CLASSIFICATION OF ECG ARRYHTMIA BEATS WITH ARTIFICAL NEURAL NETWORKS

by

Seçil Zeybekoğlu

Bachelor Department of Electronics and Telecommunications, Yildiz Technical University, 2005

Submitted to the Institute of Biomedical Engineering
in partial fulfillment of the requirements
for the degree of
Master of Science
in
Biomedical Engineering (or Biomedical Science)

Boğaziçi University
September 2009

CLASSIFICATION OF ECG ARRYHTMIA BEATS WITH ARTIFICAL NEURAL NETWORKS

APPROVED BY:

Prof. Dr. Mehmed Özkan ....
(Thesis Advisor)

Assoc. Prof. Dr. H. Özcan GÜLÇÜR ....

Assistant Prof. Şenol Mutlu ....

DATE OF APPROVAL: ../../....

ACKNOWLEDGMENTS

I would like to express my gratitude to my thesis supervisor, Prof. Dr. Mehmed Özkan. Without his support and encouragement this thesis would not have been completed. His guidance, advices and feedbacks were the most important factors that helped me continue my studies.

I would like to thank Serdar Özmen, for all his help with Latex environment and his support with the documentation of this thesis.

I also would like to thank my sister, Senem Zeybekoğlu Sadri and her husband Hossein Sadri for their guidance as Academicians.

Finally, I would like to thank my parents for their support.

ABSTRACT

CLASSIFICATION OF ECG ARRYHTMIA BEATS WITH ARTIFICAL NEURAL NETWORKS

Electrocardiography (ECG) is a very useful noninvasive imaging method of the heart's electrical activity. Based on these recordings, a wide range of heart conditions can be diagnosed. These conditions may vary from minor to life threatening ones. Therefore, the scientists started to work on automatic systems that would detect any kind of abnormalities in the heart's electrical activity. These automated systems are expected to help patients monitor themselves or the clinicians monitor their patients for any kind of abnormalities. With the help of these automated systems, there is a big contribution to early, quick and efficient diagnose of the heart diseases. Based on this need, this thesis presents an automated arrhythmia detection system. The classification of beats is performed in a Graphical User Interface, namely Patient Monitoring GUI. Based on the user's selection, the GUI displays the type of beats that flow on the screen. In the background, the GUI uses an Artificial Neural Network (ANN) trained to classify the 7 different types of arrhythmias. During the training process of ANNs, the ECG recordings from MIT BIH Arrhythmia database are used as references. The arrhythmia samples are extracted from the database and preprocessed to create input sets to train ANNs. The Fourier Transforms of a predefined window of signals were taken as a feature extraction method. The training was performed in multiple steps in order to obtain best performing ANN that will be finally used by the Patient Monitoring GUI. The training of the ANNs was performed by using the Neural Network Toolbox in Matlab 2008b and the results were recorded to track the difference between the training attempts. The overall success rate of the best performing ANN was measured as 80%.

Keywords: Artificial Neural Network, ECG, Classification, Neural Network Toolbox, Fourier Transform, Arrhythmia

ÖZET

YAPAY SİNİR AĞLARIYLA EKG ARİTMİ SINIFLANDIRMASI

EKG (Elektrokardiyografi) kalbin elektriksel aktivitelerinin görüntülenmesini sağlayan çok kullanışlı noninvaziv bir görüntüleme metodudur. EKG kayıtlarına bakılarak kalbin sağlığıyla ilgili bir çok bilgi edinilebilir. Bu bilgiler ise küçük çapta veya hayati önem taşıyan bilgiler olabilir. Bunun üzerine bilim adamları kalbin elektriksel aktivitesindeki anormallikleri tanıyabilen otomatik sistemler üzerinde çalışmaya başlamışlardır. Bu sistemlerin gerek hastanın kendisini, gerek klinik ortamda hekimlerin hastalarını gözlemleyebilmelerine yardımcı olmaları beklenmektedir. Bu otomatik tanıyıcı sistemler kalp hastalıklarının erken ve efektif teşhisine büyük katkıda bulunmaktadır. Buna dayanarak bu tezde otomatik aritmi tanıyıcı bir system öne sürülmektedir. Aritmilerin tanınması bir kullanıcı arayüzü içinde gerçekleştirilmektedir. Hasta gözlem arayüzünde kullanıcının seçimine göre, ekranda akan sinyalde hangi aritmi çeşidinin olduğu gösterilmektedir. Arka planda ise kullanıcı arayüzü 7 farklı aritmi çeşidini birbirinden ayırt edebilmek için daha önceden eğitilmiş bir yapay sinir ağını kullanmaktadır. Yapay sinir ağlarının eğitilmesi sırasında MIT BIH Aritmi veritabanında bulunan EKG kayıtları kullanılmıştır. Aritmi örnekleri YSA'nı eğitirken kullanılacak eğitim kümesi oluşturacak şekilde işlemden geçirilmiştir. Bu işlemler sırasında önceden belirlenmiş bir uzunlukta ana kayıttan çıkarılan sinyal Fourier transformundan geçirilmiştir. YSA'nın eğitilmesi arayüzde kullanılacak olan en iyi YSA'nın üretilmesi amaçlanarak birden çok adımda gerçekleşirilmiştir. YSA'nın eğitilmesi MATLAB 2008b programının Neural Network Toolbox u kullanılarak gerçekleştirilmiştir. Bu eğitimler sırasındaki çıktılar tüm eğitim denemelerinin karşılaştırılması amacıyla kaydedilmiştir. Bu çalışmada en iyi YSA'nın %80 performansla çalıştığı gözlenmiştir.

Anahtar Sözcükler: Yapay Sinir Ağı, EKG, Sınıflandırma, Neural Network Toolbox, Fourier Dönüşümü, Aritmi

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

v_j	total inputs to neuron j
w_{ji}	weight matrix between neuron j and i
y_j	output of neuron i
e_j	error signal at the output of neuron j
d_j	desired signal at the output of neuron j
ε	instantaneous sum of squared errors
φ	activation function
δ_j	local gradient of neuron j
Δw_{ji}	Changes in the weights
η	learning rate parameter

LIST OF ABBREVIATIONS

AFIB	Atrial Fibrillation
ANN	Artificial Neural Network
APC	Atrial Premature Contraction
AV	Atrioventricular
ECG	Electrocardiography
FFT	Fast Fourier Transform
GUI	Graphical User Interface
L	Left Bundle Branch Block
LBB	Left Bundle Branch Block
MIT-BIH	Massachusetts Institute Of Technology-Beth Israel Hospital
MLII	Modified Limb Lead II
N	Normal
PVC	Premature Ventricular Contraction
R	Right Bundle Branch Block
RBB	Right Bundle Branch Block
RN	Right Branch Block + Normal
SA	Sinoatrial
VT	Ventricular Tachycardia
VTN	Ventricular Tachycardia + Normal

1. Introduction

The scope of this thesis is to make an automated system to detect heart abnormalities.

The most common symptoms for heart diseases are chest pain, dizziness, pain in upper side of the body. A heart attack starts slowly with these symptoms. A patient may listen himself for any kind of symptoms, however, the symptoms even may come and go. Detecting heart diseases on time saves many lives and many times. At this point, consider that a patient has a monitoring device where the heart beats are monitored and the patient is warned in the presence of any kind of abnormalities. Therefore, having automated systems which can detect heart abnormalities is getting more and more important in today's clinical environment.

Many researches have been performed previously to obtain successful ECG beat classifiers with several feature extraction and classification methods and different beat types.

S. Isaac Niwas et al. [1] Classified 9 different beat types from MIT BIH Arrhythmia database which were Left Bundle Branch Block, Right Bundle Branch Block, Atrial Premature Beat, Supraventricular Ectopic Premature Beat, Premature Ventricular Contraction, Atrial Fibrillation, Ventricular Fibrillation, Sick Sinus Syndrome, Fusion of Ventricular and Normal Beat.

The overall accuracy of their Neural Network which was using Backpropagation Algorithm was 99.02 %. In their study, baseline wandering due to power line interference was removed by using 2 median filters to the signal. Feature sets were based on heartbeat intervals, RR intervals and spectral entropy of the ECG signal. RR sequences were detected by using a heartbeat detection algorithm. Tables which summarize the overall performance of their work can be seen in Appendix A.

Tamer Ölmez et al. [2] studied and compared different networks which includes NetGA, MLP and Kohonen networks. MLP is based on a supervised learning and Kohonen and GAL are based on competitive learning. They extracted 7 beats from MIT BIH for classification which were as follows: Normal, Left Bundle Branch Block, Premature Ventricular Contraction, Paced Beat, Aberrated Atrial Premature Beat, Right Bundle Branch Block, Ventricular Escape Beat. In their study, after the R peak was detected (by using an amplitude threshold method), the amplitudes were normalized, a window (data length of 256) containing a period around the R peak was formed. The frequency components of the signal in the window was used for feature set. All possible noises like 50 Hz power line noise, muscle noise and base-line wander were removed by applying a preprocessing stage. The summary of their comparison can be found in Appendix A.

Another article from the same authors [3] also investigates two different feature extraction methods, Fourier and Wavelet analysis comparatively with a hybrid neural network.

Branko Celler classified 7 different beats by using power spectral density estimate [4]. He used two discrete wavelets, Daubechies wavelet of order 10 and Symlet wavelet of order 8. ECG recordings were bandpass filtered with a linear phase filter with 3DB points at 0.5Hz and 40 Hz. The isopotential value was subtracted and the beat was centered in a window of 125 samples using calculated QRS onset and offset information. The data window was multiplied with a Hanning window of the same length to remove edge effects. Power spectral density estimates were obtained for each X Y and Z beats using standard FFT. The overall classification accuracy in his work was 68%.

Murale Kanapathipillai et al. [5] used Morlet Wavelet transform to extract features from MIT BIH Arrythmia Database and the overall accuracy of the classification with Neural Network was 70%. In this study the author only classified the signals as normal and abnormal.

In their study, Ming-Yao Yang et al. [6] achieved 97.77% accuracy in classifying 7 different types of arrythmias extracted from MIT BIH Arrthmia database. The beats included Left Bundle Branch Block, Right Bundle Branch Block, Premature Ventricular Contraction, Wolff-Parkinson-White Syndrome, Myocardial Ischemia and Myocardial Injury. 520 ECG feature patterns were collected and 160 of these data were used in training. The author first performed a baseline correction by using discrete least squares. Then the feature extraction method was performed with Wavelet transform (Dyadic wavelet). After the ECG features have been extracted by WT, following 12 features were collected to be used in training the ANN: P duration, PR interval, QRS duration, S duration, T duration, QT interval, P amplitude, R amplitude T amplitude ST segment level and QT interval area. Two types of learning, supervised and unsupervised learning methods were studied by these authors.

Susan Ciarroca Lee [7] used a translation invariant method to create a feature set for the Backpropagation Neural Network input set. Short pieces from the original ECG rhythm segment were extracted. These pieces were repeated to produce a facsimile of the original signal. The window is guaranteed to have at least one QRS complex. With translation invariance method, the features of 50 points for each window are extracted. Translation invariance is introduces by constraining the weights on the first order inputs to be independent of input position and the second order weights to depend only on the difference between indices. She classified 3 beat types : Normal Beat, Ventricular Tachicardia, Ventricular Fibrillation. The summary of the results of this study can be seen in Appendix A.

Victor-Emil Neagoe et al. used The Principle Component Analysis and Discrete Cosine Transform in their study as a feature extraction method [8]. Again Wei Jiang, used principle component analysis to establish his work on classifying the ECG signals [9].

By looking at the previous studies, one can see that the R peak detection method is used in many studies as one of the first steps of the feature extraction method. Detecting the R peak requires the beat of interest to be morphologically identifiable.

However, in detection of arrhythmia, the morphology of the heart beat is outside of normal and makes the data unreliable. Therefore, to overcome this fact, a method independent of the morphology should be investigated.

In this thesis, the arrhythmia beats are classified by using feedforward back propagation neural network structure. The Fourier Transform of the arrhythmia beats are used as the method of feature extraction. During the data extraction phase morphological components were not detected.

The explanations for the key points of the study will be provided prior to Methodology section as an introduction to the study, which are as follows:

1. ElectroCardiography;
2. Arrythmia;
3. Neural Networks;
4. Back Propagation Learning Algorithm;
5. MIT BIH Database.

1.1 Electrocardiography(ECG)

The Electrocardiography is a very important image of the electrical activity of the heart. Many useful information are gathered from tracing the ECG signals. The ECG records the electrical activity that results when the heart muscle cells in the atria and ventricles contract. A normal sinus beat with PQRST peaks is shown in Figure 1.1.

"Atrial contractions (both right and left) show up as the P wave. Ventricular contractions (both right and left) show as a series of 3 waves, Q-R-S, known as the QRS complex. The third and last common wave in an ECG is the T wave. This reflects the

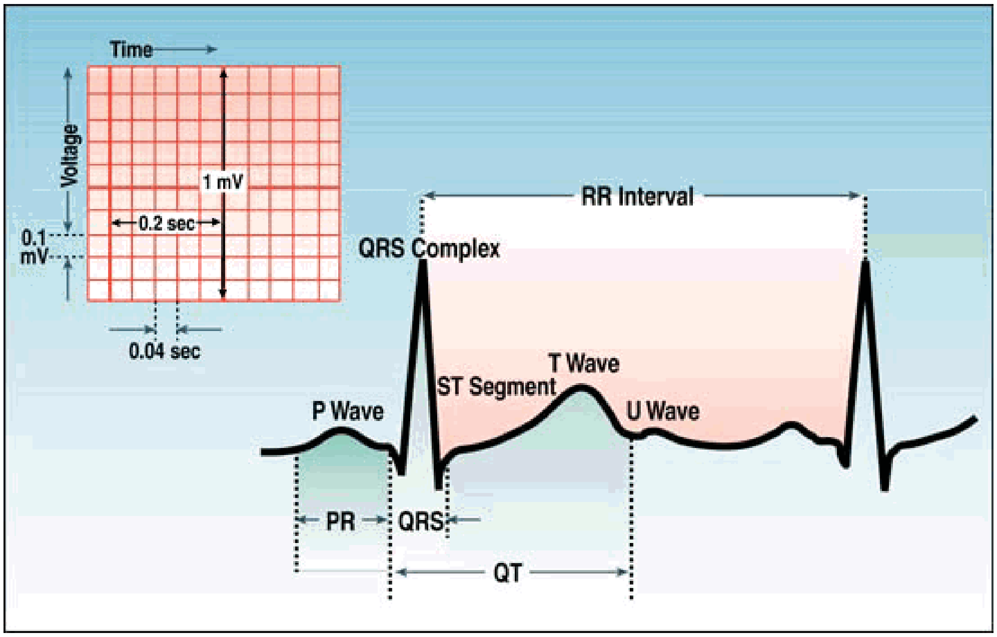


Figure 1.1 Normal Sinus Beat With PQRST Peaks [10]

electrical activity produced when the ventricles are recharging for the next contraction (repolarizing). Interestingly, the letters P, Q, R, S, and T are not abbreviations for any actual words but were chosen many years ago for their position in the middle of the alphabet. The electrical activity results in P, QRS, and T waves that have a myriad of sizes and shapes. When viewed from multiple anatomic-electric perspectives (that is, leads), these waves can show a wide range of abnormalities of both the electrical conduction system and the muscle tissue of the heart's 4 pumping chambers." [10]

The electrical activity produced by the contraction of the chambers are shown in Figure 1.2.

"A typical ECG tracing of a normal heartbeat (or cardiac cycle) consists of a P wave, a QRS complex and a T wave. A small U wave may be visible in 50 to 75% of ECGs. The baseline voltage of the electrocardiogram is known as the **isoelectric line**. Typically the isoelectric line is measured as the portion of the tracing following the T wave and preceding the next P wave." [11]

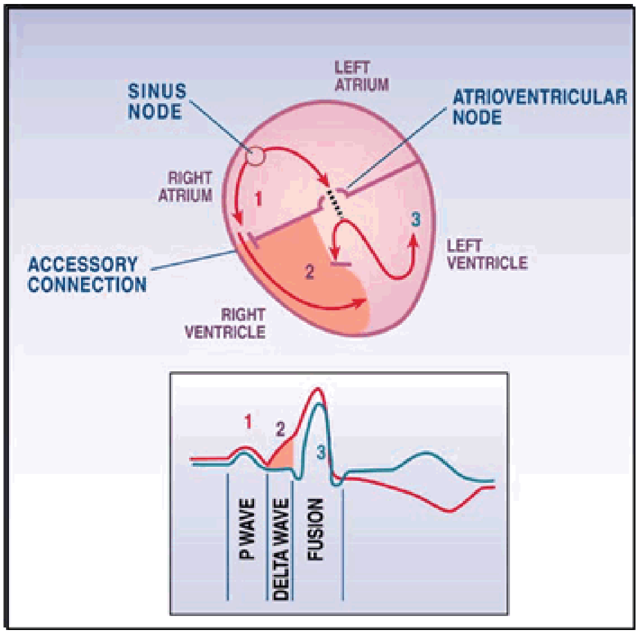


Figure 1.2 Electrical Activity of the Chambers [10]

1.2 Arrythmia

Arrhythmia is a term that is used to describe an abnormal electrical pattern within the heart

"The heart is divided into two upper chambers (atria) and two lower chambers (ventricles) as shown in Figure 1.2. Before the chambers can contract, they must receive an electrical signal, the same way that you must plug in an electrical appliance before the motor will work. The electrical signal is picked up by electrodes on the chest and displayed as a waveform call an Electrocardiogram or ECG, as shown in Figure 1.1. Normally, the electrical signal takes on a very predictable pattern. The upper chambers send the first electrical signal, followed by a signal from the lower chambers. The electrical signal begins in the right atrium in a special area called the "sinoatrial node". When the electrical pattern of the heart takes on a normal sequence, the sinoatrial node (or sinus node) produces the first wave. This wave is followed by a signal from the lower chambers. The term "normal rhythm" or "normal sinus rhythm" is used to describe the normal electrical waveform pattern. Normal sinus rhythms are regular. A normal heart rate is 60-100 beats per minute. For many reasons, different areas of the heart can initiate the electrical message. Beats that start from a location other than the sinus node are called "ectopic beats" (outside the normal). They are easy to recognize because the ECG pattern looks different. An ectopic beat is also called an "arrhythmia" [12].

1.3 Artificial Neural Networks

The scientists have inspired by the method that the brain learns, remembers and decides so that they invented a mathematical model which has synaptic information storage system in artificial neurons which are called weights. An artificial neural network system is a biologically inspired system consisting of artificial neurons connected to each other with weights. Many researches have been made to improve this highly interconnected neuron system in order to solve specific problems. These problems may be pattern recognition, classification or estimation.

The most significant feature of the artificial neural networks is the ability of being trained. The training inputs are presented to the ANN iteratively and the weights between the neurons are updated accordingly until the ANN starts to produce the best expected results. The procedure used during training is the learning algorithm. The method of updating the weights highly depend on the learning algorithm. The ANNs are beneficial to work with as they are able to provide following features [13]:

1. **Nonlinearity**: ANNs can be both linear or nonlinear. Nonlinearity is highly important where the input signals are nonlinear.

2. **Input Output mapping**: By using the supervised learning method, the weights of the ANN are updated by computing the error between the output calculated during the feedforward calculation and the desired output of the ANN at time t. The training of the ANN is repeated for many examples in the set of training until the network reaches a steady state where there are no further significant changes in the synaptic weights. Therefore, the ANN would learn by constructing

an input output mapping in the end.

3. **Adaptivity**: Neural networks have the ability to adapt their synaptic weights to the changes in their surround environment. The ANN can be developed to adapt itself in real time if it is operating in a non stationary environment.
4. **Evidential response**: The neural network can also provide information about the confidence level of the decision it make.
5. **Contextual information**: Every neuron in the network is potentially affected by the activity of all other neurons in the network.
6. **Fault tolerance**: The ANNs has the ability of robust computation, however in order to be assured that the network is fault tolerant, it is useful to take corrective measures while designing the algorithm that would train the network.

Artificial neural networks consist of neurons in their architectures. A neuron is the simplest information processing unit of an artificial neural network which has three basic components described below:

1. A set of synaptic weights. Specifically, a signal x_j at the input of synapse j connected to the neuron k is multiplied by the synaptic weight w_{kj}.
2. An adder that performs the summation of the inputs calculated by the weights.
3. An activation function for limiting the amplitude of the output of a neuron

Figure 1.3 shows a nonlinear model of a neuron, S represents the number of neurons in the layer whereas R represents the number of elements in the input vector.

Eq. 1.1 and Eq. 1.2 describe the activity on one neuron:

$$v_j(n) = \sum_{i=0}^{m} w_{ji}(n) y_i(n); \tag{1.1}$$

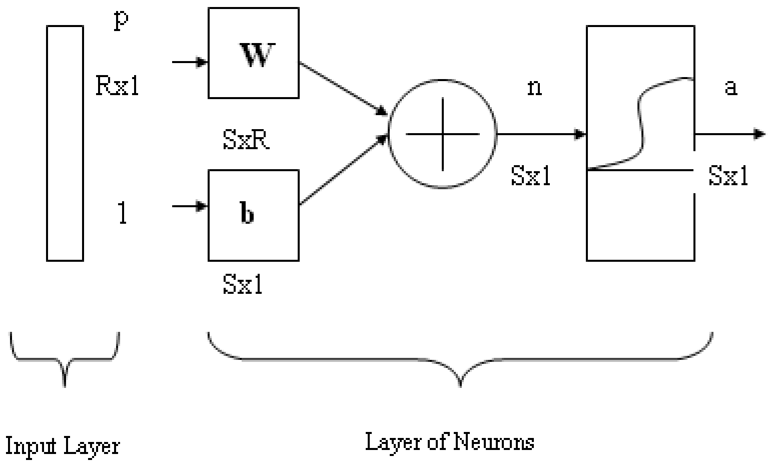


Figure 1.3 Single Layer Neuron Structure

$$y_j(n) = \varphi_j(v_j(n)); \tag{1.2}$$

where y_i is the input signals, w_{ji} is the synaptic weights of neuron i, v_j is the linear combiner output, φ is the activation function.

1.4 Backpropagation Learning Algorithm

In this thesis, the Artificial Neural Networks were trained by using Backpropagation Learning Algorithm.

In the application of Backpropagation algorithm, there are two main calculation processes [13]. First is a feedforward process where all the inputs and the weights are calculated, the sum of this calculation is presented to activation functions and finally the output of the entire network is found for the presented input example and the set of synaptic weights. The backward process starts from the output neuron by calculating the error by comparing the output calculated with the desired network response and passes this error information backwards to the inner layers by a layer by layer computation. The error is propagated backward with the local gradients.

The Backpropagation algorithm is explained step by step with the equations from Eq. 1.3 to Eq. 1.15 [13].

The error calculated in the output neuron $\boldsymbol{j}$ at iteration $\boldsymbol{n}$ is defined as follows:

$$e_j(n) = d_j(n) - y_j(n). \tag{1.3}$$

Instantenous sum of output errors is calculated by using this error:

$$\varepsilon(n) = \frac{1}{2}\sum_{j \in C} e_j^2(n). \tag{1.4}$$

The net input of the activation function of neuron $\boldsymbol{j}$ is calculated as:

$$v_j(n) = \sum_{i=0}^{m} w_{ji}(n) y_i(n); \tag{1.5}$$

where $\boldsymbol{m}$ is the total number of inputs applied to neuron $\boldsymbol{j}$.

