State University Location: Stillwater, Oklahoma

Title of Study: AUTOMATED EKG ANNOTATION WITH NEURAL NETWORKS

Pages in Study: 76 Candidate for the Degree of Master of Science

Major Field: Electrical Engineering

Abstract

We have developed two automated systems using 11,810 physician-annotated single lead EKG signals. A focused tap delay line multilayer neural network followed by a threshold detection module is used to develop the QRS detection system. Approximately, a miss rate of 5.6%, false detection rate of 5.5%, error rate of 0.049%, and relative error rate of 10.6% is observed with a precision of 10 ms. With this same precision, the Pan and Tompkins QRS detection algorithm is tested on these sequences and encountered 20.3% of false detection rate, 19.8% of miss rate, 0.2% error rate, and 33.4% of relative error rate. For a precision of 167 ms, the neural network has only 55% of the error made by the Pan and Tompkins QRS detection algorithm. The beat classification system uses the correctly detected beats along with the physician indication of normal and abnormal for the beat as an input for the multilayer neural network. The training and validation error for this system is 24.7% error rate, 24% false detection rate and miss rate. The inconsistency in the physician annotation has a significant effect on these systems. Sixty percent of the error in the QRS detection system is due to the inconsistency. About 41.4% of the false negative and 30.5% of the false positive errors of the beat classification system are also due to this inconsistency. A more accurate data set will augment the performance of these systems.

ADVISOR'S APPROVAL: _____

VITA

Mesfin B. Taye

Candidate for the Degree of

Master of Science

Thesis: AUTOMATED EKG ANNOTATION WITH NEURAL NETWORKS

Major Field: Electrical Engineering

Biographical:

Personal Data: Born in Addis Ababa, Ethiopia

Education:

Received the B.S. degree from Bahir Dar University, Bahir Dar, Ethiopia, 2004, in Electrical Engineering

Received the M.S. degree from Addis Ababa University, Addis Ababa, Ethiopia, 2012, in Mathematics

Completed the requirements for the degree of Master of Science with a major in Electrical Engineering Oklahoma State University in December, 2016.