The output of neuron $\boldsymbol{j}$ is calculated as:

$$y_j(n) = \varphi_j(v_j(n)). \tag{1.6}$$

The partial derivative $\partial\varepsilon(n)/\partial w_{ji}(n)$ is calculated with the chain rule, as follows:

$$\frac{\partial\varepsilon(n)}{\partial w_{ji}(n)} = \frac{\partial\varepsilon(n)}{\partial e_j(n)}\frac{\partial e_j(n)}{\partial y_j(n)}\frac{\partial y_j(n)}{\partial v_j(n)}\frac{\partial v_j(n)}{\partial w_{ji}(n)}; \tag{1.7}$$

Differentiating Eq. 1.4 with respect to $e_j(n)$, we get:

$$\frac{\partial\varepsilon(n)}{\partial e_j(n)} = e_j(n). \tag{1.8}$$

Differentiating Eq. 1.3 with respect to $y_j(n)$, we get:

$$\frac{\partial e_j(n)}{\partial y_j(n)} = -1. \tag{1.9}$$

Differentiating Eq. 1.6 with respect to $v_j(n)$, we get:

$$\frac{\partial y_j(n)}{\partial v_j(n)} = \varphi_j'(v_j(n)). \tag{1.10}$$

Finally, differentiating Eq. 1.5 with respect to $w_{ji}(n)$, we get:

$$\frac{\partial v_j(n)}{\partial w_{ji}(n)} = y_i(n)). \tag{1.11}$$

Therefore, we can define $\partial\varepsilon(n)/\partial w_{ji}(n)$ as follows:

$$\frac{\partial\varepsilon(n)}{\partial w_{ji}(n)} = -e_j(n)\varphi_j'(v_j(n))y_i(n). \tag{1.12}$$

Updates to the weights$w_{ji}(n)$, which is defined as $\Delta w_{ji}(n)$ is defined by the *delta rule*:

$$\Delta w_{ji}(n) = -\eta\frac{\partial\varepsilon(n)}{\partial w_{ji}(n)}. \tag{1.13}$$

Accordingly, the use of Eq. 1.12 in Eq. 1.13 yields:

$$\Delta w_{ji}(n) = \eta\delta_j(n)y_i(n); \tag{1.14}$$

where the local gradient of neuron ***j*** $\delta_j(n)$ is defined by:

$$\delta_j(n) = -\frac{\partial\varepsilon(n)}{v_j(n)} = -\frac{\partial\varepsilon(n)}{e_j(n)}\frac{\partial e_j(n)}{\partial y_j(n)}\frac{\partial y_j(n)}{v_j(n)}. = e_j(n)\varphi_j'(v_j(n)) \tag{1.15}$$

Depending on where in the network neuron ***j*** is located, there are two distinct cases. In case 1, neuron ***j*** is an output node, where it is supplied with a desired response of its own. In case 2, neuron ***j*** is a hidden node. Even though hidden nodes are not visible and can not be directly measured, they are affected by any error made at the output of the network.

Case 1 Neuron j is an output node: The error signal in the output node can be calculated as:

$$e_j(n) = d_j(n) - y_j(n). \tag{1.16}$$

Therefore, the local gradient of output neuron j is calculated as:

$$\delta_j(n) = e_j(n)\varphi_j'(v_j(n)). \tag{1.17}$$

Figure 1.4 explains the signal flow of the backpropagation algorithm with 2 neurons.

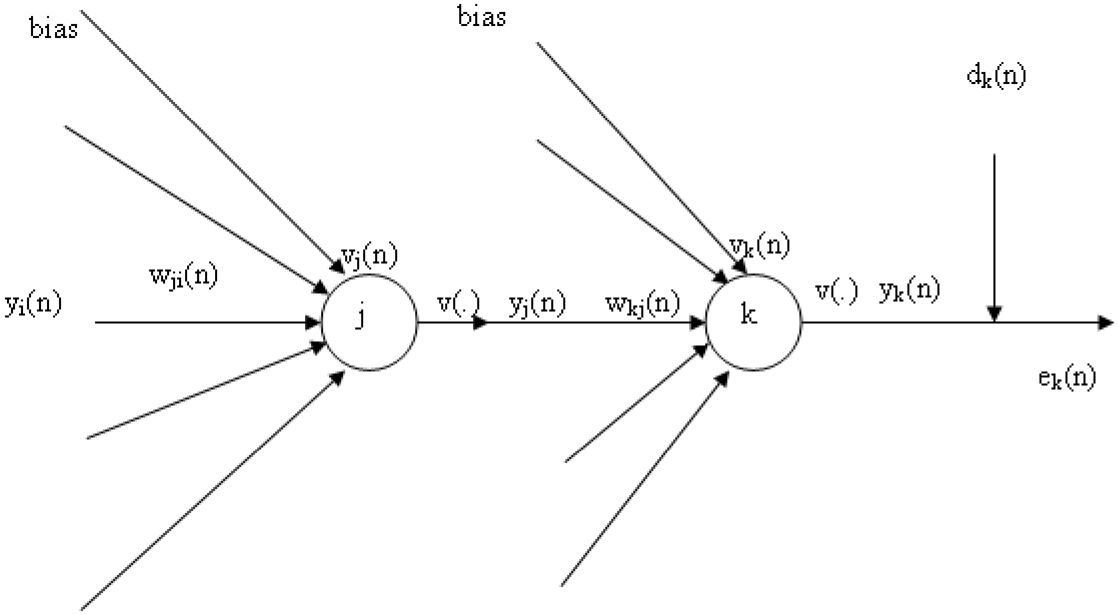


Figure 1.4 Signal Flow of Back Propagation Algorithm

Case 2 Neuron j is a hidden node: When neuron j is located in a hidden layer of the network, there is no specified desired response for that neuron. Therefore, the error signal of a hidden node neuron should be defined in terms of the error signals of 32 to which that hidden neuron is directly connected [13]. Therefore we may redefine the local gradient $\delta_j(n)$ for hidden neuron j as:

$$\delta_j(n) = -\frac{\partial\varepsilon(n)}{y_j(n)}\frac{\partial y_j(n)}{\partial v_j(n)} = -\frac{\partial\varepsilon(n)}{\partial y_j(n)}\varphi_j'(v_j(n)). \tag{1.18}$$

To calculate the partial derivative $\partial\varepsilon(n)$ / $\partial y_j(n)$, we may proceed as follows:

$$\varepsilon(n) = \frac{1}{2}\sum_{k\in C} e_k^2(n); \tag{1.19}$$

where neuron k is an output node. Differentiating Eq. 1.19 with respect to $y_j(n)$, we get:

$$\frac{\partial\varepsilon(n)}{\partial y_j(n)} = \sum_k e_k \frac{\partial e_k(n)}{\partial y_j(n)}. \tag{1.20}$$

Next we use the chain rule for the partial derivative $\frac{\partial e_k(n)}{\partial y_j(n)}$ and rewrite Eq. 1.20 in the equivalent form:

$$\frac{\partial\varepsilon(n)}{\partial y_j(n)} = \sum_k e_k \frac{\partial e_k(n)}{\partial v_k(n)}\frac{\partial v_k(n)}{\partial y_j(n)}. \tag{1.21}$$

The error in the output node is:

$$e_k(n) = d_k(n) - y_k(n) = d_k(n) - \varphi_k(v_k(n)). \tag{1.22}$$

Therefore;

$$\frac{\partial e_k(n)}{\partial v_k(n)} = -\varphi_k'(v_k(n)). \tag{1.23}$$

The net input for neuron $\boldsymbol{k}$ is:

$$v_k(n) = \sum_{j=0}^{m} w_{kj}(n) y_j(n). \tag{1.24}$$

Differentiating Eq. 1.24 with respect to $y_j(n)$, we get:

$$\frac{\partial v_k(n)}{\partial y_j(n)} = w_{kj}(n). \tag{1.25}$$

By using Eq. 1.25 and Eq. 1.23 in Eq. 1.21, we get the desired partial derivative:

$$\frac{\partial \varepsilon(n)}{y_j(n)} = -\sum_k e_k(n) \varphi_k'(v_k(n)) w_{kj}(n) = -\sum_k \delta_k(n).w_{kj}(n) \tag{1.26}$$

Finally, using Eq. 1.26 in Eq. 1.14, we get the *back-propagation formula* for the local gradient $\delta_j(n)$ as described:

$$\delta_j(n) = \varphi_j'(v_j(n)) \sum_k \delta_k(n) w_{kj}(n). \tag{1.27}$$

The correction $\Delta w_{ji}(n)$ applied to the synaptic weight connecting neuron $\boldsymbol{i}$ to neuron $\boldsymbol{j}$ is defined as follows:

$$\left(\begin{array}{c} Weight \\ correction \\ \Delta w_{ji}(n) \end{array} \right) = \left(\begin{array}{c} learning- \\ rate\,parameter \\ \eta \end{array} \right) \cdot \left(\begin{array}{c} local \\ gradient \\ \delta_j(n) \end{array} \right) \cdot \left(\begin{array}{c} inputsignal \\ of neuronj \\ y_i(n) \end{array} \right) .$$

In this thesis, the sigmoid function is used as the activation function.

$$f(x) = \frac{1}{1 + exp(-x)}). \tag{1.28}$$

Activation function must be a continuous function so that its derivative can be used during weight adaptation process. The derivative of the sigmoid function is given in Eq. 1.29:

$$f'(x) = \frac{exp(-x)}{(1 + exp(-x))^2}. \tag{1.29}$$

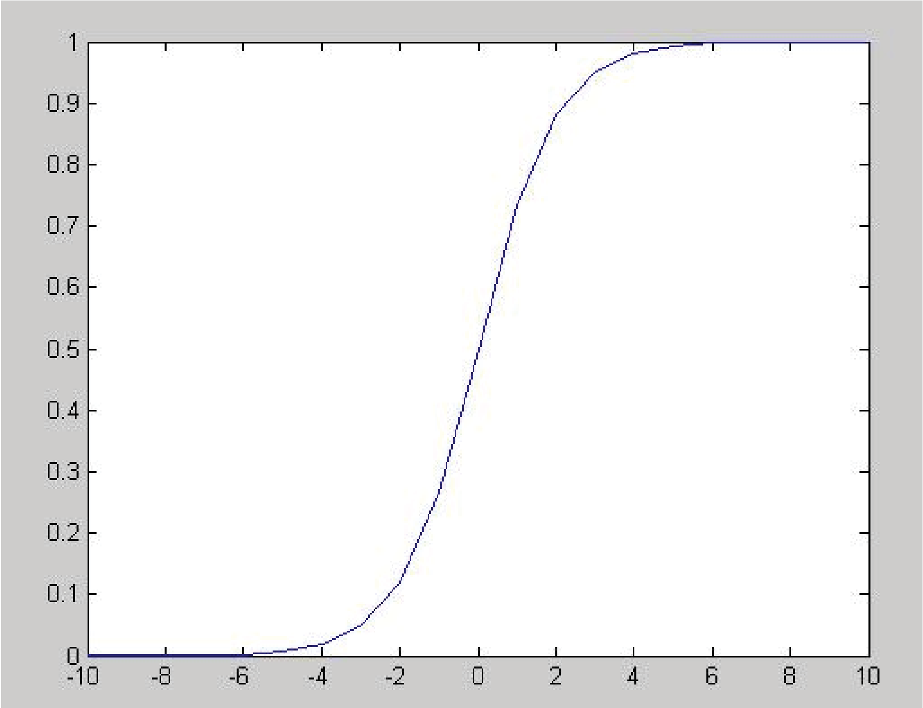


Figure 1.5 Sigmoid Function

1.5 MIT-BIH ECG Arrythmia Database

The data set that is used both in training the network and test is extracted from the recordings of MIT BIH Arrythmia database [14].

"This database consists of 48 annotated records obtained from 47 studied by the Arrythmia Laboratuary of Beth Israel Hospital in Boston between 1975-1979. The database contains 23 records (the '100' series) chosen at random from a set of over 4000 24 hours Holter tapes and 25 records (the '200' series) selected from the same set to include a variety of rare but clinically important phenomena which wouldn't be well-represented by a small random sample.

The first group is intended to serve as a representative sample of the variety of waveforms and artifact that an arrhythmia detector might encounter in routine clinical use. Records in the second group were chosen to include complex ventricular, junctional, and supraventricular arrhythmias and conduction abnormalities The subjects were 25 men aged 32 to 89 years, and 22 women aged 23 to 89 years.

In most records, the upper signal is a modified limb lead II (MLII), obtained by placing the electrodes on the chest. The lower signal is usually a modified lead V1 (occasionally V2 or V5, and in one instance V4); as for the upper signal, the electrodes are also placed on the chest.

Each record in the MIT-BIH Arrythmia database is slightly over 30 minutes in length. Each signal file contains two signals sampled at 360 Hz. 'Header' files include information about the leads used, the patient's age, sex and medications used. The reference annotation files include beat, rhythm and signal quality annotations. Each of the beats was manually annotated by at least two cardiologists working independently; their annotations were compared; consensus on disagreements was obtained; and the reference annotations files were prepared. The database is delivered via a CD ROM and it is also available online." [14]

An example recording output from one of the 100 series can be seen in Figure 1.6. The first 20 seconds of the recording is shown.

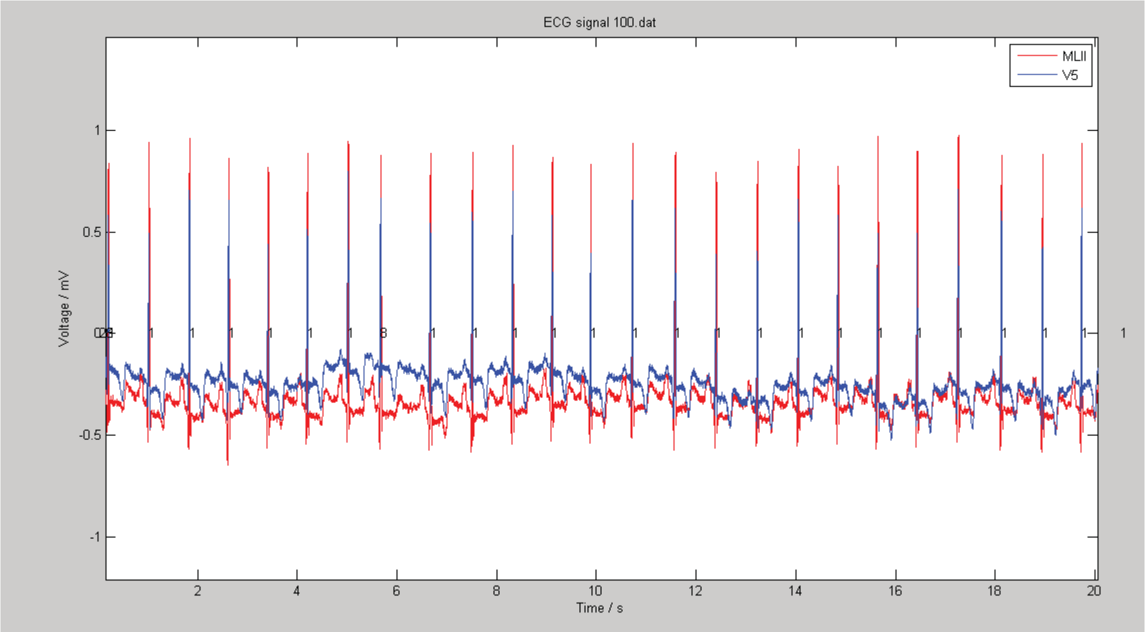


Figure 1.6 Recording Output Example

The details of the recordings, the types of arrhythmias and the types of annotations can be found in the tables in the Appendix B. The statistical summary of the recordings can also be seen in the Appendix section.

1.6 How the Data is read

The ECG signals were in a 2-1-2 bit format and there are 3 files associated with the same recording. First file is the .dat file which contains the signal itself, the second file is the .hea header file which contains many information like leads used in the recording, and the last .atr file contains the annotations used in plotting which marked the beats one by one according to the beat types. The matlab program to read the ECG recordings is provided at [14].

The ECG read program uses the **header**, **annotation** and the **dat** file to read the two lead ECG recording with the annotations. In this thesis study, this program delivered by [14] was used to view the ECG recordings.

Recording 100 was plotted and shown in Figure 1.7.

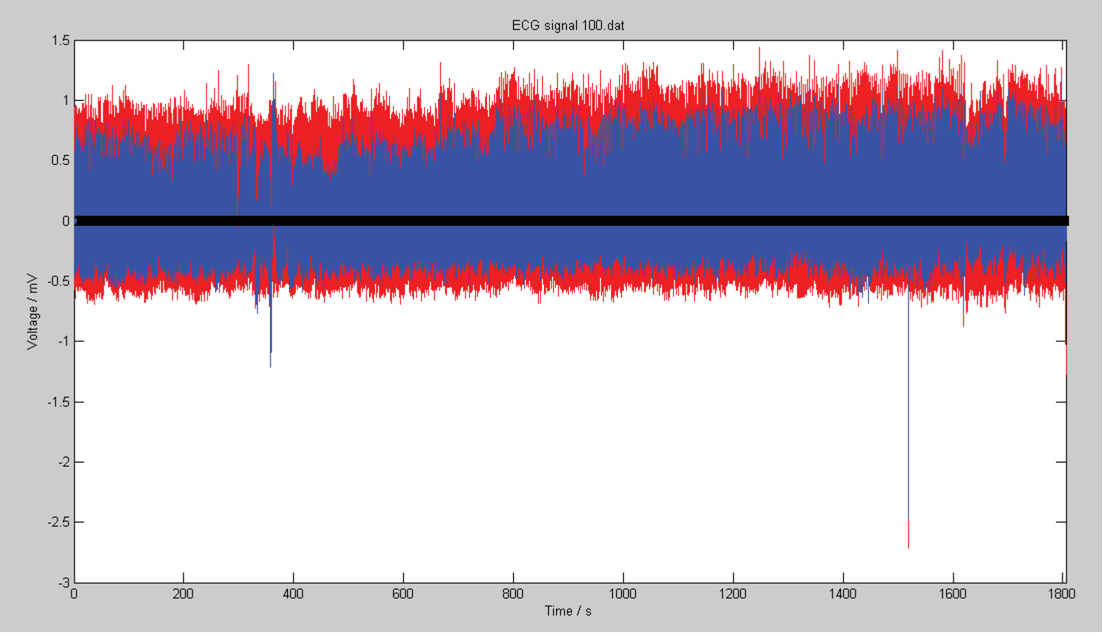


Figure 1.7 ECG Signal View of Patient #100

Figure 1.8 an example normal sinus rhythm with annotations. The zoomed view of the following signal part belongs to Recording 100.

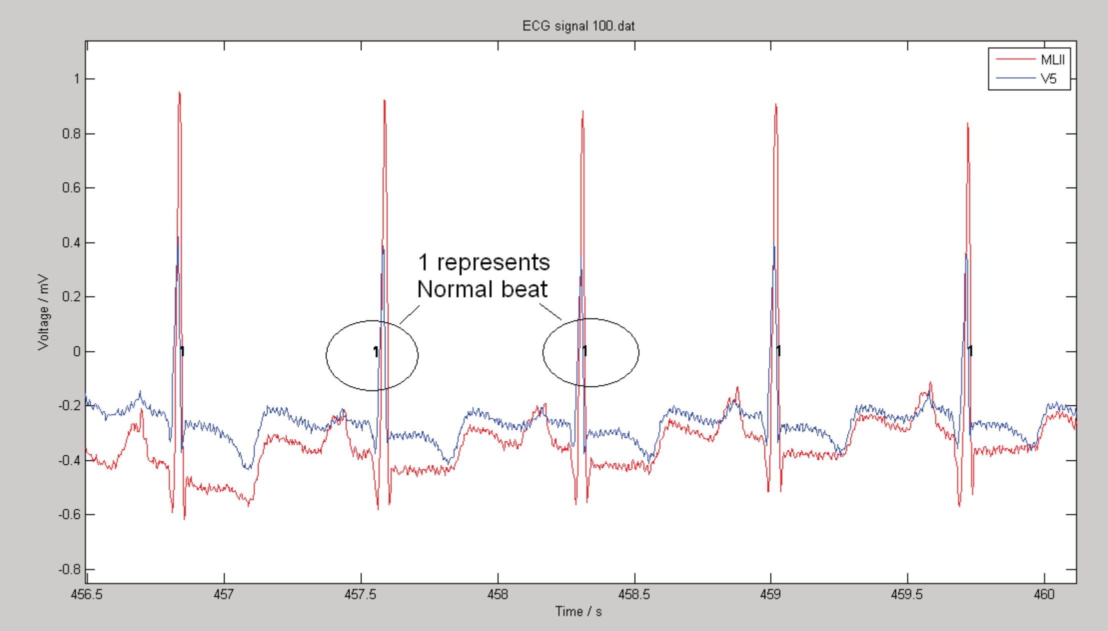


Figure 1.8 Normal Sinus Rhythm with Annotations

In addition to this Matlab program, the ECG recordings can also be viewed online at [14].

In this Web page, from the input drop down menu at the top, users can select the MIT BIH Arrythmia database and select the recording to be viewed. Both leads with annotations can be viewed by navigating through the signal by using the control buttons in the web page. Please see the example screenshot of the signal as shown in Figure 1.9.

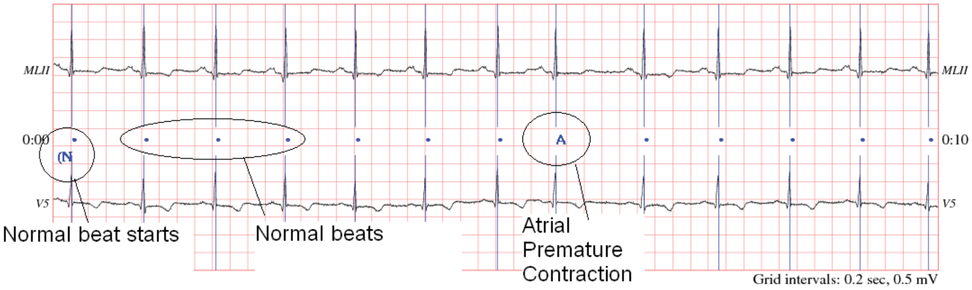


Figure 1.9 Sample View From MIT BIH Web Site [14].

The data used for this study is taken from the recordings by using the matlab program and the sample annotated beats were extracted carefully by comparing both sources.

To see how the ECG data was extracted, please see the methodology section.

2. Methodology

To accomplish the classification aimed in this study, first, the ECG arrhythmia signals were extracted from the MIT BIH Arrhythmia database which can be found online at [14].

Secondly, using these extracted samples, feedforward neural networks with different architectures has been trained to classify the ECG Arrhythmia types. Multiple network training steps were performed by changing several parameters explained in later sections . All the networks were tested both with a test set which was different from the training set and extracted again from the same MIT BIH database, and, the training set itself. The network that classified the best was chosen to be the final network to classify the beat types.

Lastly, a simple patient monitoring GUI with basic control functions was designed in matlab. With this GUI users are able to select an ECG signal and view the arrhythmic signals if there are any while the GUI runs.

2.1 Data Used in the Study

In this study, MIT BIH Arrhythmia database was used to extract all the data to train the ANN. This database contains many different types of Arrhythmias. For classification purposes during this study,the following types of arrhythmias were chosen to train the ANN: Normal, Atrial Fibrillation, Ventricular Tachycardia, Right Branch Block, Left Branch Block.

Figure 2.1 shows the processes followed in order to get the final training data in block diagrams.

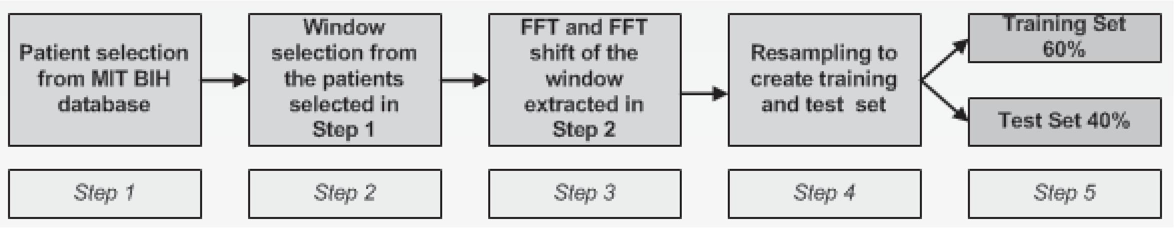

Patient selection from MIT BIH database
Window selection from the patients selected in Step 1
FFT and FFT shift of the window extracted in Step 2
Resampling to create training and test set
Training Set 60%
Test Set 40%
Step 1
Step 2
Step 3
Step 4
Step 5

Figure 2.1 Process Flow

2.1.1 Patient Selection

5 types of beats were chosen to be used as the ECG data in this study. They were placed in a predefined sized window and extracted from the predetermined recordings. The normal beats were extracted from the recordings: 100, 108, 205, 121, 222, 209, 201, 101, 103, 105, 113, 200, 210, 203, 213 and 212. The ventricular tachycardia beats were extracted from the recordings 210, 213, 106 and 200 The Right Branch Block beats were extracted from the recordings 212, 231 and 232. The Left Branch Block betas were extracted from recordings 109,111, 201, 207 and 214. The atrial fibrillation beats were extracted from the recordings 201, 202, 203, 219, 221 and 222. Table 2.1 shows which recordings were used to extract the example beat types as a summary. The beats in Table 2.1 are as follows:

N: Normal beat;

VT: Ventricular Tachicardia;

VT+N: Ventricular Tachycardia and Normal beats in the same window;

R: Right Branch Block;

R+N: Right Branch Block and Normal beat in the same window;

L: Left Branch Block;

AFIB: Atrial Fibrillation.

The ECG data extraction was performed by placing the ECG beat of interest into a certain predefined window and extracting the entire window. The entire extractions were performed from one single lead MLII.

Table 2.1
Recordings Used to Extract Beat Types

Normal	VT	VT+N	R	R+N	L	AFIB
100	210	210	212	212	109	201
108	213	213	231	231	111	202
205	106	106	232	232	201	203
121	200	200			207	219
222	203	203			214	221
209	214	214				222
201	215	215				
101	217	217				
103	221	221				
105	223	223				
113	233	233				
200						
210						
203						
213						
212						

An example window extracted from recording 100 which contains the Normal beats is shown in Figure 2.2.

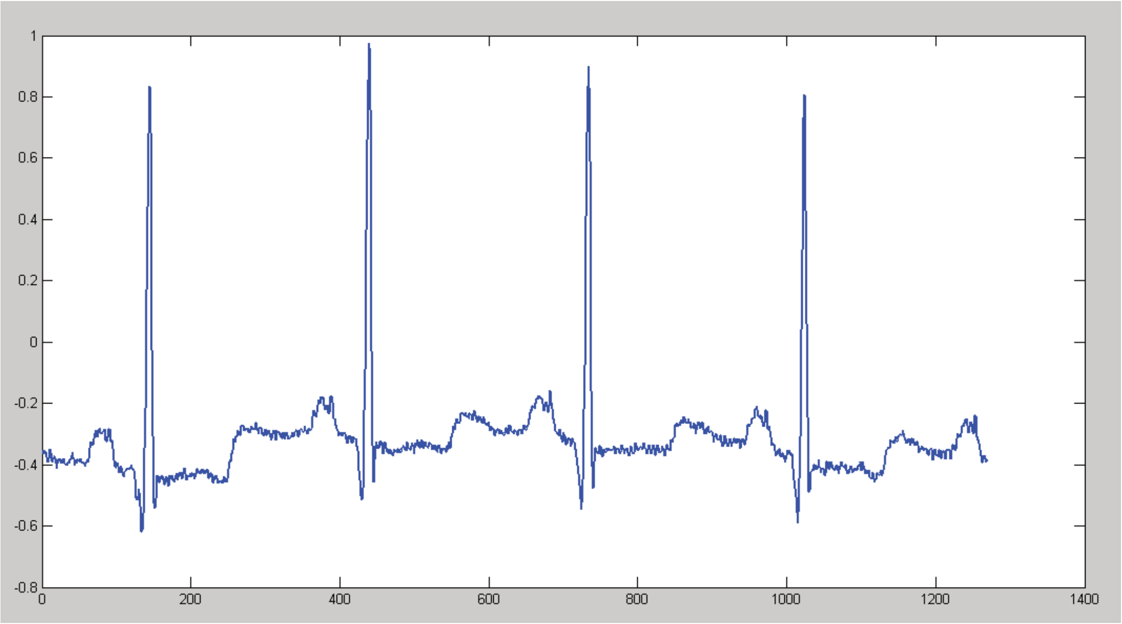


Figure 2.2 Normal Beat Extracted From Patient #100

All the windows extracted from the recordings are kept in the same size in order to handle the data easily. The window contains 1270 data points.

The data to be used for training and testing the neural network is extracted from these fixed window pre-extracted signals. The Fourier transform of these extracted beats were taken and the transformed signals were processed to create the training and test sets. The Fourier transform shows the frequency components of a signal and the time component would no longer be available. Therefore the time dependent heart rate information that would differ from patient to patient is an important point for this study.

2.1.2 Window Selection

The recordings in the MIT BIH database were fully investigated by cardiologists and all the beat types in all the recordings were fully annotated. By using these annotations, the exact times of the beat types of interest were detected in the recordings.

The number of annotated beats and the points of interest were given in [14]. The beats of interest were viewed again by using [14].

Figure 2.3 shows an example for Ventricular Tachicardia in recording 210.

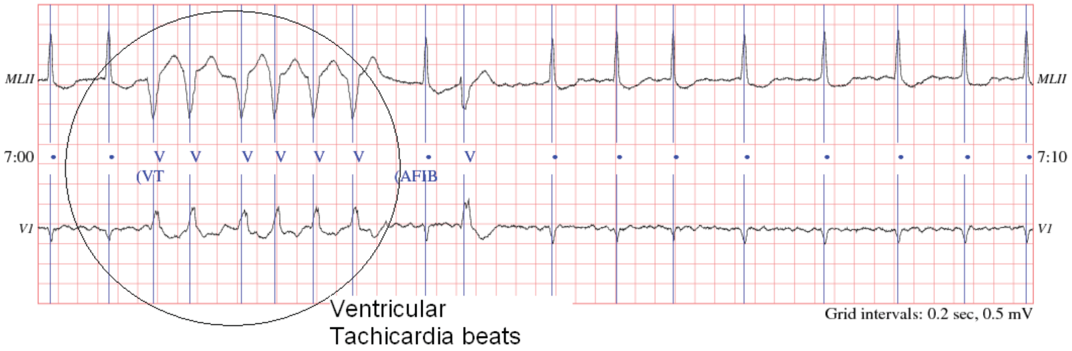

MLII
7:00
V
V
V
V
V
V
(VT
(AFIB
V
V1
MLII
7:10
V1
Ventricular Tachicardia beats
Grid intervals: 0.2 sec, 0.5 mV

Figure 2.3 Sample Ventricular Tachicardia from Patient #210 [14].

Since there can be Normal beats before and after the Ventricular Tachycardia beats, several windows were extracted from this type of beat sequences.

One window, is extracted to have Ventricular Tachycardia beats after the Normal beats which can be seen in Figure 2.4.

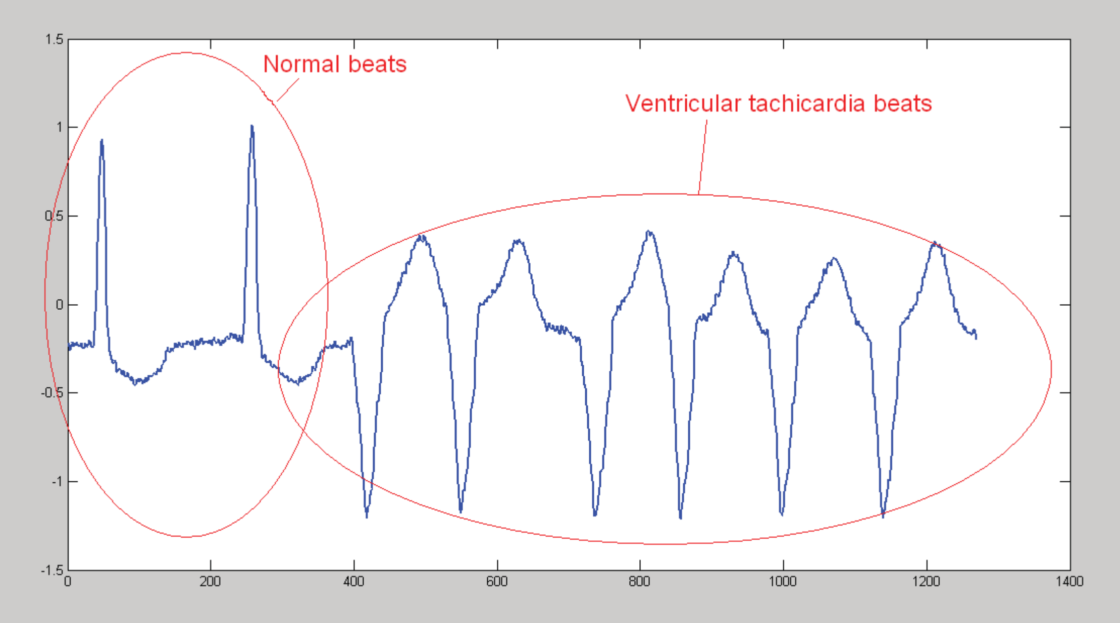


Figure 2.4 Ventricular Tachicardia Beats After Normal Ones

Another type of window like in the example in Figure 2.5 from recording 214, is extracted to have only Ventricular tachycardia beats:

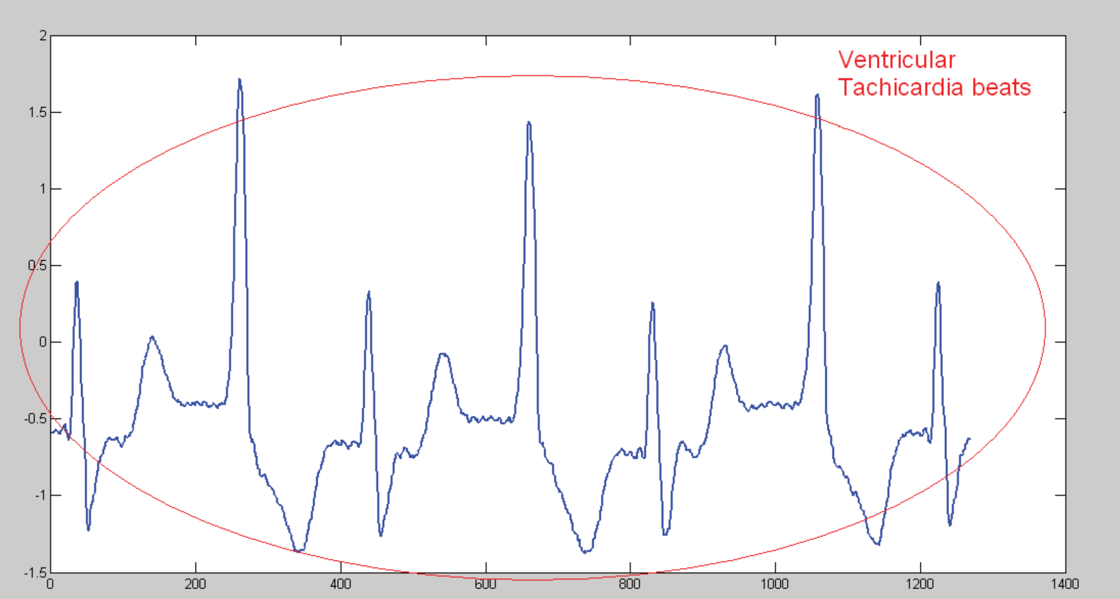


Figure 2.5 Ventricular Tachicardia Beats

Another type of window is set to contain Normal beats and Ventricular Tachycardia beats and then Normal beats again together like in Figure 2.6.

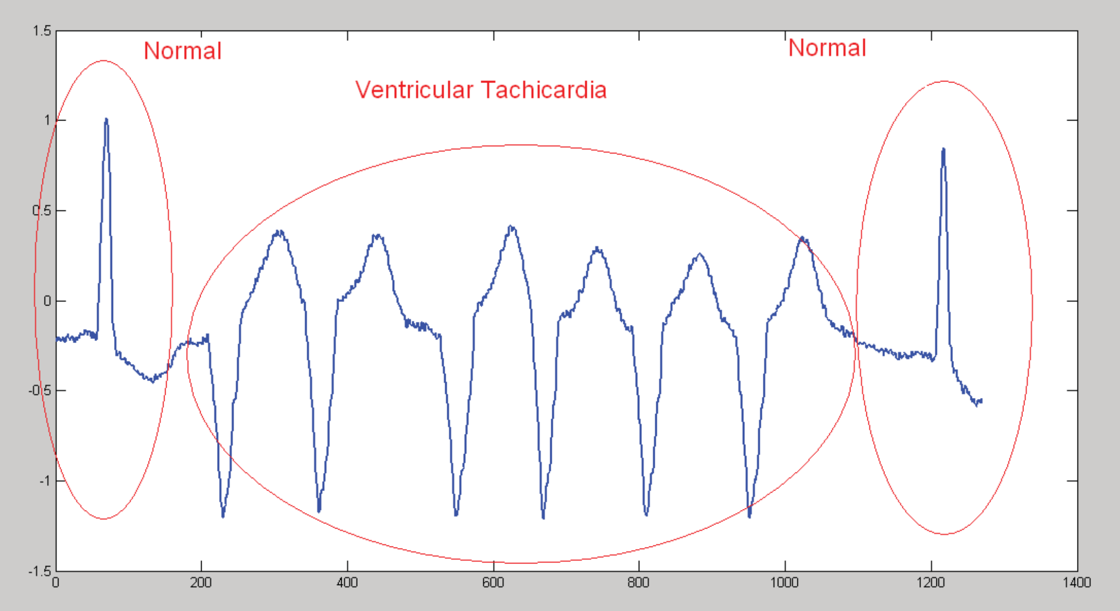


Figure 2.6 Ventricular Tachicardia Between Normal Beats

Or Normal-Ventricular Tachycardia-Normal-Ventricular Tachycardia like in Figure 2.7.

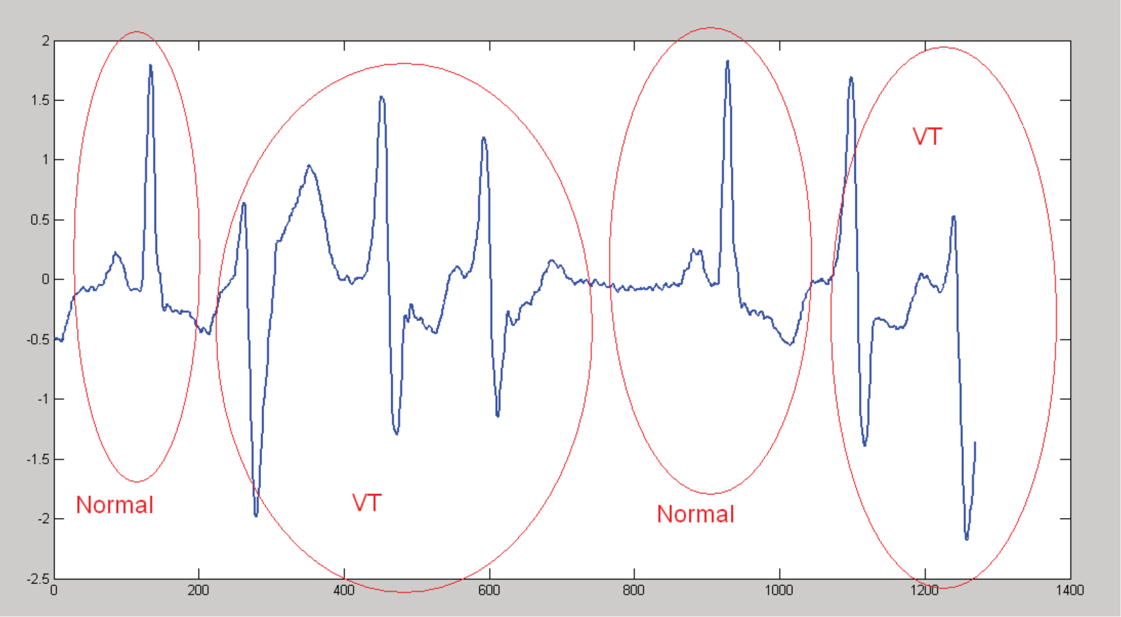


Figure 2.7 Subsequent Normal and Ventricular Tachicardia Beats

The reason for selecting this kind of mixed beat windows is in the context of the aim of this study.

It is aimed to classify the signals according to their frequency components that distinguishes them from each other. It is also aimed to perform the classification via a graphical user interface where the floating ECG signals were preprocessed before they were given to the classifier as inputs.

During this preprocessing step of the final classifier GUI, the same window size with the training data is used to preprocess the ECG wave from the patient. It wouldn't be unexpected to see several beat types appearing in the same window , therefore the classifier should be able to classify beat types regardless of the presence of different beat types in a window.

At that point it is necessary to configure the training inputs such that some sets have at least normal signals in their window.

Accordingly, it is aimed to train the network which will classify the ECG beats such that it will classify both Normal and Ventricular Tachicardia at the same time when they appear in the window.

The final extracted data for input set is separated according to the beat types:

1. Normal beat data;
2. Ventricular Tachycardia data;
3. Ventricular Tachycardia + Normal beat data;
4. Right Branch Block data;
5. Right Branch Block + Normal beat data;
6. Left Branch Block data;
7. Atrial Fibrillation data.

The reason that there is no combination for "Left Branch Block + Normal Beat" and "Atrial Fibrillation + Normal Beat" is that the recordings for LBB and AFIB did not have normal beats following or preceding these arrhythmic beats.

2.1.3 Feature extraction

The Fourier Transform would be a preferred method to represent the data in the frequency domain as the beats are placed in a window regardless of the time of their existence and morphology.

By doing so, if there is a beat of interest in this predefined window, the frequency components of the entire window would change and the classifier would warn the users as long as that beat stays in the window. In addition to that, since there is no time domain, the beat will be classified wherever it is placed in the window.

2.1.3.1 FFT. After gathering all the windows for different types of beats that contain both the arrhythmic beat solely or the beat with Normal signals, the windows were placed into matrices for easy handling of data.

The Fourier Transforms of these window matrices were taken and saved to other matrices of Fourier Transforms.

The Fourier Transform converts a signal in time domain to a signal in frequency domain. So by taking the Fourier Transform of the ECG windows extracted like explained above, the time domain information is no longer available and the signals are now distinguished according to their frequency components.

The Fourier transform of a Normal beat window is given in Figure 2.8.

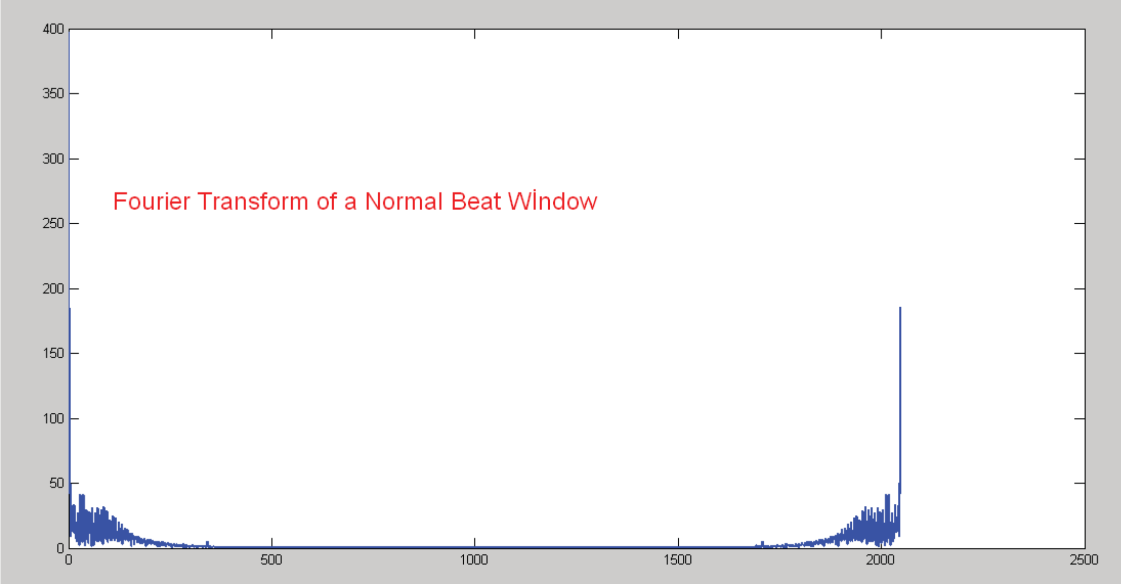


Figure 2.8 Fourier Transform of a Normal Beat

<u>2.1.3.2 FFTSHIFT.</u> The Fourier Transform of the windows extracted for different types of beats from the 30 minutes recordings were then subjected to the fftshift command. The fftshift command shifts the zero frequency components of the signal to the center of the spectrum. The shifted transform of the Normal beat window can be seen in the Figure 2.9.

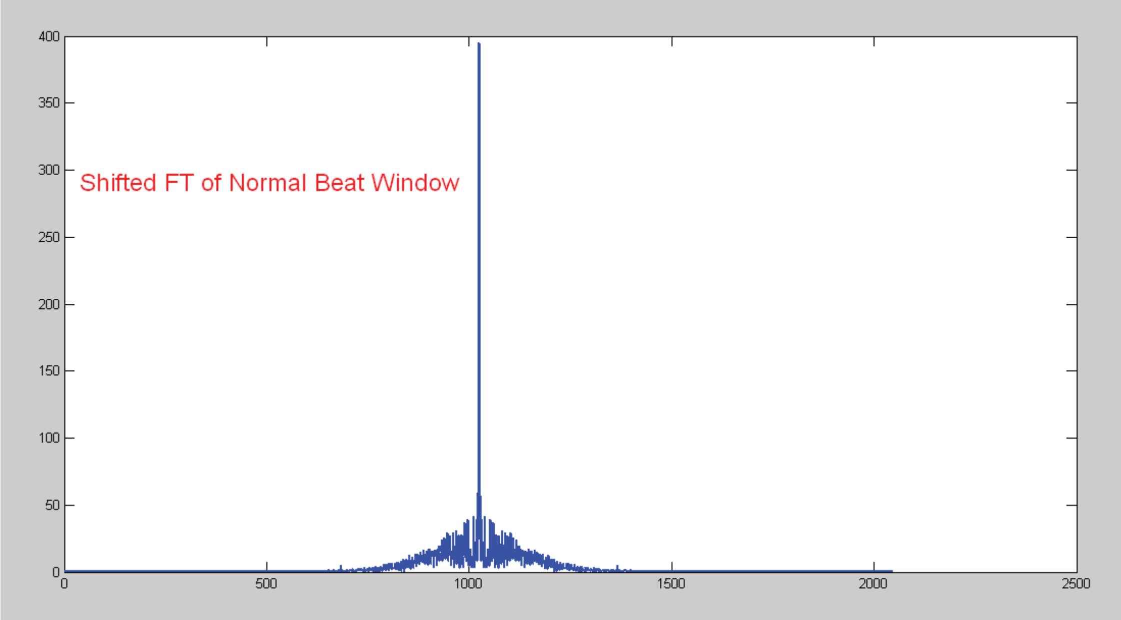


Figure 2.9 Shifted Fourier Transform of a Normal Beat

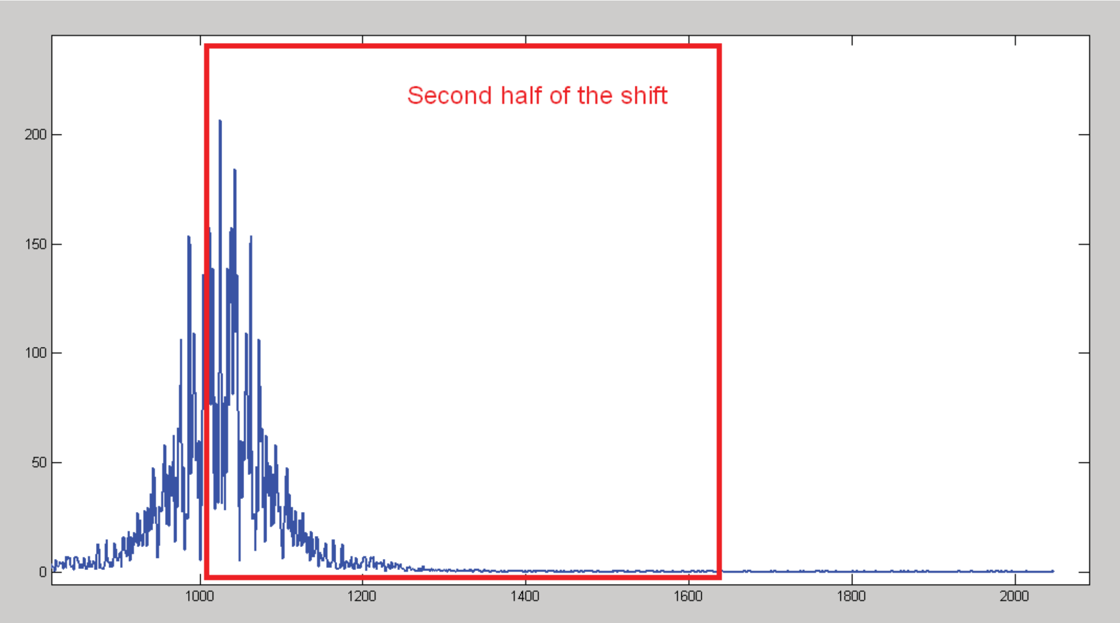


Figure 2.10 Second Half of the Shift

2.1.4 Resampling

The shifted windows will be the data that would be given to the neural network for training and testing inputs. However, there is too many data in the shifted signal, therefore, one last step is taken to minimize the number of input data that will be presented to the network. That is, since the shifted signal is a symmetric signal, taking the second half of the shift and getting average values at certain intervals.

Figures through 2.10 to 2.13 give a summary for the resampling process.

The second half of the shift to be extracted is determined to have 135 data points. This 135 point is selected so that by taking 135 data, one could cover entire information for this second half of the fftshift.

Since 135 is quite high number to use as training example, it is determined to take the average of the signal to represent this 135 data points. Therefore the 135 data points were divided into 27 intervals where there will be 5 data points in each interval.

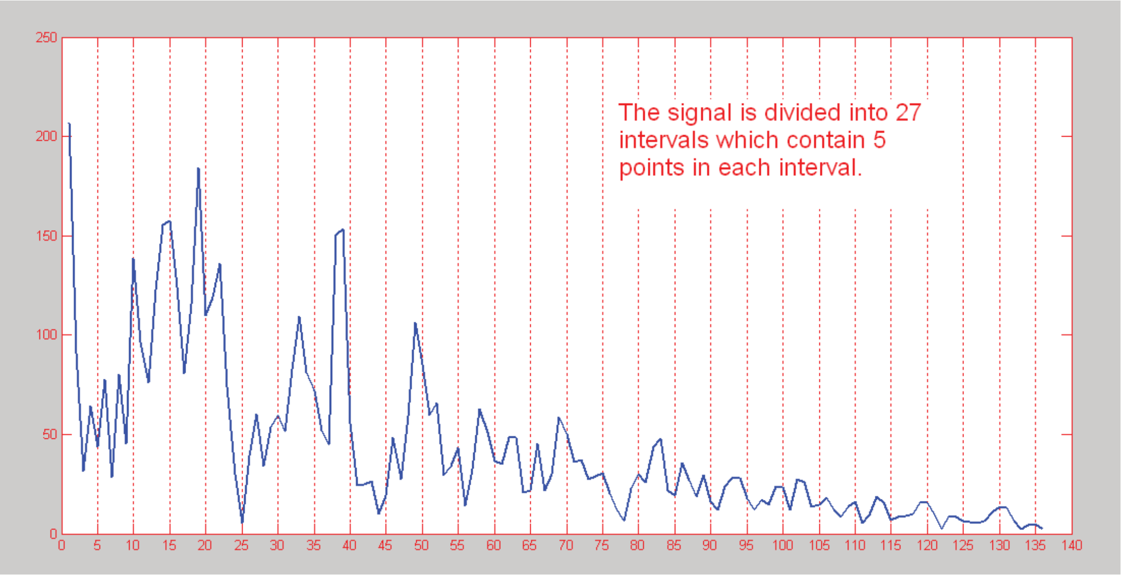


Figure 2.11 Signal Divided Into Intervals

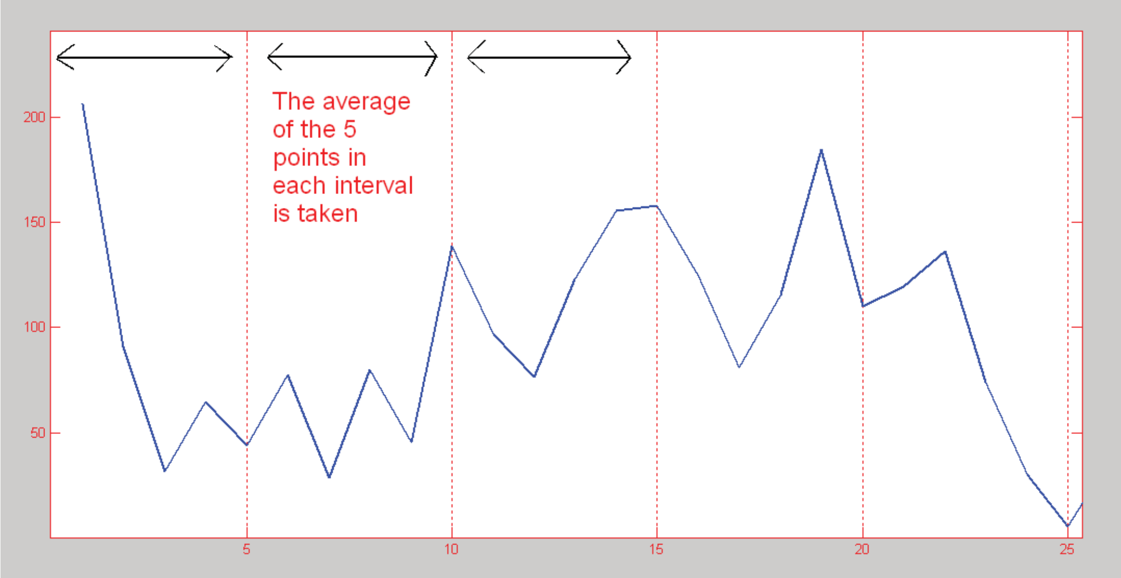


Figure 2.12 Average of Each Interval

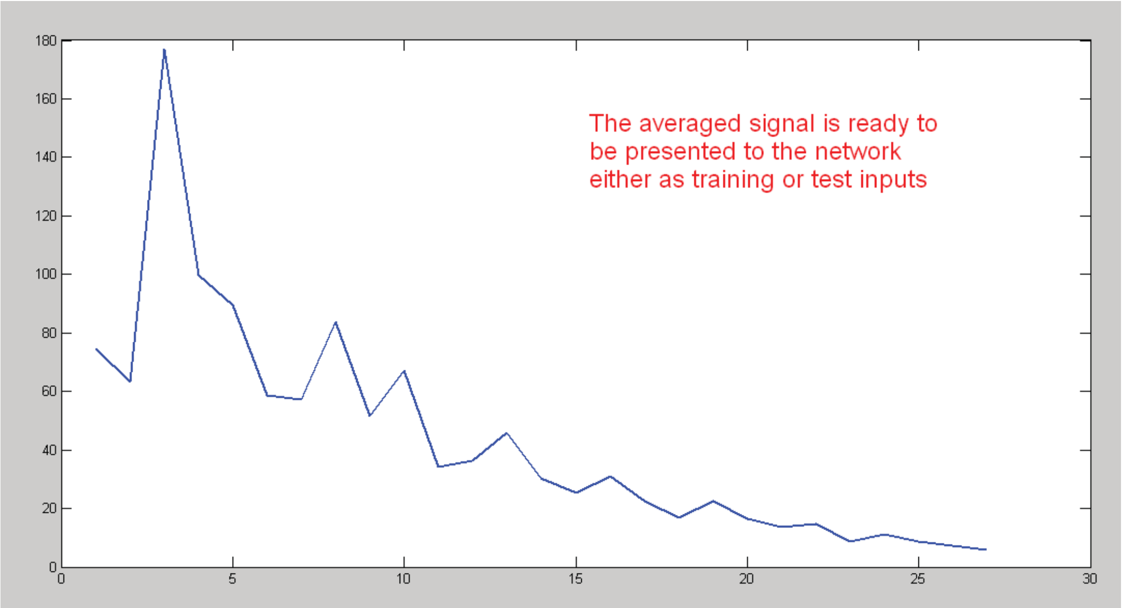


Figure 2.13 Input Ready to be Presented to the Network

The average of 5 data points in each interval is taken so that the input size is reduced to a reasonable value and there will be no data lost.

The final averaged data is the final input data that is used as one training example. One example input data consists of 27 data points.

2.1.4.1 Number of Examples. There are totally 233 beat examples and these 233 examples consist of maximum 50, minimum 12 examples per beats. The number of examples extracted for different beats is shown in Table 2.2.

Table 2.2
Number of Examples

N	VT	VTN	R	RN	L	AFIB
48	12	35	39	12	50	45

The average of examples per beat types can be seen in Figure 2.14 .

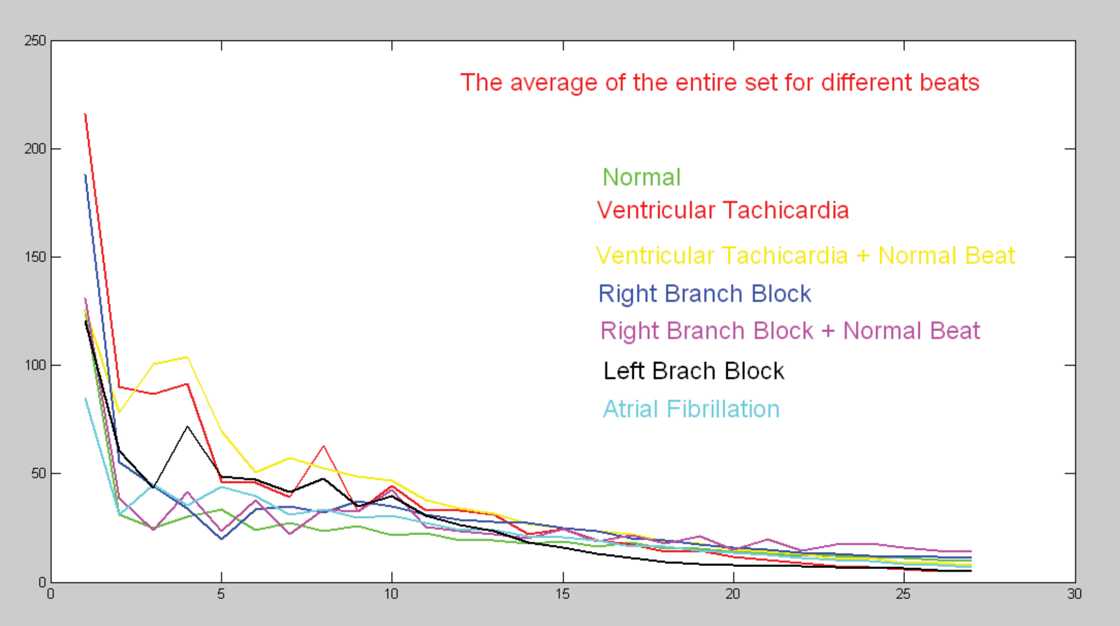


Figure 2.14 The average of the set for different beats

2.2 Data Noise Specification

The recordings taken from MIT BIH database have the frequency artifacts shown in Table 2.3.

Table 2.3
Frequencies and Sources

Frequency (Hz)	Source
0.042	Recorder pressure wheel
0.083	Playback unit capstan (for twice real-time playback)
0.090	Recorder capstan
0.167	Playback unit capstan (for real-time playback)
0.18-0.10	Takeup reel (frequency decreases over time)
0.20-0.36	Supply reel (frequency increases over time)

The details of the artifacts are given at [14].

"The most significant of these artifacts by far is the 0.167 Hz artifact on recordings that were played back at real time. The next largest is the 0.090 Hz artifact; the 0.083 Hz artifact on recordings that were played back at twice real-time is of roughly the same magnitude as the 0.090 Hz artifact. The 0.042 Hz artifact is of much lower magnitude. Other frequencies related to the drive train (at 0.42 Hz, 1.96 Hz, 9.1 Hz, and 42 Hz) do not appear as noticeable artifacts. The frequencies of the last two artifacts listed in the table depend on how much tape is on the supply and takeup reels; the supply reel causes a much more noticeable artifact than does the takeup reel. Other frequency-domain artifacts generated by the supply reel appear in the 0.10-0.18 Hz and 0.30-0.54 Hz bands. Four of the 48 records (102, 104, 107, and 217) include paced beats. The original analog recordings do not represent the pacemaker artifacts with sufficient fidelity to permit them to be recognized by pulse amplitude (or slew rate) and duration alone, the method commonly used for real-time processing. The database records reproduce the analog recordings with sufficient fidelity to permit use of pacemaker artifact detectors designed for tape analysis, however" [14].

2.3 Training and Test Set

As per the previous studies, many researchers used a test set of approx 30-50% of the entire set to test the system after training. Therefore, the separation in Table 2.4 was performed in the data extracted for this thesis.

Table 2.4
Training and Test Set Percentages

	Training	Test
N	30	18
VT	8	6
VTN	20	15
R	18	11
RN	8	4
L	30	20
AFIB	25	20
Total	139	94

As the Table 2.4 shows, 60% of the data extracted is selected to be the training set and remaining 40% of the data is selected to be the test set.

2.3.1 Training Input Design

Each averaged signal for each beat type is placed into one single matrix of inputs like in the example in Table 2.5.

Table 2.5
Matrix of Inputs for one beat type

$x_{1,1}$	$x_{1,2}$	$x_{1,3}$	...	...	$x_{1,26}$	$x_{1,27}$
$x_{2,1}$	$x_{2,2}$	$x_{2,3}$	...	...	$x_{2,26}$	$x_{2,27}$
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
$x_{m,1}$	$x_{m,2}$	$x_{m,3}$	...	...	$x_{m,26}$	$x_{m,27}$

Each row in Table 2.5 show one example of one beat type X, e.g. Normal.

Table 2.6
Matrix of Inputs for another beat type

$y_{1,1}$	$y_{1,2}$	$y_{1,3}$	...	...	$y_{1,26}$	$y_{1,27}$
$y_{2,1}$	$y_{2,2}$	$y_{2,3}$	...	...	$y_{2,26}$	$y_{2,27}$
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
$y_{m,1}$	$y_{m,2}$	$y_{m,3}$	·	·	$y_{m,26}$	$y_{m,27}$

Likewise, each row in Table 2.6 show one example of one beat type Y, e.g. Atrial Fibrillation.

In the end, all the input matrices are cascaded to have one final input matrix like in the example in Table 2.7. The final INPUT matrix has a dimension of 139x27.

Table 2.7
Final Input Matrix

$x_{1,1}$	$x_{1,2}$	$x_{1,3}$	...	...	$x_{1,26}$	$x_{1,27}$
$x_{2,1}$	$x_{2,2}$	$x_{2,3}$	...	...	$x_{2,26}$	$x_{2,27}$
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
$x_{m,1}$	$x_{m,2}$	$x_{m,3}$	...	...	$x_{m,26}$	$x_{m,27}$
$y_{1,1}$	$y_{1,2}$	$y_{1,3}$	...	...	$y_{1,26}$	$y_{1,27}$
$y_{2,1}$	$y_{2,2}$	$y_{2,3}$	...	...	$y_{2,26}$	$y_{2,27}$
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
$y_{m,1}$	$y_{m,2}$	$y_{m,3}$	...	...	$y_{m,26}$	$y_{m,27}$

2.3.2 Target Output Design

The target outputs is determined as in Table 2.8.

Table 2.8
Target Output Design

	N	N+VT	VT	RB+N	RB	LB	AFIB
N	1	0	0	0	0	0	0
VT	0	1	1	0	0	0	0
RB	0	0	0	1	1	0	0
LB	0	0	0	0	0	1	0
AFIB	0	0	0	0	0	0	1

Each column in the table shows the target output values for a given input. When, for example, a normal beat is presented to the ANN, it is expected to produce the following output.

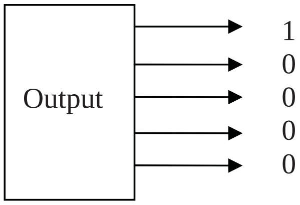


Figure 2.15 Expected Output of Normal Beat

The expected output for both the Ventricular Tachycardia (VT) and the Ventricular Tachycardia + Normal beat (VT+N) are the same as shown in Figure 2.16.

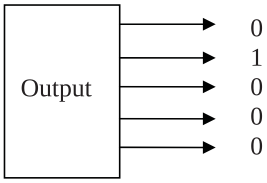


Figure 2.16 Expected Output of VT and VT+N Beats

The target set for VT and the VT+N beat is the same whereas the input set created for both sets are separated in order to make it easy to follow up during tests. In other words, if we assume that there are 5 leds at the output, and if the VT beat is presented, the led for VT will be lit at the output which is the second led from top and the row corresponding to VT in Table 2.8.

2.4 Neural Network Structure

In this study, the Feedforward Backpropagation network is used. The training attempts were performed with 1 hidden layered and 2 hidden layered architectures respectively.

During the training and test of each ANN architecture, the number of neurons in the layers and the learning rate parameters were changed. All the architectures were formed with feedforward backpropagation and the sigmoid function is used as the activation function.

In each attempt of the training for both 1 hidden layered networks and the 2 hidden layered networks,

1. the number of neurons were increased in the hidden layers;
2. 2 different learning rate values were assigned to the networks;
3. networks were trained with brand new initial weights.

The neural network toolbox training tool automatically stops training when one of the following criteria is met:

1. the desired network error is reached (0);
2. the desired minimum gradient reached $(1xexp(-10))$;
3. maximum fail for validation checks is reached (100).

If the desired network error reaches the desired value which is zero, this means the network calculates the exact expected outputs during training. If the network reaches the preset minimum gradient value, this means the network started to use very little amount of delta to update its weights. If the validation checks fail continuously, this means the network is not performing any better.

Figures 2.17 and 2.18 show the architectures in block diagram.

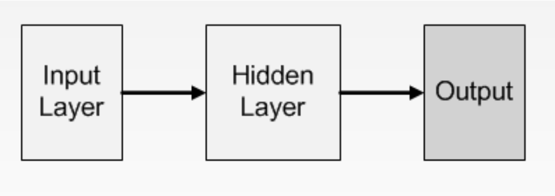


Figure 2.17 ANN Architecture with One Hidden Layer

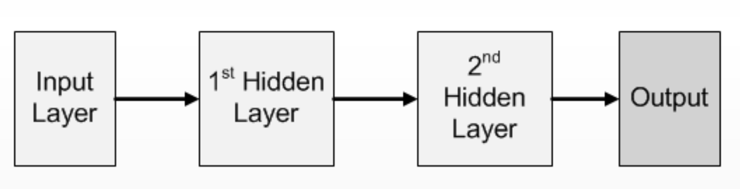


Figure 2.18 ANN Architecture with Two Hidden Layers

2.5 Experimental Design of the Network

Since one example data is represented with 27 points, the network is designed to have 27 inputs nodes in the input layer. The same logic applies to the output unit, since there are 5 classes, it is designed to have 5 output nodes in the output layer.

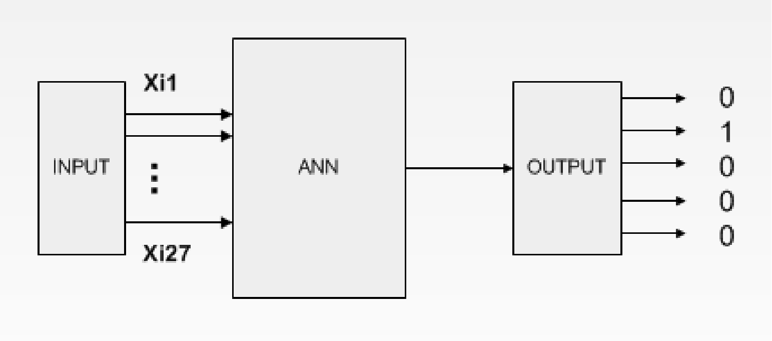


Figure 2.19 Experimental Design of the Network

The trained networks were tested both with the test set created at the beginning and back with the training set used during training and the results were recorded into a confusion matrix as in the example in Table 2.9.

The confusion matrix show that 66% of the test signals for Normal beats were classified correctly however 11% were classified as RBB, 11% was classified as AFIB and 11% was classified as LBB. The remaining 11% was not recognized at all.

Table 2.9
Confusion Matrix Example

	N	VT	R	AFIB	L	NON
N	66	0	11	11	11	11
VT	0	79	0	0	0	21
R	0	10	90	0	0	0
AFIB	14	0	0	72	0	14
L	0	0	0	0	90	10

The VT test inputs were classified 79% correctly. The remaining of the test signals were not classified at all. The network classified the RBB test inputs 90% correctly . The remaining 10% of the RBB was misclassified as VT. The percentage of correctly classified AFIB test inputs are 72% in the matrix, the 14% of the test signal was not classified at all. Last but not least,the LBB test inputs were 90% correctly classified whereas the network couldn't classify the remaining 10%.

2.6 Patient Monitoring GUI

In order to perform a user friendly interface for clinical use of the ANN trained to classify the ECG Arrhythmia types, a Graphical User Interface with basic controls was designed.

The final GUI is designed to have only following components :

1. Drop down menu for recording selection at the top left of the GUI window;
2. "Run" and "Pause" push buttons at the top right of the GUI window;
3. An axis which shows a window of 4 seconds out of the entire recording in the middle of the GUI window;
4. A static text bar which shows the classification result for the specified window

at the bottom middle of the GUI window.

However, one additional drop down menu was also included in order to perform observations on different types of networks during this study.

In Figure 2.20 a screenshot of the designed GUI is shown and the main components are circled:

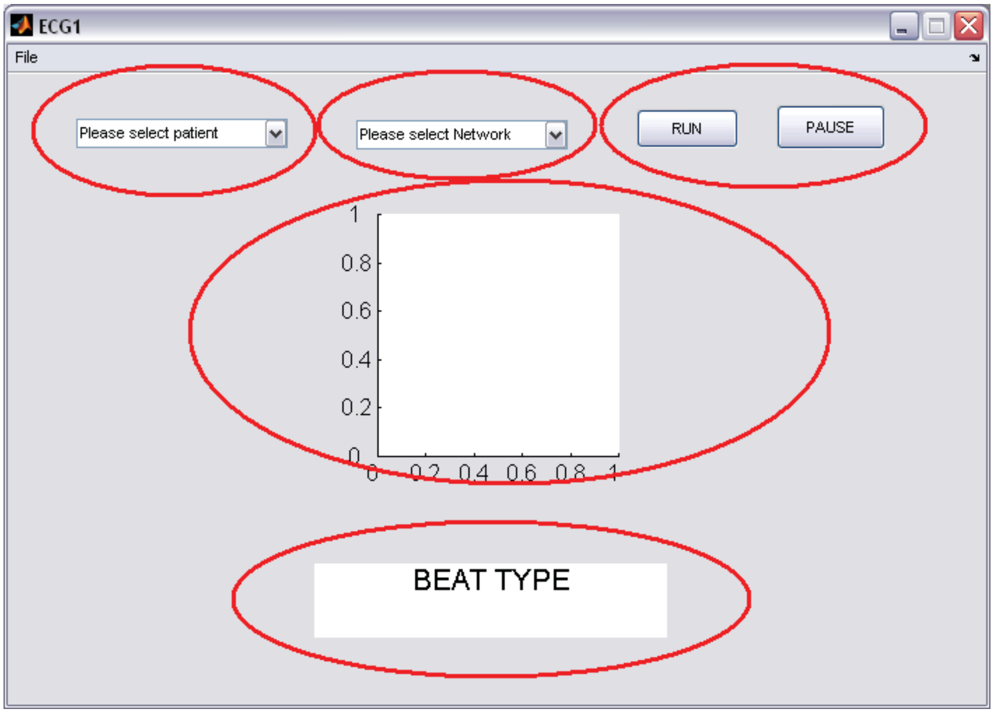


Figure 2.20 Patient Monitoring GUI

The user needs to select a patient and then a network. See Figures 2.21 and 2.22

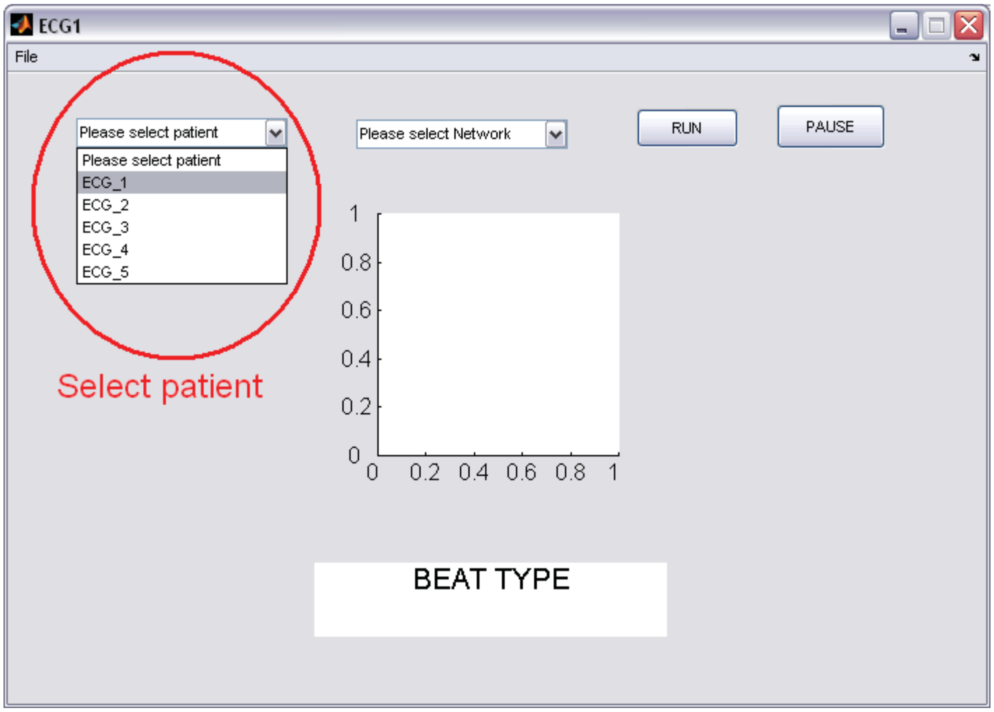


Figure 2.21 Select Patient

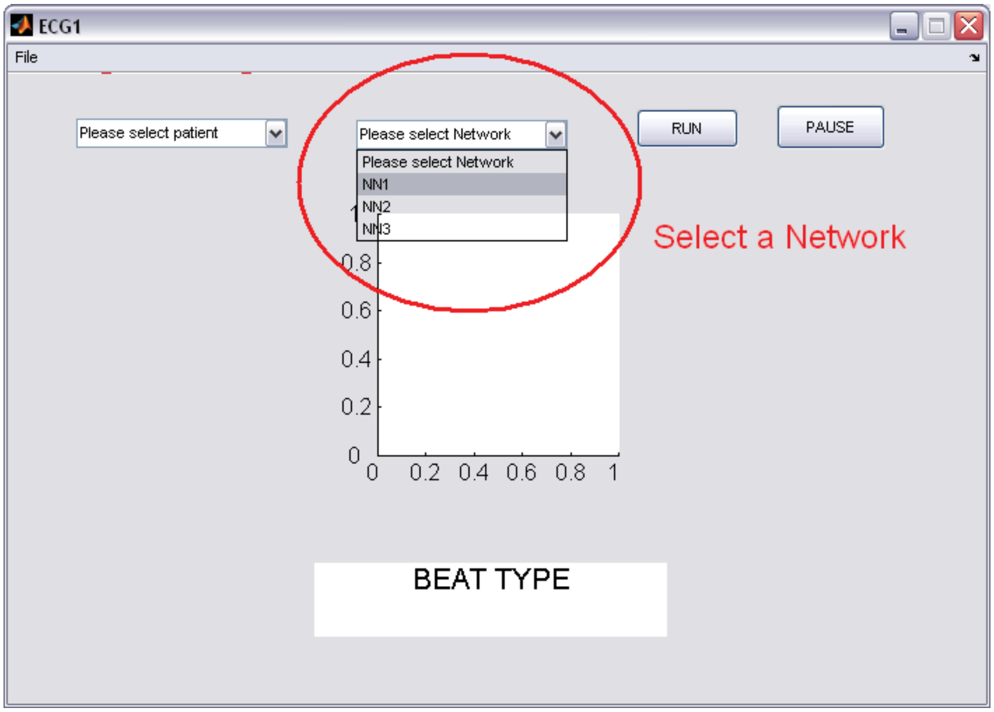
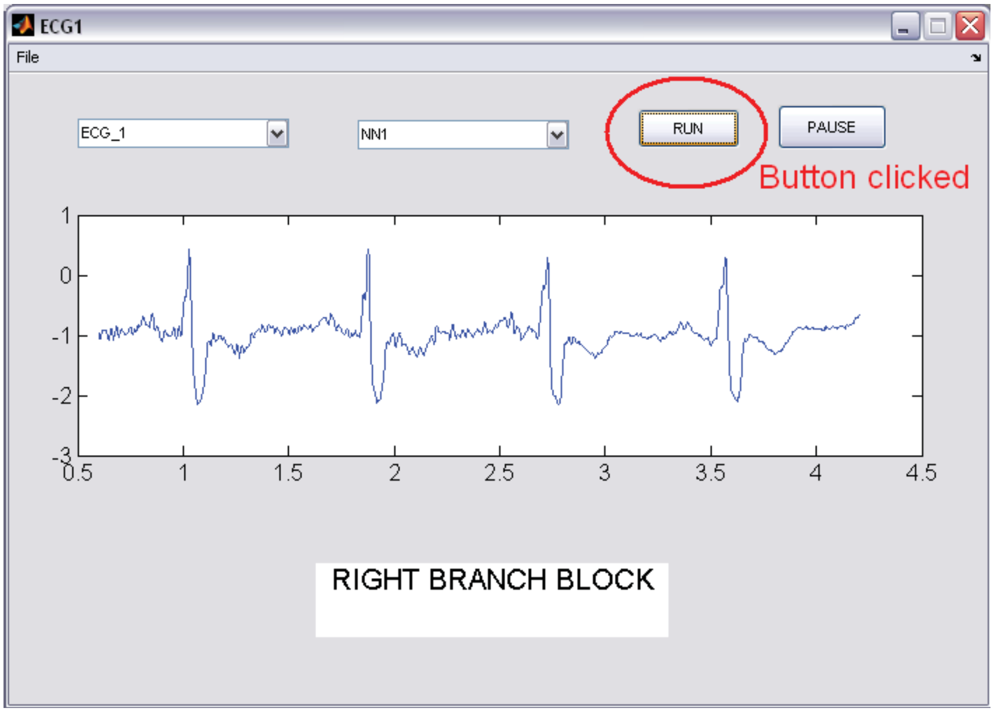


Figure 2.22 Select Network

When the user selects both patient and the network, the GUI is ready to start the simulation. When the user clicks on the RUN button shown in Figure 2.23, the ECG recording will start to flow on the screen.

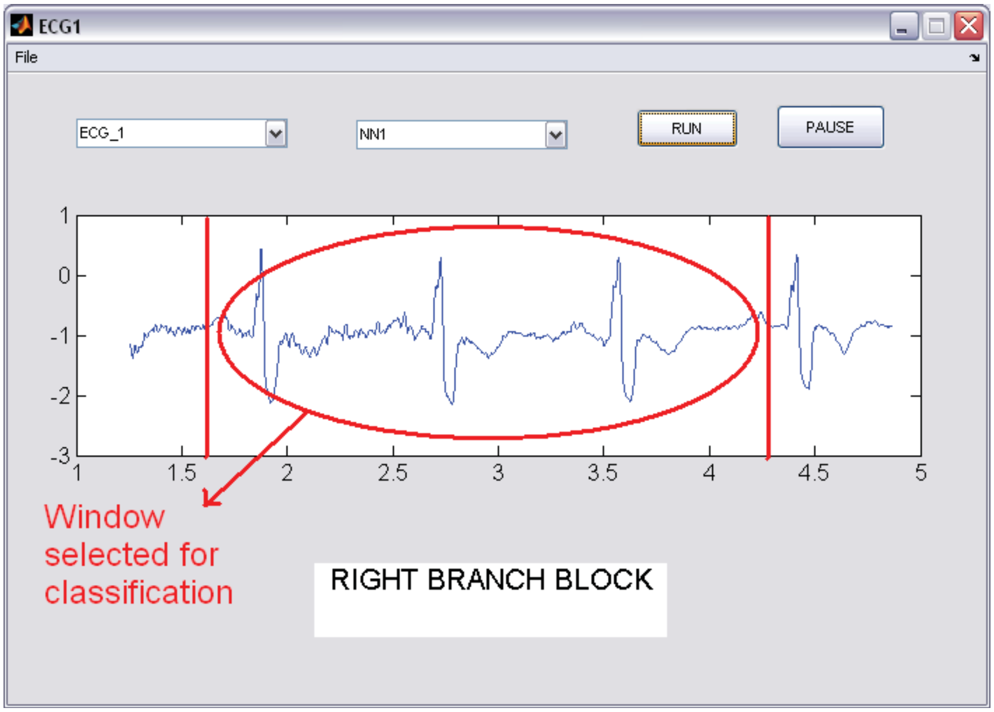


RIGHT BRANCH BLOCK

Figure 2.23 RUN button clicked on the GUI

In the background, here is what the GUI performs in order to classify the beats and display the results:

First, a window of 1270 data points were selected from the window displayed as shown in Figure 2.24. This 1270 data point is the same size with the data extracted from the ECG recording for training at the first place.



RIGHT BRANCH BLOCK

Figure 2.24 Classification Area

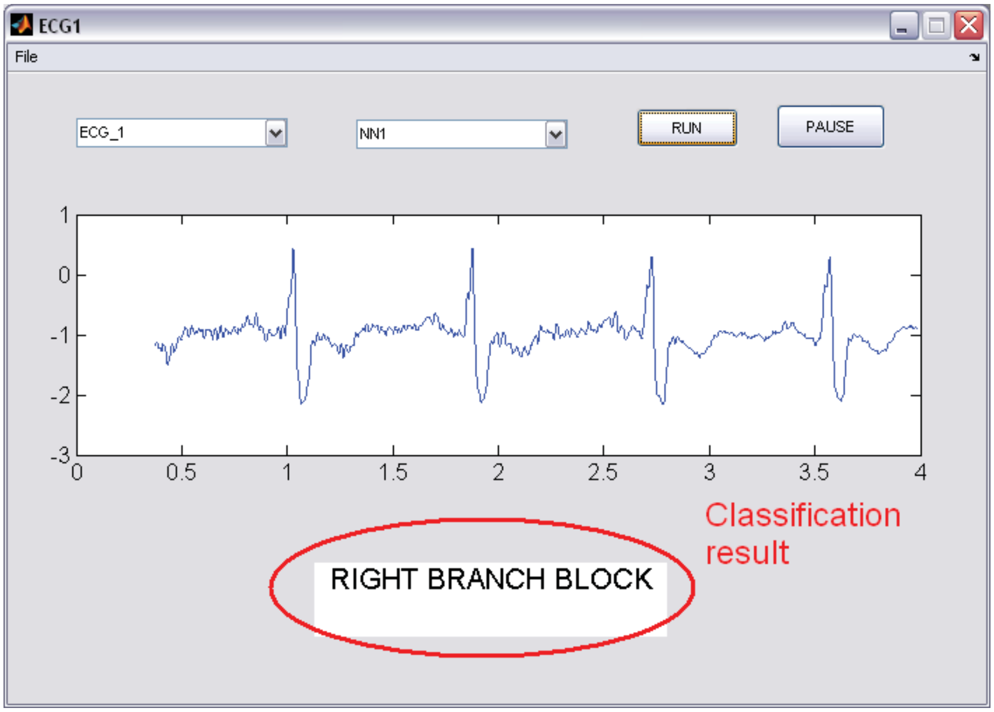
The rest of the processing of the window selected in Figure 2.24 is the same with the process during the data extraction for training.

Fourier Transform of the window extracted above is taken and then with the fftshift command, the zero frequency components of the signal were shifted to the center of the spectrum. The second half of the shifted signal is extracted and average of 5 data points at 27 intervals were taken as explained in the Feature extraction section.

After this preprocessing process, the final extracted data from the flowing ECG recording has the proper size in order to be given to the pretrained ANN as test inputs. The ANN is tested with the newly extracted data out of the window and the results are saved in a simple matrix of results.

The test results matrix created with the GUI code is monitored in order to make a decision on the classification. Two different methods were used at the output matrix. First, a threshold method is used where the absolute value of the error matrix of the output layer with respect to the desired output matrix is calculated. If this calculation was less than 0.5, therefore the classification was announced to be correct and displayed in the static text panel as shown in Figure 2.25. If the absolute value of the error was more than 0.5, then the classification was not displayed at all.

Secondly, the maximum of the results matrix were found and a Winner Take All algorithm is used in order to decide on the classification. The maximum output of the results of the test is the output that wins, and therefore the led that corresponds to the predefined outputs are said to be classified and the beat type is displayed on the static text panel window at the bottom accordingly.



ECG_1

NN1

RIGHT BRANCH BLOCK

Classification result

Figure 2.25 Classification Result

3. Results

Figures from 3.1 to 3.7 show the entire examples extracted for 5 different beat types. Remember that the total number of data extracted for 5 different beat types is 233 and the training set consists of 139 example and the test set consists of 94 example.

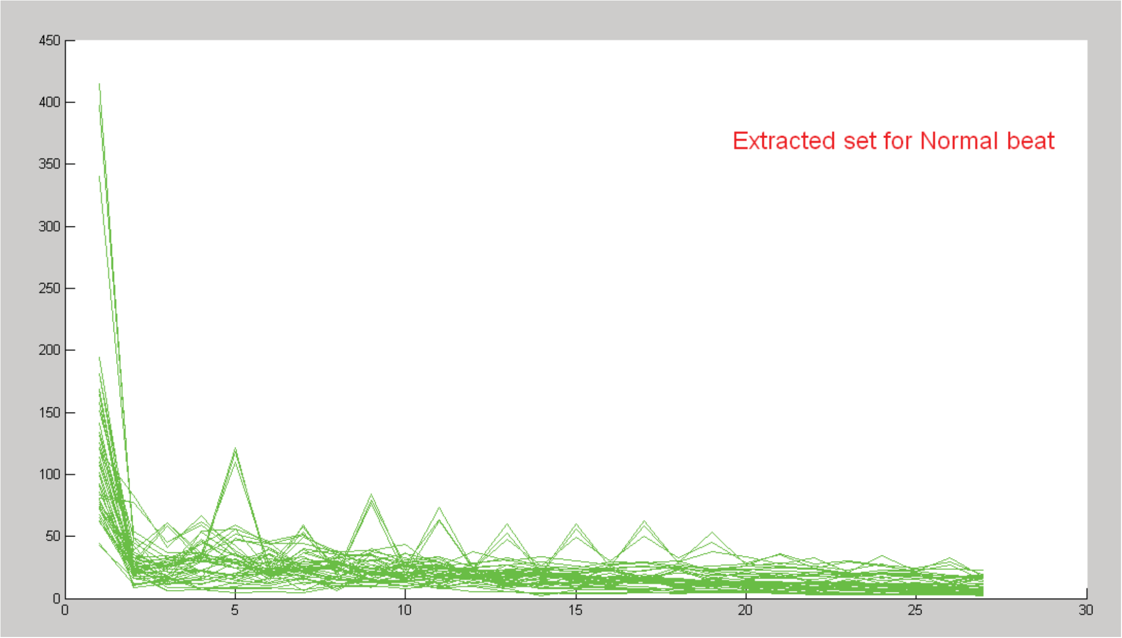


Figure 3.1 Extracted Set for Normal Beat

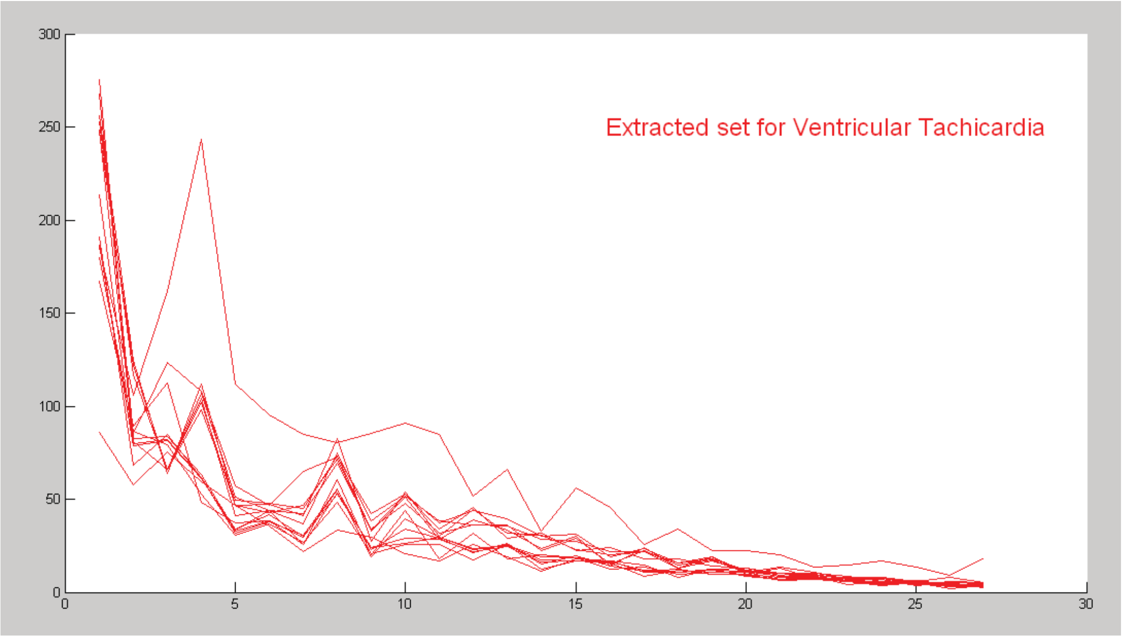


Figure 3.2 Extracted Set for Ventricular Tachicardia

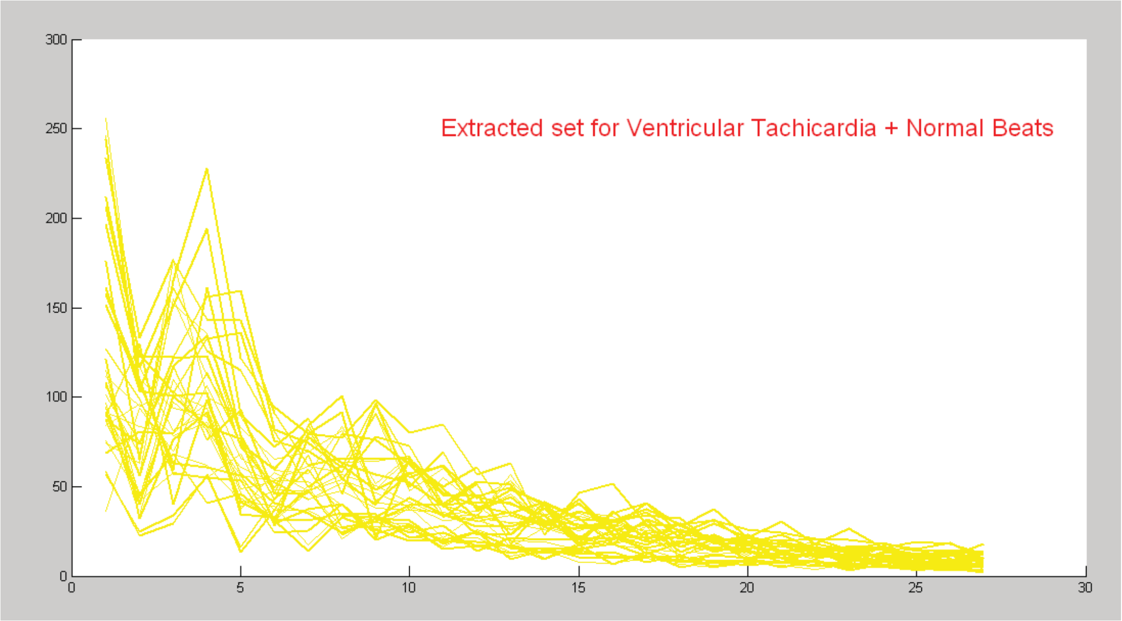


Figure 3.3 Extracted Set for Ventricular Tachicardia and Normal Beats

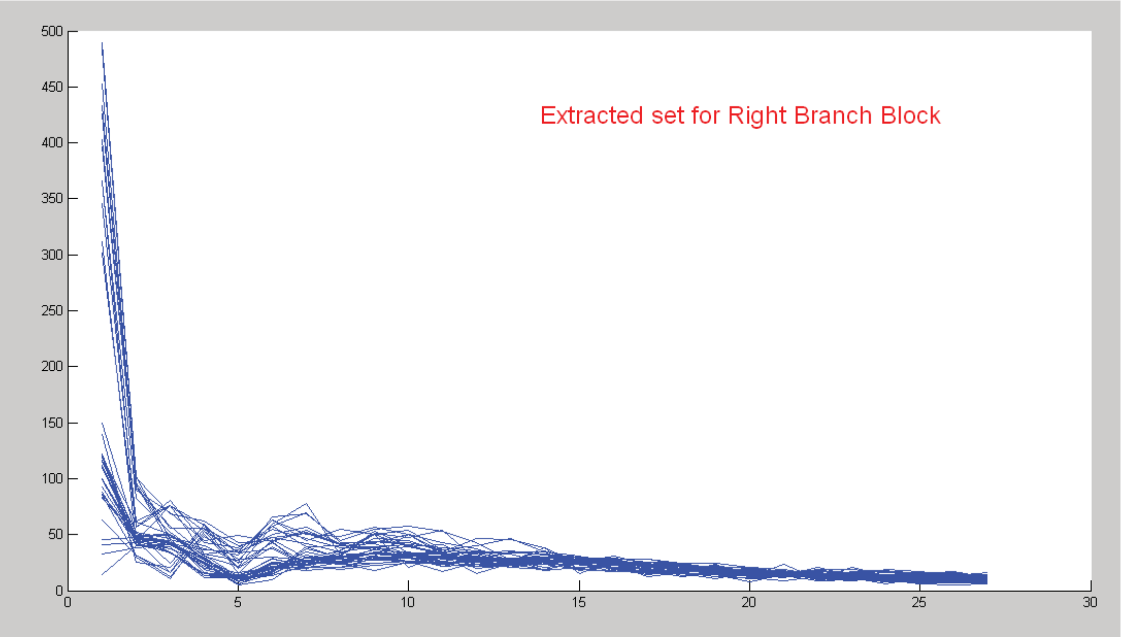


Figure 3.4 Extracted Set for Right Branch Block

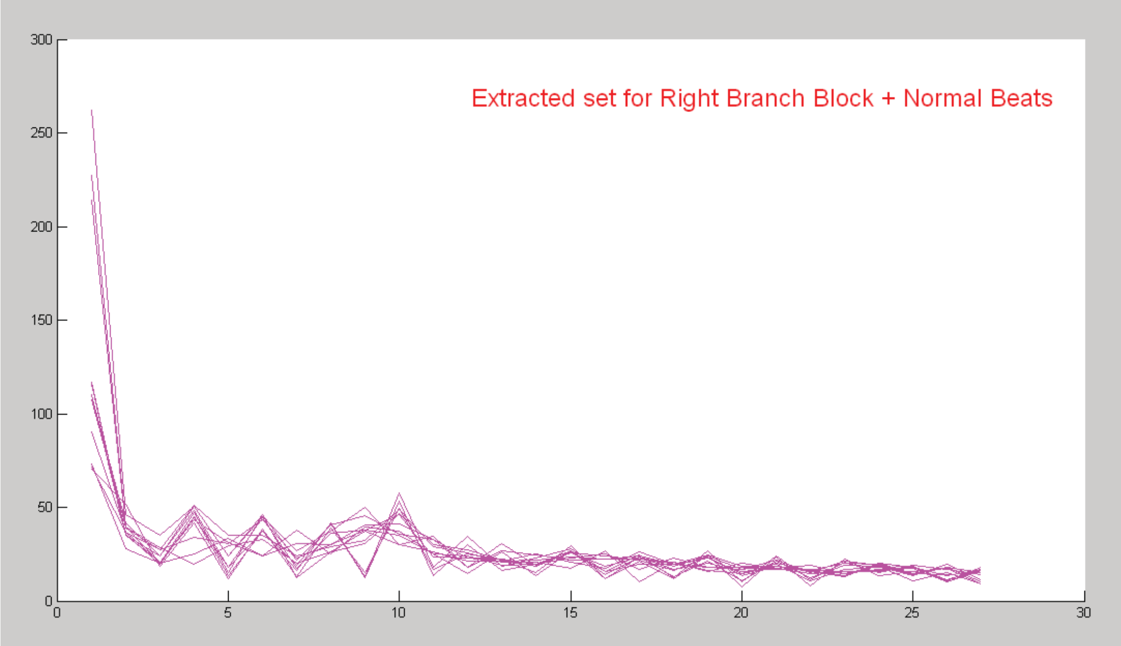


Figure 3.5 Extracted Set for Right Branch Block and Normal Beats

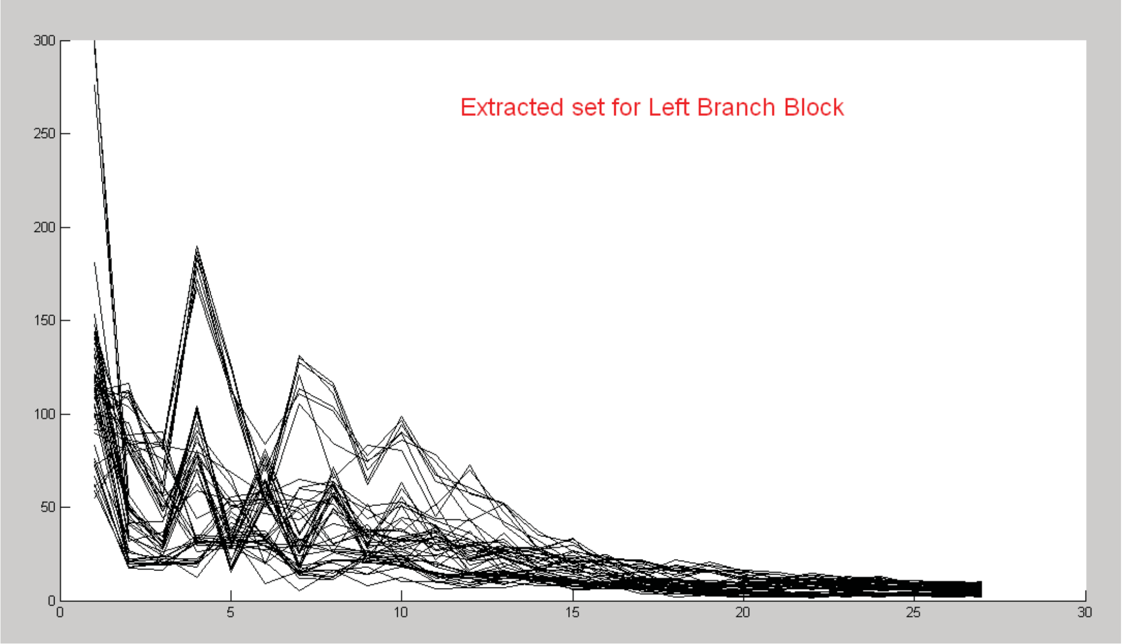


Figure 3.6 Extracted Set for Left Branch Block

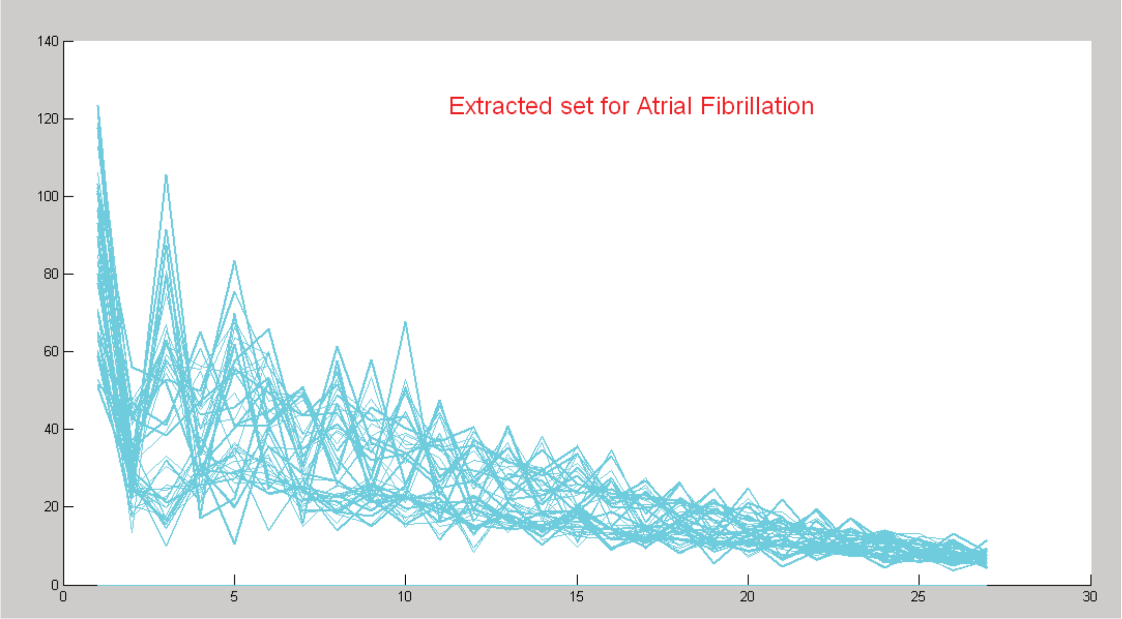


Figure 3.7 Extracted Set for Atrial Fibrillation

The entire data was separated into two sections for training and testing. 60% of the data was selected to be the traning set. Figure 3.8 shows all beat types in the training set together.

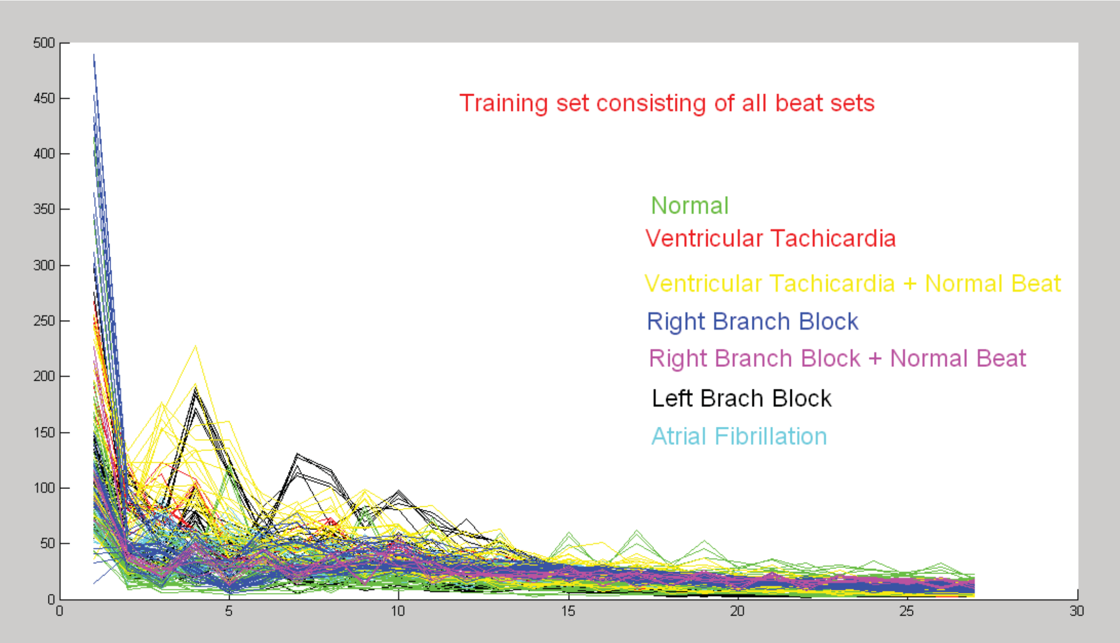


Figure 3.8 Training Set Consisting of All Beat Sets

The remaining 40% of the data, selected as the test set is shown in Figure 3.9.

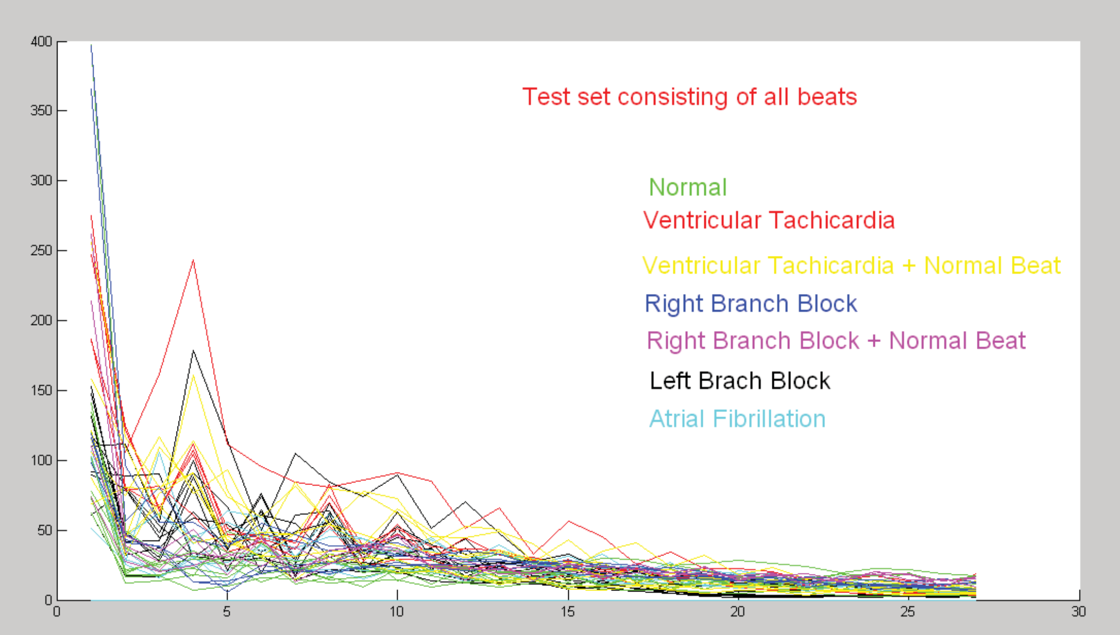


Figure 3.9 Test Set Consisting of All Beats

The data shown in the figures above are the input sets used in training the ANN. In the ANN architecture, the number of neurons were changed from 10 to 50 and during the training, the learning rate parameter was changed. Therefore, 10 different experiments have been performed in total. The Table 3.1 shows the experiments performed with 1 hidden layered networks.

Table 3.1
1 Hidden Layered Experiments

#neurons	Learning Rate	
10	0.01	0.001
20	0.01	0.001
30	0.01	0.001
40	0.01	0.001
50	0.01	0.001

The same experiments were performed with the 2 hidden layered networks as well. In Table 3.2, the experiment architectures are shown.

Table 3.2
2 Hidden Layered Experiments

#neurons	Learning Rate	
10-10	0.01	0.001
20-20	0.01	0.001
30-30	0.01	0.001
40-40	0.01	0.001
50-50	0.01	0.001

At the end of each experiment the confusion matrices and the performance graphics were recorded.

3.1 1 Hidden Layered Architectures

Tables through 3.3 to 3.22 show the results for experiments for networks with 1 hidden layer. Both training set and the test set are given to the network as inputs at the end of the training. Training inputs were used during training, whereas test inputs were never introduced to the network during training.

Tables on the top give the results with the test inputs given to the network The 0.5 threshold method is used to identify the correct classification. When the network produced its results for the inputs, the results were saved and compared with the desired outputs. The absolute value of the error at the output layer is calculated. The node which had an absolute value less than 0.5 was counted as correctly classified Second tables give results with the training inputs given back to the network to find out how good the network learned the training data. The 0.5 threshold method is used again to gather the results. Third tables give the results of test inputs without the threshold method. This time the maximum output is queried and the corresponding beat is the winner beat and declared to be correctly classified. Fourth table likewise gives the results of training inputs without the threshold method.

Table 3.3
Network with 10 Neurons Test Inputs

Learning Rate=0.001							Learning Rate=0.01					
	N	VT	R	AFIB	L	NON	N	VT	R	AFIB	L	NON
N	33	11	6	17	6	27	33	0	17	22	6	22
VT	5	77	0	0	5	13	0	67	0	0	10	23
R	12	0	76	4	0	8	0	0	76	4	0	20
AFIB	20	0	0	45	0	35	35	0	0	53	0	12
L	0	5	0	0	85	10	0	10	5	0	70	15

Table 3.4
Network with 10 Neurons Training Inputs

Learning Rate=0.001						Learning Rate=0.01				
	N	VT	R	AFIB	L	N	VT	R	AFIB	L
N	87	0	0	0	0	77	0	0	0	0
VT	0	54	0	0	0	0	64	0	0	0
R	0	0	50	0	0	0	0	43	0	0
AFIB	0	0	0	84	0	0	0	0	100	0
L	0	0	0	0	96	0	0	0	0	87

Table 3.5
Network with 10 Neurons Test Inputs - Threshold Removed

Learning Rate=0.001						Learning Rate=0.01				
	N	VT	R	AFIB	L	N	VT	R	AFIB	L
N	33	0	0	0	0	39	0	0	0	0
VT	0	54	0	0	0	0	58	0	0	0
R	0	0	32	0	0	0	0	36	0	0
AFIB	0	0	0	60	0	0	0	0	55	0
L	0	0	0	0	85	0	0	0	0	80

Table 3.6
Network with 10 Neurons Training Inputs - Threshold Removed

Learning Rate=0.001						Learning Rate=0.01				
	N	VT	R	AFIB	L	N	VT	R	AFIB	L
N	87	0	0	0	0	80	0	0	0	0
VT	0	59	0	0	0	0	69	0	0	0
R	0	0	50	0	0	0	0	50	0	0
AFIB	0	0	0	96	0	0	0	0	100	0
L	0	0	0	0	100	0	0	0	0	90

Table 3.7
Network with 20 Neurons Test Inputs

Learning Rate=0.001							Learning Rate=0.01					
	N	VT	R	AFIB	L	NON	N	VT	R	AFIB	L	NON
N	44	0	11	6	6	33	44	0	7	0	11	38
VT	5	77	0	0	0	18	4	83	0	3	10	
R	8	0	81	8	0	3	8	0	88	4	0	
AFIB	15	0	0	45	5	35	30	0	0	40	20	10
L	0	0	0	0	75	25	10	5	0	0	75	10

Table 3.8
Network with 20 Neurons Training Inputs

Learning Rate=0.001						Learning Rate=0.01				
	N	VT	R	AFIB	L	N	VT	R	AFIB	L
N	87	0	0	0	0	80	0	0	0	0
VT	0	65	0	0	0	0	50	0	0	0
R	0	0	50	0	0	0	0	45	0	0
AFIB	0	0	0	92	0	0	0	0	84	0
L	0	0	0	0	80	0	0	0	0	80

Table 3.9
Network with 20 Neurons Test Inputs - Threshold Removed

Learning Rate=0.001						Learning Rate=0.01				
	N	VT	R	AFIB	L	N	VT	R	AFIB	L
N	56	0	0	0	0	50	0	0	0	0
VT	0	68	0	0	0	0	58	0	0	0
R	0	0	31	0	0	0	0	33	0	0
AFIB	0	0	0	65	0	0	0	0	60	0
L	0	0	0	0	95	0	0	0	0	90

Table 3.10
Network with 20 Neurons Training Inputs - Threshold Removed

Learning Rate=0.001						Learning Rate=0.01				
	N	VT	R	AFIB	L	N	VT	R	AFIB	L
N	87	0	0	0	0	83	0	0	0	0
VT	0	72	0	0	0	0	59	0	0	0
R	0	0	50	0	0	0	0	50	0	0
AFIB	0	0	0	96	0	0	0	0	96	0
L	0	0	0	0	87	0	0	0	0	94

Table 3.11
Network with 30 Neurons Test Inputs

Learning Rate=0.001							Learning Rate=0.01					
	N	VT	R	AFIB	L	NON	N	VT	R	AFIB	L	NON
N	67	0	0	0	11	22	44	0	0	0	0	56
VT	0	83	5	4	4	4	0	90	0	4	0	6
R	0	0	100	0	0		12	0	88	0	0	
AFIB	5	0	0	55	0	40	30	0	0	50	10	10
L	0	15	0	0	65	20	15	0	0	10	65	10

Table 3.12
Network with 30 Neurons Training Inputs

	Learning Rate=0.001					Learning Rate=0.01				
	N	VT	R	AFIB	L	N	VT	R	AFIB	L
N	77	0	0	0	0	67	0	0	0	0
VT	0	68	0	0	0	0	72	0	0	0
R	0	0	51	0	0	0	0	37	0	0
AFIB	0	0	0	84	0	0	0	0	92	0
L	0	0	0	0	67	0	0	0	0	74

Table 3.13
Network with 30 Neurons Test Inputs - Threshold Removed

	Learning Rate=0.001					Learning Rate=0.01				
	N	VT	R	AFIB	L	N	VT	R	AFIB	L
N	78	0	0	0	0	56	0	0	0	0
VT	0	63	0	0	0	0	72	0	0	0
R	0	0	37	0	0	0	0	34	0	0
AFIB	0	0	0	80	0	0	0	0	75	0
L	0	0	0	0	75	0	0	0	0	75

Table 3.14
Network with 30 Neurons Training Inputs - Threshold Removed

	Learning Rate=0.001					Learning Rate=0.01				
	N	VT	R	AFIB	L	N	VT	R	AFIB	L
N	84	0	0	0	0	74	0	0	0	0
VT	0	72	0	0	0	0	76	0	0	0
R	0	0	51	0	0	0	0	43	0	0
AFIB	0	0	0	85	0	0	0	0	96	0
L	0	0	0	0	70	0	0	0	0	77

Table 3.15
Network with 40 Neurons Test Inputs

Learning Rate=0.001							Learning Rate=0.01					
	N	VT	R	AFIB	L	NON	N	VT	R	AFIB	L	NON
N	56	0	11	0	0	33	61	6	11	0	6	16
VT	0	64	0	0	9	28	0	62	0	0	10	28
R	0	0	96	0	0	4	12	0	72	0	0	16
AFIB	10	0	0	45	0	45	20	0	0	30	0	50
L	5	0	0	5	65	25	0	0	0	0	90	10

Table 3.16
Network with 40 Neurons Training Inputs

Learning Rate=0.001						Learning Rate=0.01				
	N	VT	R	AFIB	L	N	VT	R	AFIB	L
N	61	0	0	0	0	55	0	0	0	0
VT	0	73	0	0	0	0	73	0	0	0
R	0	0	83	0	0	0	0	97	0	0
AFIB	0	0	0	30	0	0	0	0	45	0
L	0	0	0	0	90	0	0	0	0	65

Table 3.17
Network with 40 Neurons Test Inputs - Threshold Removed

Learning Rate=0.001						Learning Rate=0.01				
	N	VT	R	AFIB	L	N	VT	R	AFIB	L
N	72	0	0	0	0	72	0	0	0	0
VT	0	53	0	0	0	0	54	0	0	0
R	0	0	34	0	0	0	0	34	0	0
AFIB	0	0	0	55	0	0	0	0	55	0
L	0	0	0	0	95	0	0	0	0	95

Table 3.18
Network with 40 Neurons Training Inputs - Threshold Removed

	Learning Rate=0.001					Learning Rate=0.01				
	N	VT	R	AFIB	L	N	VT	R	AFIB	L
N	94	0	0	0	0	97	0	0	0	0
VT	0	59	0	0	0	0	62	0	0	0
R	0	0	51	0	0	0	0	51	0	0
AFIB	0	0	0	73	0	0	0	0	85	0
L	0	0	0	0	100	0	0	0	0	94

Table 3.19
Network with 50 Neurons Test Inputs

	Learning Rate=0.001						Learning Rate=0.01					
	N	VT	R	AFIB	L	NON	N	VT	R	AFIB	L	NON
N	66	0	0	6	6	28	50	0	6	0	6	38
VT	0	67	0	14	10	24	0	79	0	0	7	14
R	0	0	98	0	0	2	0	0	84	0	0	16
AFIB	10	0	0	40	0	30	20	0	0	60	0	20
L	5	0	0	0	75	20	0	0	0	0	95	5

Table 3.20
Network with 50 Neurons Training Inputs

	Learning Rate=0.001					Learning Rate=0.01				
	N	VT	R	AFIB	L	N	VT	R	AFIB	L
N	96	0	0	0	0	80	0	0	0	0
VT	0	55	0	0	0	0	58	0	0	0
R	0	0	47	0	0	0	0	47	0	0
AFIB	0	0	0	76	0	0	0	0	88	0
L	0	0	0	0	90	0	0	0	0	90

Table 3.21
Network with 50 Neurons Test Inputs - Threshold Removed

Learning Rate=0.001						Learning Rate=0.01				
	N	VT	R	AFIB	L	N	VT	R	AFIB	L
N	67	0	0	0	0	72	0	0	0	0
VT	0	58	0	0	0	0	39	0	0	0
R	0	0	37	0	0	0	0	36	0	0
AFIB	0	0	0	75	0	0	0	0	80	0
L	0	0	0	0	100	0	0	0	0	95

Table 3.22
Network with 50 Neurons Training Inputs - Threshold Removed

Learning Rate=0.001						Learning Rate=0.01				
	N	VT	R	AFIB	L	N	VT	R	AFIB	L
N	97	0	0	0	0	87	0	0	0	0
VT	0	66	0	0	0	0	62	0	0	0
R	0	0	51	0	0	0	0	51	0	0
AFIB	0	0	0	92	0	0	0	0	96	0
L	0	0	0	0	100	0	0	0	0	90

3.2 2 Hidden Layered Architectures

Below are the results for experiments for networks with 2 hidden layers:

Table 3.23
Network with 10 - 10 Neurons Test Inputs

Learning Rate=0.001							Learning Rate=0.01					
	N	VT	R	AFIB	L	NON	N	VT	R	AFIB	L	NON
N	45	0	33	6	0	16	45	0	11	22	0	22
VT	0	76	0	0	0	28	4	70	0	8	14	4
R	4	0	72	4	8	12	12	0	84	0	0	4
AFIB	10	5	0	30	15	40	0	0	0	70	0	30
L	5	0	0	5	70	20	10	0	0	0	70	20

Table 3.24
Network with 10 - 10 Neurons Training Inputs

	Learning Rate=0.001					Learning Rate=0.01				
	N	VT	R	AFIB	L	N	VT	R	AFIB	L
N	80	0	0	0	0	70	0	0	0	0
VT	0	69	0	0	0	0	72	0	0	0
R	0	0	44	0	0	0	0	51	0	0
AFIB	0	0	0	73	0	0	0	0	92	0
L	0	0	0	0	77	0	0	0	0	80

Table 3.25
Network with 10 - 10 Neurons Test Inputs - Threshold Removed

	Learning Rate=0.001					Learning Rate=0.01				
	N	VT	R	AFIB	L	N	VT	R	AFIB	L
N	50	0	0	0	0	50	0	0	0	0
VT	0	58	0	0	0	0	44	0	0	0
R	0	0	27	0	0	0	0	34	0	0
AFIB	0	0	0	40	0	0	0	0	80	0
L	0	0	0	0	85	0	0	0	0	75

Table 3.26
Network with 10 - 10 Neurons Training Inputs - Threshold Removed

	Learning Rate=0.001					Learning Rate=0.01				
	N	VT	R	AFIB	L	N	VT	R	AFIB	L
N	81	0	0	0	0	81	0	0	0	0
VT	0	72	0	0	0	0	72	0	0	0
R	0	0	48	0	0	0	0	51	0	0
AFIB	0	0	0	85	0	0	0	0	100	0
L	0	0	0	0	87	0	0	0	0	97

Table 3.27
Network with 20 - 20 Neurons Test Inputs

Learning Rate=0.001							Learning Rate=0.01					
	N	VT	R	AFIB	L	NON	N	VT	R	AFIB	L	NON
N	72	0	6	11	0	11	56	0	11	0	0	33
VT	4	73	0	0	9	14	0	77	0	0	9	14
R	0	0	80	0	0	20	0	0	76	4	0	20
AFIB	10	0	5	75	0	10	10	0	0	65	0	25
L	0	0	0	0	100	0	5	10	0	10	65	10

Table 3.28
Network with 20 - 20 Neurons Training Inputs

Learning Rate=0.001						Learning Rate=0.01				
	N	VT	R	AFIB	L	N	VT	R	AFIB	L
N	90	0	0	0	0	83	0	0	0	0
VT	0	84	0	0	0	0	64	0	0	0
R	0	0	82	0	0	0	0	51	0	0
AFIB	0	0	0	92	0	0	0	0	92	0
L	0	0	0	0	100	0	0	0	0	77

Table 3.29
Network with 20 - 20 Neurons Test Inputs - Threshold Removed

Learning Rate=0.001						Learning Rate=0.01				
	N	VT	R	AFIB	L	N	VT	R	AFIB	L
N	74	0	0	0	0	67	0	0	0	0
VT	0	60	0	0	0	0	63	0	0	0
R	0	0	77	0	0	0	0	32	0	0
AFIB	0	0	0	85	0	0	0	0	75	0
L	0	0	0	0	100	0	0	0	0	75

Table 3.30

Network with 20 - 20 Neurons Training Inputs - Threshold Removed

Learning Rate=0.001						Learning Rate=0.01				
	N	VT	R	AFIB	L	N	VT	R	AFIB	L
N	96	0	0	0	0	84	0	0	0	0
VT	0	87	0	0	0	0	72	0	0	0
R	0	0	83	0	0	0	0	51	0	0
AFIB	0	0	0	100	0	0	0	0	100	0
L	0	0	0	0	100	0	0	0	0	87

Table 3.31

Network with 30 - 30 Neurons Test Inputs

Learning Rate=0.001							Learning Rate=0.01					
	N	VT	R	AFIB	L	NON	N	VT	R	AFIB	L	NON
N	46	0	11	16	0	27	46	0	11	5	5	33
VT	0	82	0	0	9	9	0	80	0	0	0	20
R	8	0	84	0	0	8	4	0	72	0	0	24
AFIB	20	0	0	45	0	35	15	10	0	55	0	20
L	5	0	0	5	50	40	0	10	0	0	75	15

Table 3.32

Network with 30 - 30 Neurons Training Inputs

Learning Rate=0.001						Learning Rate=0.01				
	N	VT	R	AFIB	L	N	VT	R	AFIB	L
N	75	0	0	0	0	83	0	0	0	0
VT	0	64	0	0	0	0	71	0	0	0
R	0	0	48	0	0	0	0	51	0	0
AFIB	0	0	0	73	0	0	0	0	88	0
L	0	0	0	0	83	0	0	0	0	67

Table 3.33
Network with 30 - 30 Neurons Test Inputs - Threshold Removed

Learning Rate=0.001						Learning Rate=0.01				
	N	VT	R	AFIB	L	N	VT	R	AFIB	L
N	50	0	0	0	0	56	0	0	0	0
VT	0	63	0	0	0	0	63	0	0	0
R	0	0	34	0	0	0	0	35	0	0
AFIB	0	0	0	70	0	0	0	0	60	0
L	0	0	0	0	85	0	0	0	0	70

Table 3.34
Network with 30 - 30 Neurons Training Inputs - Threshold Removed

Learning Rate=0.001						Learning Rate=0.01				
	N	VT	R	AFIB	L	N	VT	R	AFIB	L
N	81	0	0	0	0	84	0	0	0	0
VT	0	66	0	0	0	0	72	0	0	0
R	0	0	49	0	0	0	0	51	0	0
AFIB	0	0	0	85	0	0	0	0	96	0
L	0	0	0	0	87	0	0	0	0	87

Table 3.35
Network with 40 - 40 Neurons Test Inputs

Learning Rate=0.001							Learning Rate=0.01					
	N	VT	R	AFIB	L	NON	N	VT	R	AFIB	L	NON
N	46	0	16	11	11	16	46	0	11	16	5	22
VT	5	81	0	0	9	5	0	73	0	4	9	14
R	8	0	80	0	0	12	0	0	92	0	0	8
AFIB	20	0	5	50	0	25	10	0	0	55	10	25
L	0	5	5	0	70	20	0	10	0	0	70	20

Table 3.36
Network with 40 - 40 Neurons Training Inputs

Learning Rate=0.001						Learning Rate=0.01				
	N	VT	R	AFIB	L	N	VT	R	AFIB	L
N	93	0	0	0	0	87	0	0	0	0
VT	0	62	0	0	0	0	65	0	0	0
R	0	0	50	0	0	0	0	50	0	0
AFIB	0	0	0	92	0	0	0	0	84	0
L	0	0	0	0	93	0	0	0	0	93

Table 3.37
Network with 40 - 40 Neurons Test Inputs - Threshold Removed

Learning Rate=0.001						Learning Rate=0.01				
	N	VT	R	AFIB	L	N	VT	R	AFIB	L
N	61	0	0	0	0	44	0	0	0	0
VT	0	54	0	0	0	0	63	0	0	0
R	0	0	35	0	0	0	0	34	0	0
AFIB	0	0	0	70	0	0	0	0	55	0
L	0	0	0	0	80	0	0	0	0	85

Table 3.38
Network with 40 - 40 Neurons Training Inputs - Threshold Removed

Learning Rate=0.001						Learning Rate=0.01				
	N	VT	R	AFIB	L	N	VT	R	AFIB	L
N	94	0	0	0	0	90	0	0	0	0
VT	0	62	0	0	0	0	66	0	0	0
R	0	0	50	0	0	0	0	50	0	0
AFIB	0	0	0	100	0	0	0	0	92	0
L	0	0	0	0	97	0	0	0	0	94

Table 3.39
Network with 50 - 50 Neurons Test Inputs

Learning Rate=0.001							Learning Rate=0.01					
	N	VT	R	AFIB	L	NON	N	VT	R	AFIB	L	NON
N	46	0	5	16	11	12	46	0	5	5	5	39
VT	4	58	0	0	14	24	0	68	4	4	0	24
R	0	0	96	0	0	4	0	0	76	0	0	24
AFIB	0	0	0	75	0	25	5	0	0	55	0	40
L	0	0	0	0	95	5	0	0	0	0	60	40

Table 3.40
Network with 50 - 50 Neurons Training Inputs

Learning Rate=0.001						Learning Rate=0.01				
	N	VT	R	AFIB	L	N	VT	R	AFIB	L
N	61	0	0	0	0	83	0	0	0	0
VT	0	69	0	0	0	0	59	0	0	0
R	0	0	48	0	0	0	0	51	0	0
AFIB	0	0	0	84	0	0	0	0	88	0
L	0	0	0	0	61	0	0	0	0	100

Table 3.41
Network with 50 - 50 Neurons Test Inputs - Threshold Removed

Learning Rate=0.001						Learning Rate=0.01				
	N	VT	R	AFIB	L	N	VT	R	AFIB	L
N	61	0	0	0	0	61	0	0	0	0
VT	0	68	0	0	0	0	57	0	0	0
R	0	0	34	0	0	0	0	37	0	0
AFIB	0	0	0	70	0	0	0	0	75	0
L	0	0	0	0	70	0	0	0	0	100

Table 3.42
Network with 50 - 50 Neurons Training Inputs - Threshold Removed

	Learning Rate=0.001					Learning Rate=0.01				
	N	VT	R	AFIB	L	N	VT	R	AFIB	L
N	81	0	0	0	0	87	0	0	0	0
VT	0	69	0	0	0	0	61	0	0	0
R	0	0	50	0	0	0	0	51	0	0
AFIB	0	0	0	92	0	0	0	0	92	0
L	0	0	0	0	67	0	0	0	0	100

4. Discussions

The results are evaluated for the following points:

1. Overall success for the test results with both training input and test input;
2. Most successful network;
3. The success rate per beats in the networks;
4. Failed classifications.

4.1 Overall Success for the Test Results

When we compare the results from the tests, we can summarize the successful classification results with Table 4.1.

Table 4.1
Percentages of overall test results with 0.5 threshold method

#Neurons	10		20		30		40		50		Averaged Total
Learning Rate	0.01	0.001	0.01	0.001	0.01	0.001	0.01	0.001	0.01	0.001	
1HL	59	63	66	64	67	74	76	63	69	73	67
2HL	68	59	68	80	61	66	65	67	74	61	67

The Table 4.1 shows the percentage of overall success rate for testing in both 1 hidden layered and 2 hidden layered networks. The overall success rate is calculated by taking the percentage of the total successfully classified beats over total test inputs.

The example in Table 4.2 for 1 hidden layered network with 10 neurons and learning rate as 0.01 shows this calculation.

Table 4.2
Diagonals of the confusion matrix

	N	VT	R	AFIB	L	NON
N	**33**	11	6	17	6	27
VT	5	**77**	0	0	5	13
R	12	0	**76**	4	0	8
AFIB	20	0	0	**45**	0	35
L	0	5	0	0	**85**	10

All the numbers in the diagonals in bold which show the successful classifications are added. In this example this sum is 316. This number is divided into the number of classes which is 5 and therefore the final overall success rate is obtained as 63%.

According to Table 4.1, it can be concluded that the most successful network is the 2 hidden layered network with 20-20 architecture, lr= 0.001 learning rate with 80% success. The least successful network is the 1 hidden layered network with 10 neurons, lr=0.01 with 59% success rate.

In overall, the average success rate of the studies performed with 1 hidden layered networks were equal to the average success rate of the 2 hidden layered networks, which is 67%.

Accordingly, by using the same method, the overall success rates were also calculated for the following three conditions

1. Training data is given back to the network with the presence of 0.5 threshold comparison check. The results for this test is given in Table 4.3

2. Test data is given to the network with winner takes all comparison method at the output layer. The results for this test is given in Table 4.4.

Table 4.3
Percentages of overall test results with training inputs with 0.5 threshold method

	10		20		30		40		50		
	0.01	0.001	0.01	0.001	0.01	0.001	0.01	0.001	0.01	0.001	Total
1HL	74.2	74.2	64.8	74.8	68.4	69.4	67	67.4	72.6	72.8	70.56
2HL	73	68.6	73.4	89.6	72	68.6	75.8	78	76.2	64.6	67

Table 4.4
Percentages of overall test results with Winner Takes All method

	10		20		30		40		50		
	0.01	0.001	0.01	0.001	0.01	0.001	0.01	0.001	0.01	0.001	Total
1HL	53.6	52.8	58.2	63	62.4	72.4	62	61.8	64.4	67.4	61.8
2HL	56.6	52	62.4	82	56.8	49	56.2	60	66	54.6	59

3. Training data is given back to the network with winner takes all comparison method at the output layer. The results for this test is given in Table 4.5.

Table 4.5
Percentages of overall test results with training inputs with Winner Takes All method

	10		20		30		40		50		
	0.01	0.001	0.01	0.001	0.01	0.001	0.01	0.001	0.01	0.001	Total
1HL	77.8	78.4	76.4	78.4	73.2	72.4	77.8	75.4	77.2	81.2	76.83
2HL	80.2	74.6	78.8	93.2	62.4	73.6	78.4	80.6	78.2	71.8	77.18

4.2 Most Successful Network

By looking at the confusion matrices, one can see that only 2 networks managed to classify at east one beat type with 100% success rate. One of these networks is the 1 hidden layered network with 30 neurons, lr=0.001, which classified the Atrial Firbrillation 100% correctly. The other one is the 2 hidden layered network with 20-20 architecture, lr=0.001, which classified the Left Branch Block 100% correct.

4.3 The success rate per beats in the networks

If one focuses on the beat types, it can be noticed that the Normal beats had the least success rate in all training attempts.

Table 4.6 is a summary of all the tests performed with the networks. At the right hand side of the table the average success rates of the 1 hidden layered and 2 hidden layered networks per beat types is shown separately.

Table 4.6
Test Summary

	#Neurons	10		20		30		40		50			
	Learning Rate	0.01	0.001	0.01	0.001	0.01	0.001	0.01	0.001	0.01	0.001	AV-1H	AV-2H
N	1HL	33	33	44	44	44	67	56	61	66	50	49.8	
	2HL	45	45	56	72	46	46	46	46	46	46		49.4
VT	1HL	67	77	83	77	90	83	64	62	67	79	74.9	
	2HL	70	76	77	73	82	80	81	73	58	68		73.8
RBB	1HL	76	76	88	81	88	100	96	72	98	84	85.9	
	2HL	84	72	76	80	84	72	80	92	96	76		81.2
AFIB	1HL	53	45	40	45	50	55	45	30	40	60	46.3	
	2HL	70	30	65	75	45	55	50	55	75	55		57.5
LBB	1HL	70	85	75	75	65	65	65	90	75	95	76	
	2HL	70	70	65	100	50	75	70	70	95	60		72.5

It can be seen that the Normal beat is the least successfully classified beat whereas the RBB is the most successfully classified one.

At this point the dataset created for the training should be reconsidered. According to Table 3.3, the Normal beats contain the highest number of patients which is 16 whereas it has 30 examples for training. 18 examples were selected for testing the network out of the entire Normal beat examples. Therefore, the examples form some patients might not even be introduced to the network during training and that would be one of the explanations for why the test results for Normal beats are worse than other beat types.

Looking at the RBB dataset, the number of patients are the least among all beat types which is only 3. Therefore all 26 examples for training were extracted from these 3 patients as well as the 15 test examples.

The success in RBB classification can be related to the fact that the network has seen the examples from the same patients before.

It is a known fact that the higher number of training examples results in better success for classification. Therefore one may conclude that the classification of the beats would be even better if there was more data extracted for as many different window combinations as possible.

4.4 Failed classifications

The successful classification percentages were shown in Table 4.6. The remaining unsuccessful classifications are summarized in Table 4.7.

Table 4.7
Failed Classifications

	10				20				30				40				50			
	0.01		0.001		0.01		0.001		0.01		0.001		0.01		0.001		0.01		0.001	
	NON	OTR	NON	OTR	NON	OTR	NON	OTR	NON	OTR	NON	OTR	NON	OTR	NON	OTR	NON	OTR	NON	OTR
1HL	18	21.8	19	18.2	12	22.4	23	12.8	16	16.2	17	26	27	2.8	24	13	21	10	19	7.8
2HL	16	14.2	23	13.6	20	11.6	15	5	24	14.6	23	12	16	19	18	25	16	10	33	5.6

The fields "NON" show the percentages of those beats, which were not classified into any classes, but remained unidentified. By looking at the table, the 2 hidden layered network with 50-50 architecture, lr=0.001 had the most unidentification percentage. However the most successful network which is the 2 hidden layered network with 20-20 architecture, lr=0.001, had the least number of unidentified beats.

The fields "OTR" shows the total percentage of the misclassification in the network. Table clearly shows that the most successful network had very low amount of misclassification as expected. However there was a better misclassification result in the 1 hidden layered network with 40 neurons, lr=0.01. The most misclassification was performed in the network 2 hidden layered network with 30-30 architecture, lr=0.001.

Tables through 4.8 to 4.12 go into further details about the misclassified beats. The first table, Table 4.8 examines the Normal Beat inputs that are misclassified. If we examine the first line of the table we see the 10 neuron lr=0.01 architecture put 17 % of the N beat test inputs into the RBB class, 22% into the AFIB and 7 % into the LBB class. On the other hand, again the 10 neuron but this time lr=0.001 architecture put 11% of the Normal beat test inputs into the VT class, 6% into the RBB, 17% into AFIB and 6% into the LBB class. In the overall, when we look at the averaged results at the end of the table, the table tells that the N beat was put into the RBB class most of the time with 9% average. The next misclassification was putting the N beat into the AFIB class with 7.95%. The misclassification with AFIB is mostly observed in the 2 hidden layered architectures.

The second table, Table 4.9 shows the percentages of the VT beats that are misclassified. According to the table, it is clear that the networks mostly put the VT into the LBB class with 6%. The RBB beats were put into the N class in most cases with 4% average as shown in Table 4.10. As per Table 4.11 the rate of misclassification is very high in AFIB. The 99% of the network architectures put some of the AFIB examples into the N class and for the LBB, shown in Table 4.12, it is observed that mostly the networks put the LBB test examples into the N or VT class.

Table 4.8
Misclassifications of the Normal beats

	VT	R	AFIB	L
10-0.01	0	17	22	6
10-0.001	11	6	17	6
20-0.01	0	7	0	11
20-0.001	0	11	6	6
30-0.01	0	0	0	0
30-0.001	0	0	0	11
40-0.01	0	11	0	0
40-0.001	6	11	0	6
50-0.01	0	0	6	6
50-0.001	0	6	0	6
10-10-0.01	0	11	22	0
10-10-0.001	0	33	6	0
20-20-0.01	0	11	0	0
20-20-0.001	0	6	11	0
30-30-0.01	0	11	16	0
30-30-0.001	0	11	5	5
40-40-0.01	0	16	11	11
40-40-0.001	0	11	16	5
50-50-0.01	0	5	16	11
50-50-0.001	0	5	5	5
AV	1	9	7.95	5

Table 4.9
Misclassifications of the VT beats

Architecture	N	R	AFIB	L
10-0.01	0	0	0	10
10-0.001	5	0	0	5
20-0.01	4	0	3	10
20-0.001	5	0	0	0
30-0.01	0	0	4	0
30-0.001	0	5	4	4
40-0.01	0	0	0	9
40-0.001	0	0	0	10
50-0.01	0	0	14	10
50-0.001	0	0	0	7
10-10-0.01	4	0	8	14
10-10-0.001	0	0	0	9
20-20-0.01	4	0	0	9
20-20-0.001	0	0	0	9
30-30-0.01	0	0	0	0
30-30-0.001	0	0	0	0
40-40-0.01	5	0	0	9
40-40-0.001	0	0	4	9
50-50-0.01	4	0	0	14
50-50-0.001	0	4	4	0
AV	2	0	2.05	6

According to the misclassification percentages shown in the Tables through 4.8 to 4.12, one example beat for the Normal beat is taken into account to find out why the Normal beats are put into the RBB class mostly. The beat shown in Figure 4.1

Table 4.10
Misclassifications of the RBB beats

Architecture	N	VT	AFIB	L
10-0.01	0	0	4	0
10-0.001	12	0	4	0
20-0.01	8	0	4	0
20-0.001	8	0	8	0
30-0.01	12	0	0	0
30-0.001	0	0	0	0
40-0.01	0	0	0	0
40-0.001	12	0	0	0
50-0.01	0	0	0	0
50-0.001	0	0	0	0
10-10-0.01	12	0	0	0
10-10-0.001	4	0	4	8
20-20-0.01	0	0	4	0
20-20-0.001	0	0	0	0
30-30-0.01	8	0	0	0
30-30-0.001	4	0	0	0
40-40-0.01	8	0	0	0
40-40-0.001	0	0	0	0
50-50-0.01	0	0	0	0
50-50-0.001	0	0	0	0
AV	4	0	1.4	0

Table 4.11
Misclassifications of the AFIB beats

Architecture	N	VT	R	L
10-0.01	35	0	0	0
10-0.001	20	0	0	0
20-0.01	30	0	0	20
20-0.001	15	0	0	5
30-0.01	30	0	0	10
30-0.001	5	0	0	0
40-0.01	10	0	0	0
40-0.001	20	0	0	0
50-0.01	10	0	0	0
50-0.001	20	0	0	0
10-10-0.01	0	0	0	0
10-10-0.001	10	5	0	15
20-20-0.01	10	0	0	0
20-20-0.001	10	0	5	0
30-30-0.01	20	0	0	0
30-30-0.001	15	10	0	0
40-40-0.01	20	0	5	0
40-40-0.001	10	0	0	10
50-50-0.01	0	0	0	0
50-50-0.001	5	0	0	0
AV	15	1	0.5	3

is one of the Normal test input that is tested and failed. The beat was put into the RBB class. The beat belongs to the recording 222 in the MIT BIH database and the recording contains noise in some areas as well as DC offset.

Table 4.12
Misclassifications of the LBB beats

Architecture	N	VT	R	AFIB
10-0.01	0	10	5	0
10-0.001	0	5	0	0
20-0.01	10	5	0	0
20-0.001	0	0	0	0
30-0.01	15	0	0	0
30-0.001	0	15	0	0
40-0.01	5	0	0	5
40-0.001	5	0	0	0
50-0.01	0	10	5	0
50-0.001	0	0	0	0
10-10-0.01	10	0	0	0
10-10-0.001	5	0	0	5
20-20-0.01	5	10	0	10
20-20-0.001	0	0	0	0
30-30-0.01	5	0	0	5
30-30-0.001	0	10	0	0
40-40-0.01	0	5	5	0
40-40-0.001	0	10	0	0
50-50-0.01	0	0	0	0
50-50-0.001	0	0	0	0
AV	3	4	0.8	1.25

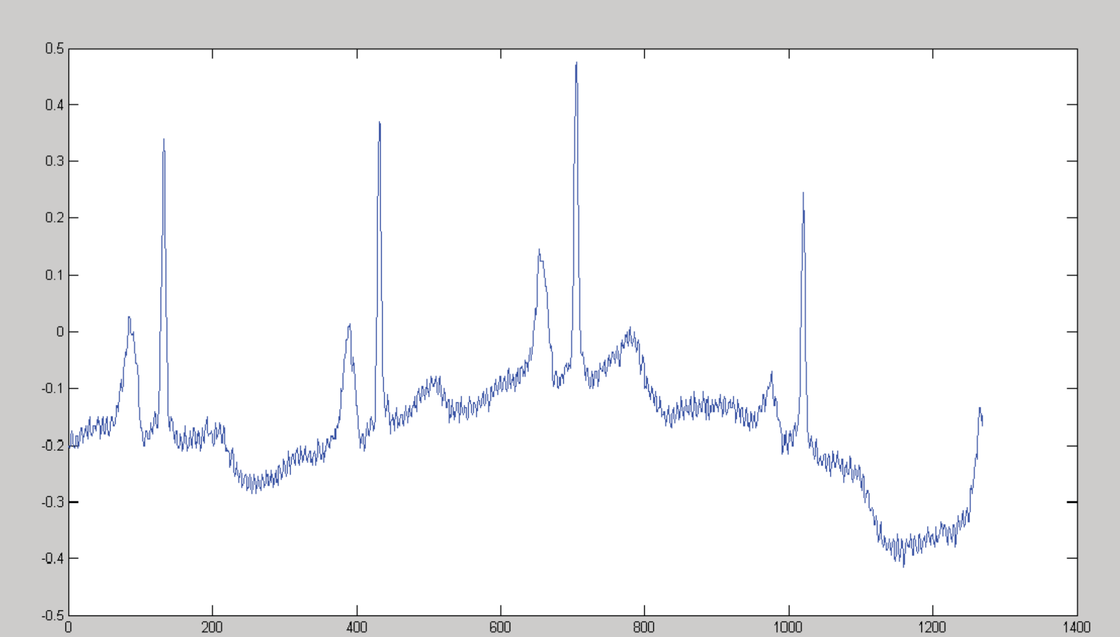


Figure 4.1 Misclassified Normal Beat

The test input for the window shown in Figure 4.1 is displayed in Figure 4.2.

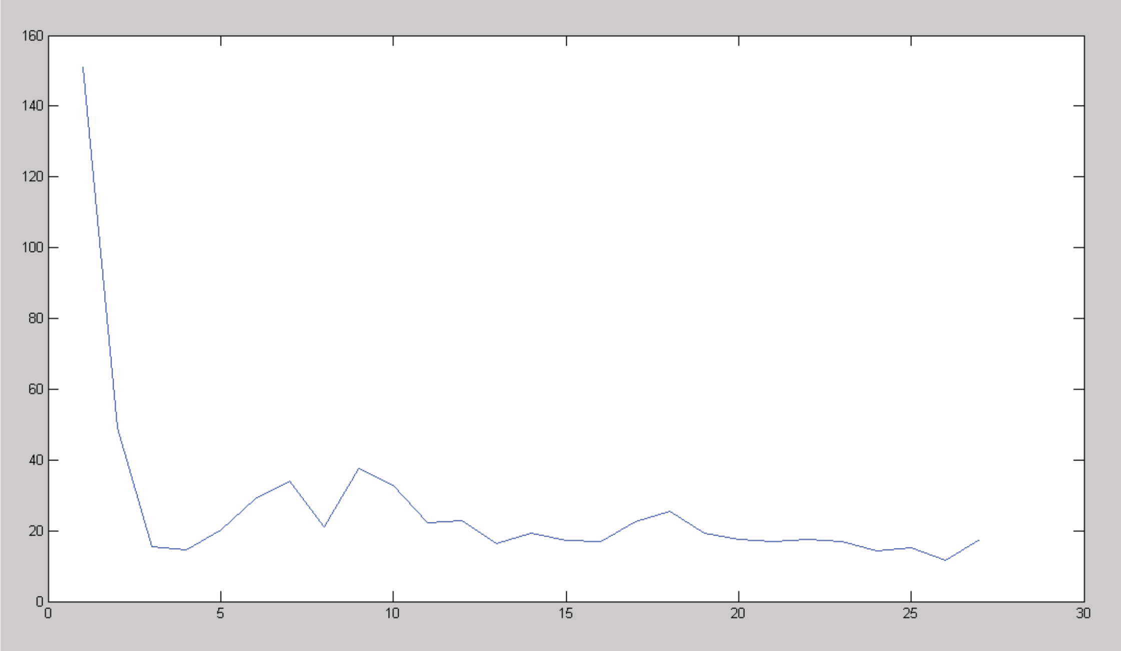


Figure 4.2 Misclassified Normal Beat Test Input

Comparing the average N input signal and the average RBB input signal, the classification result is not surprising. In Figure 4.3 the average N input signal is shown.

The misclassified signal looks more like the average RBB signal shown in Figure 4.4 rather than the average N beat.

Furthermore, when looking at the misclassifications in the VT class, it was noticed that all the failing signals were from the VT+N windows. The network was not able to detect the VT beats with the presence of the N beats in the same window.

Figure 4.5 shows that the VT+N beats have additional frequency components which make them look more like the LBB signals.

Please see misclassified VT+N, average VT and average LBB beats in the Figures through 4.5 to 4.7.

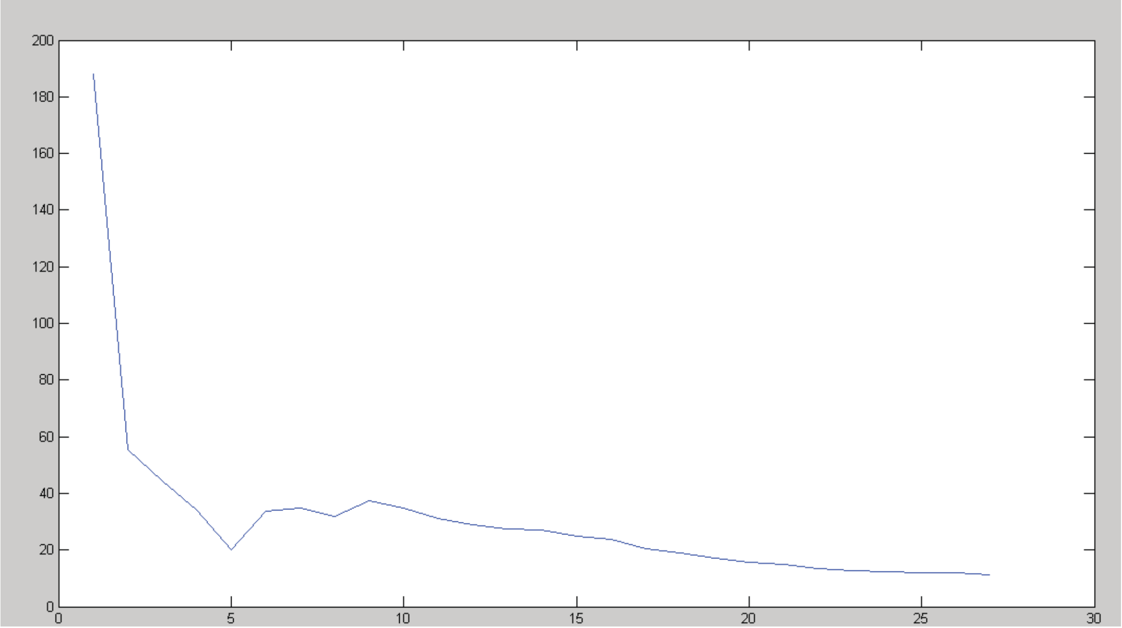


Figure 4.3 Average Normal Input Signal

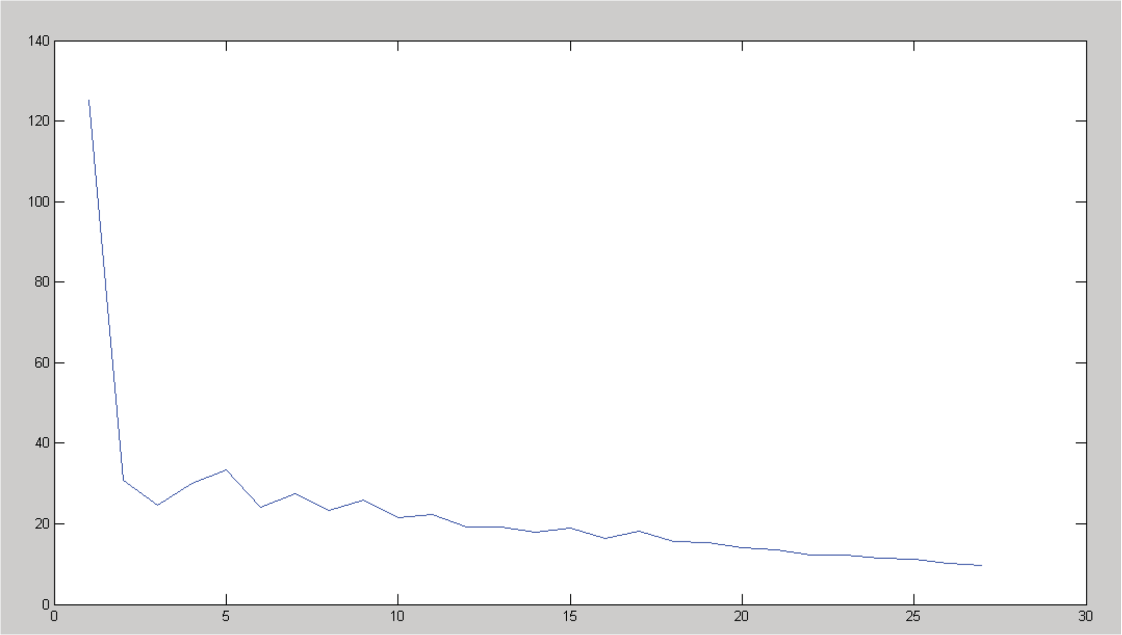


Figure 4.4 Average RBB Input Signal

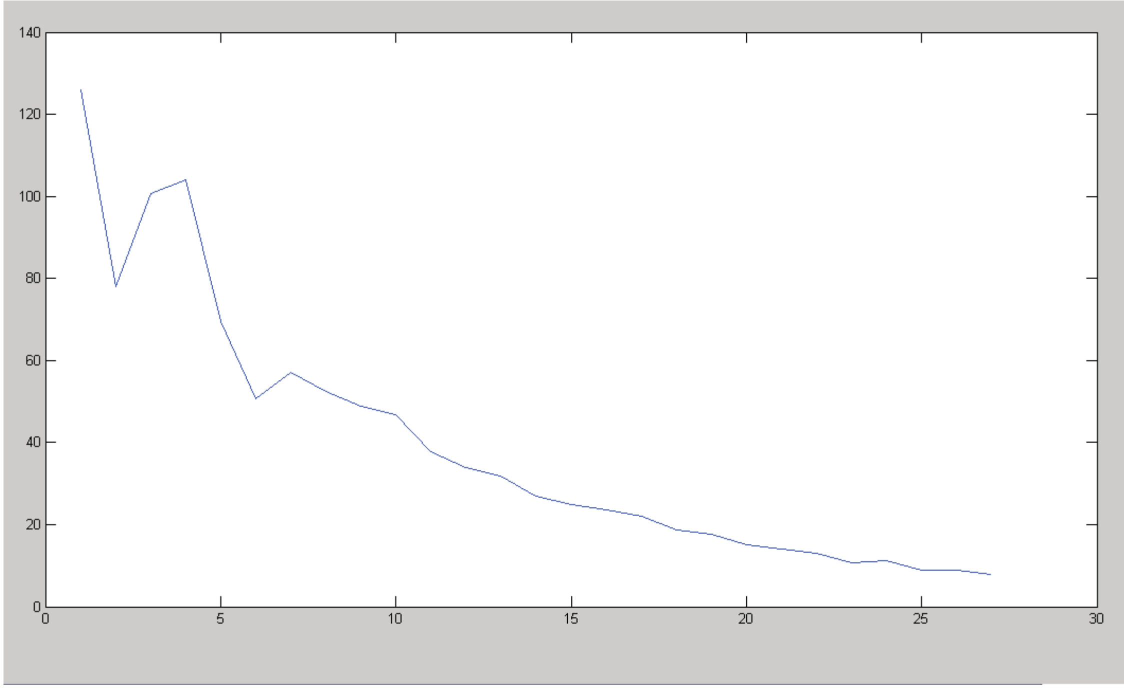


Figure 4.5 Misclassified VT+N Beat

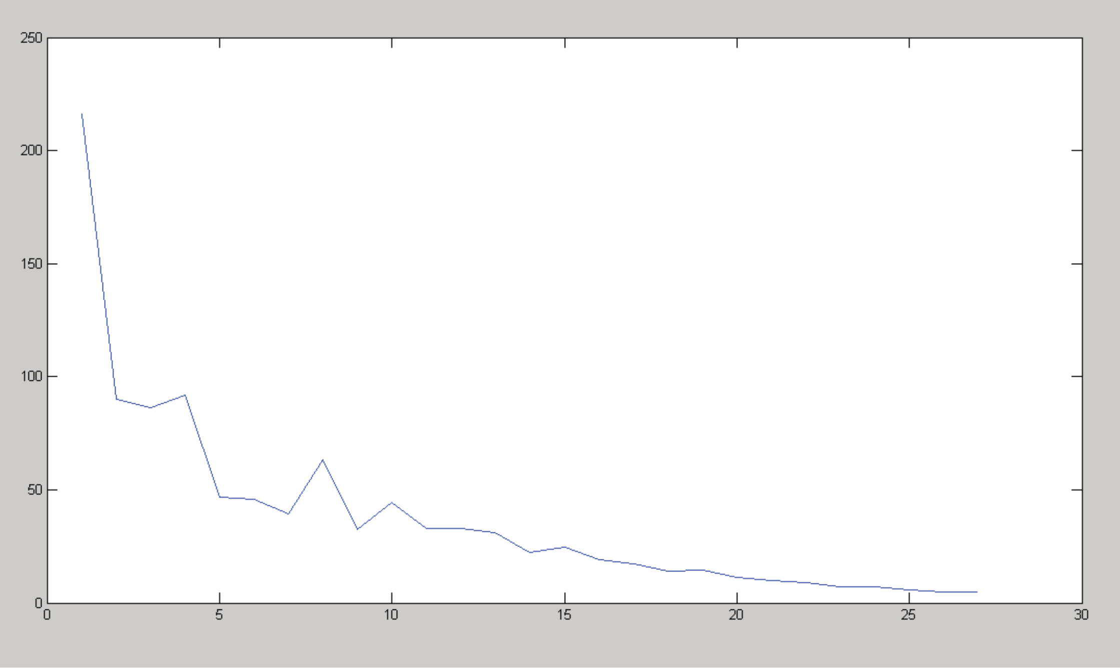


Figure 4.6 Average VT Input

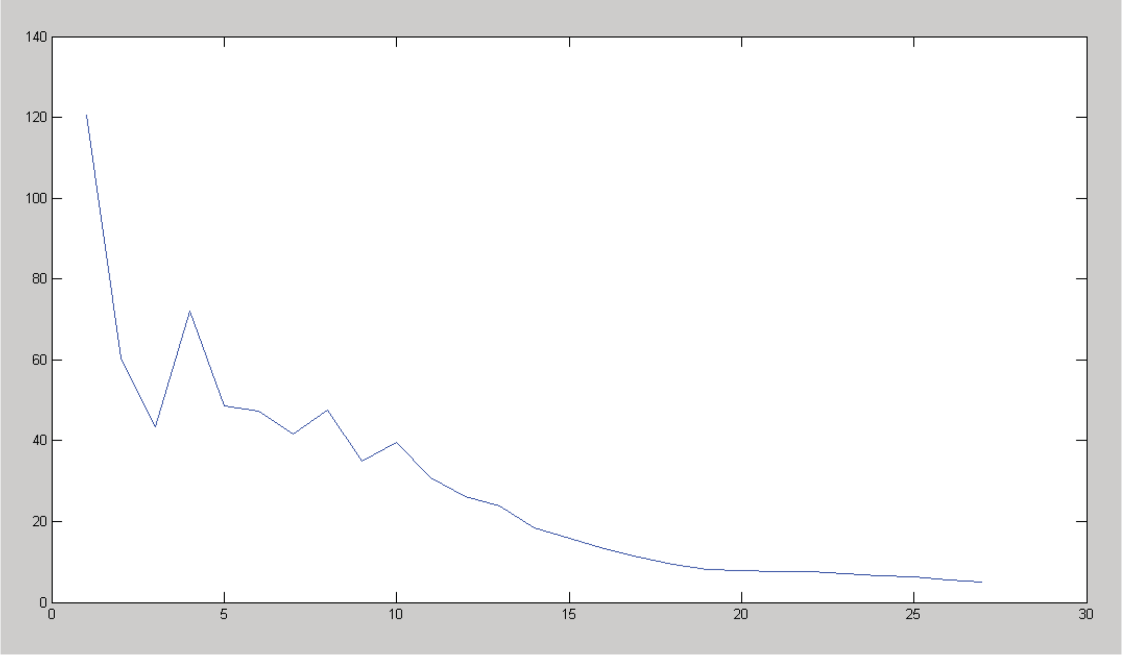


Figure 4.7 Average LBB Input

5. Conclusions

In this thesis study, an ANN based classifier which uses the Fourier Transform of a predefined window which consists of at least 3 beats together is presented.

5 types of beats are used to train the ANN. These beat types include Normal Beat, Right Branch Block, Left Branch Block, Ventricular Tachycardia and Atrial Fibrillation. The ANN was trained using Back Propagation learning algorithm with 1 and 2 hidden layered architectures. The data set extracted from original database was separated into two ; train and test set. The test set consisted of 40% of the extracted data and was not introduced to the network until the training was completed. The feature extraction was performed in 3 steps. First step was to extract windows from the original signal to contain at least 3 beats in the same windows. In the second step, the Fourier transforms of the windows were taken. The transformed signals were then post processed to get the final data to be used in training. After the trainings were completed, the ANN was tested with the test inputs created at the beginning. To compare the network test results, confusion matrices were datafilled during testing. According to the tests, the best performing network was able to classify with 80% success rate. This success rate is in the classification on overall test inputs with the presence of a threshold check at the output layer. In order to make a decision on the classification a threshold value of 0.5 was chosen. When the network produced its results for the inputs, the results were saved and compared with the desired outputs. The absolute value of the error at the output layer is calculated. The node which had an absolute value less than 0.5 was counted as correctly classified. In addition to the test set, the network was tested with the training inputs that it was trained with once more to find out how accurately it learned the training inputs. Results showed that the best performing network showed 89.6% accuracy in the training inputs. Results also showed that by removing the threshold and using a Winner Takes All method at the output layer, the most successful network performed better and gave higher percentage of correctly classified inputs.

Table 5.1 summarizes the accuracy of the best performing network:

Table 5.1
Accuracy of the best performing network

Decision Method	0.5 threshold (%)	Winner Takes All (%)
ANN tested with training inputs	89.6	93.2
ANN tested with test inputs	80	82

In addition to the classification decision method at the output layer, the study showed that the more the examples in the training set of the ANN are, the more successful the network is. It is assumed that if there are more examples for different type of windows, the network will understand the difference better and it will do less misclassification. Therefore with a sufficient amount of training input, this network could perform better and the classification could be more reliable.

Considering all the confusion matrices created after each training attempt for both 1 hidden layered networks or 2 hidden layered networks, it was noticed that the network is sensitive to the noise and DC offset in the signal. In addition to these, network is having difficulty in classifying the beats with the presence of other beats in the window which cause a change in the frequency components of the input signal.

Further work may be performed by extracting more and more data and creating larger input sets to overcome the adverse affects of these sensitivities in the network.

The Fourier Transform of a window would be a useful method in classifying ECG arrhythmia beats whenever there is sufficient amount of training examples.

REFERENCES

1. Niwas, S. I., R. S. S. Kumari, and D. V. Sadasivam, "Artificial neural network based automatic cardiac abnormalities classification," *IEEE, ICCIMA*, May 2005.

2. Olmez, T., Z. Dokur, and E. Yazgan, "Classification of ecg waveforms using a novel neural network," *IEEE*, 1998.

3. Dokur, Z., and T. Olmez, "Ecg beat classification by a novel hybrid neural network," *Computer Methods and Programs in Biomedicine*, 2000.

4. Celler, B. G., "Low computational cost classifiers for ecg diagnosis using neural networks," *Proceedings of the 20th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Vol. 20, 1998.

5. Kanapathipillai, M., I. Jouny, and P. Hamilton, *Adaptive Wavelet Representation and Classification of ECG Signals*, IEEE, 1994.

6. Yang, M., W. Hu, and L. Shyu, *ECG Events Detection and Classification Using Wavelet and Neural Networks*, IEEE 19th International Conference, 1997.

7. Lee, S. C., *Using a Translation Invariant Neural Network to Diagnose Heart Arrythmia.* PhD thesis, The Jonhs Hopkins University Applied Physics Laboratory, USA.

8. Neagoe, V., and S. Grunwald, "A neuro-fuzzy approach to classification of ecg signals for ischemic heart disease diagnosis," *AMIA Annu Symp Proc.*, no. 2003, pp. 494–498, 2003.

9. Jiang, W., S. Kong, and G. Peterson, *ECG Signal Classification using Block-based Neural Networks*, 2005.

10. 111 8th Ave 7th Floor New York, NY 10011: WebMD Network, 2009. E-medicine health website at http://www.emedicinehealth.com/electrocardiogram/.

11. P.O. Box 78350 San Francisco, CA 94107-8350 USA: Wikimedia Foundation Inc., 2009. Article is availabe on Wikipedia at http://en.wikipedia.org/wiki/Electrocardiography.

12. 800 Commissioners Road East PO Box 5010 London, Ontario N6A 5W9: London Health Sciences Centre, 2008. London Health Sciences Centre website at http://www.lhsc.on.ca.

13. Haykin, S., *Neural Networks*, USA: Macmillan College Publishing Company, 1994.

14. MIT Room E25-505A 77 Massachusetts Avenue Cambridge, MA 02139 USA: National Institute of Biomedical Imaging and Bioengineering, National Institute of General Medical Sciences, 2009. Physionet Data read website at http://www.physionet.org.

15. Öztemel, E., *Yapay Sinir Ağları*, Istanbul: Papatya Yayınevi, 2003.

16. Balomenos, T., K. Rapantzikos, and N. Tsapatsoulis, *On the use of ECG waveforms for Health Status Estimation.*

17. Dokur, Z., T. Ölmez, and E. Yazgan, *Comparison of discrete wavelet and Fourier transforms for ECG beat classification*, IEEE Electronics Letters Online No: 19991095, 1999.

18. Szilkgyi, S., *Event Recognition, Separation And Classification From ECG Recordings*, IEEE 20th Annual International Conference, 1998.

19. Minami, K., H. Nakajima, and T. Toyoshima, "Real-time discrimination of ventricular tachyarrhythmia with fourier-transform neural network," *IEEE Transactions on Biomedical Engineering*, Vol. 46, no. 2, 1999.

20. Nabney, I. T., D. J. Evans, J. Tenner, and L. Gamlyn, *Benchmarking Beat Classification Algorithms*, Cardionetics Institute of Bioinformatics, Aston University.

21. Stanislaw, O., and L. Hoai, *Analysis of features for efficient ECG signal classification using neuro-fuzzy network*, IEEE, Piscataway NJ, ETATS-UNIS (2004) (Monographie), 2004.

22. Park, K. L., and J. Lee, *ECG Signal Processing using Basis Function of Independent Component Analysis*, IAESTED, 2005.

23. Zhang, D., G. Tan, and J. Hao, "Fractal random walk and classification of ecg signal," *International Journal of Hybrid Information Technology*, Vol. 1, January 2008.

24. Yao, J., Q. Gan, and X. Zhang, "Pruning algorithm in wavelet neural network for ecg signal classification," *Engineering in Medicine and Biology Society, Proceedings of the 20th Annual International Conference of the IEEE*, Vol. 3, pp. 1482–1485, November 1998.

25. Zhao, Q., and L. Zhang, "Ecg feature extraction and classification using wavelet transform and support vector machines, neural networks and brain," *ICNN&B '05 International Conference*, Vol. 2, pp. 1089–1092, 2005.

26. Yang, M. Y., W. Hu, and L. Shyu, “Ecg events detection and classification using wavelet and neural networks,” *Proceedings of the 19th International Conference, IEEE.*

27. Kadous, M. W., *http://www.cse.unsw.edu.au/ waleed/phd/html/node37.html.* PhD thesis.

APPENDIX A. Accuracy Tables for Previous Studies

Table A.1
Overall Performance of the Method Proposed [1]

ECG Signal Type	Number of Data Sets Used for Training	Number of Data Sets Used for Testing	Training Accuracy in %	Testing Accuracy in %
NB	25	20	100	98.90
LBBB	20	15	100	96.77
RBBB	30	25	100	96.27
AP	25	15	100	91.35
SP	15	10	100	92
PVC	40	25	100	98.92
AF	20	15	100	95.96
VF	20	15	100	95.78
SSS	30	20	100	98.27
FVN	25	20	100	96.41
Total	250	180	100	99.02

Table A.2
Comparison of Different ECG Classifiers [1]

Method	Number Arrhythmia Types	Accuracy in %
Proposed Method	10	99.02
Mixture of Experts	4	94
Fuzzy Hybrid Neural Network	7	96.06
Discrete Wavelet Transform	13	97
Fourier Transform Neural Network	3	98
Discrete Fourier Transform	10	89.40

Table A.3
Performance Results of the ANN [2]

Test Results	Kohonen	Grow and Learn	Multi Layer Perceptron	Neural Network Trained by Genetic Algorithms
N	45/50	47/50	46/50	49/50
L	43/50	45/50	45/50	48/50
V	41/50	44/50	43/50	46/50
P	43/50	47/50	42/50	48/50
A	43/50	45/50	43/50	47/50
R	45/50	46/50	45/50	49/50
E	46/50	47/50	44/50	49/50
Training Time	15 sec.	13 sec.	300 sec.	20 sec.
Number of Nodes	42	18	30/20/30/7	10

Table A.4
Results of the ANN Models [2]

Architecture	η	α	Sum of Square Errors	Epochs	Training Sets	Testing Sets
12-30-12-7	0.995	0.995	<0.01	41278	100%	99.16%
12-12-12-7	0.995	0.995	<0.01	62183	100%	98.88%

Table A.5
Percentage of Correct Diagnosis [7]

	Patients	Segments
100% Correct	29	163
99%-90% Correct	19	23
90%-80% Correct	3	6
80%-70% Correct	0	4
<70% Correct	0	1
Could not be trained	1	6
Total	52	203

APPENDIX B. Matlab Code

Matlab code that is used to read the ECG recordings downloaded from MIT BIH database [14].

```
    clc; clear all;
%—— SPECIFY DATA ————————————————————
PATH= ’<write path>’; % path, where data are saved
HEADERFILE= ’<ECG recording>214.hea’; % header-file in text format
ATRFILE= ’<ECG recording.atr’; % attributes-file in binary format
DATAFILE=’<ECG recording.dat’; % data-file
SAMPLES2READ=650000; % number of samples to be read
```

```
    fprintf(1,’> WORKING ON %s ...’, HEADERFILE);
signalh= fullfile(PATH, HEADERFILE);
fid1=fopen(signalh,’r’);
z= fgetl(fid1);
A= sscanf(z, ’%*s nosig= A(1); % number of signals
sfreq=A(2); % sample rate of data
clear A;
for k=1:nosig
z= fgetl(fid1);
A= sscanf(z, ’%*s %d %d %d %d %d’,[1,5]);
dformat(k)= A(1); % format; here only 212 is allowed
gain(k)= A(2); % number of integers per mV
bitres(k)= A(3); % bitresolution
zerovalue(k)= A(4); % integer value of ECG zero point
firstvalue(k)= A(5); % first integer value of signal (to test for errors)
end;
fclose(fid1);
clear A;
```

```
    %—— LOAD BINARY DATA ——————————————————
if dformat = [212,212], error(’this script does not apply binary formats different to 212.’); end;
signald= fullfile(PATH, DATAFILE); % data in format 212
fid2=fopen(signald,’r’);
A= fread(fid2, [3, SAMPLES2READ], ’uint8’)’; % matrix with 3 rows, each 8 bits long, = 2*12bit
fclose(fid2);
M2H= bitshift(A(:,2), -4);
M1H= bitand(A(:,2), 15);
PRL=bitshift(bitand(A(:,2),8),9); % sign-bit
PRR=bitshift(bitand(A(:,2),128),5); % sign-bit
M( : , 1)= bitshift(M1H,8)+ A(:,1)-PRL;
M( : , 2)= bitshift(M2H,8)+ A(:,3)-PRR;
if M(1,:)  = firstvalue, error(’inconsistency in the first bit values’); end;
switch nosig
case 2
M( : , 1)= (M( : , 1)- zerovalue(1))/gain(1);
M( : , 2)= (M( : , 2)- zerovalue(2))/gain(2);
TIME=(0:(SAMPLES2READ-1))/sfreq;
case 1
M( : , 1)= (M( : , 1)- zerovalue(1));
M( : , 2)= (M( : , 2)- zerovalue(1));
M=M’;
M(1)=[];
sM=size(M);
sM=sM(2)+1;
M(sM)=0;
M=M’;
M=M/gain(1);
TIME=(0:2*(SAMPLES2READ)-1)/sfreq;
otherwise % this case did not appear up to now!
```

```
% here M has to be sorted!!!
disp(’Sorting algorithm for more than 2 signals not programmed yet!’);
end;
clear A M1H M2H PRR PRL;
fprintf(1,’> LOADING DATA FINISHED’);


    %—— LOAD ATTRIBUTES DATA ————————————————————-
atrd= fullfile(PATH, ATRFILE); % attribute file with annotation data
fid3=fopen(atrd,’r’);
A= fread(fid3, [2, inf], ’uint8’)’;
fclose(fid3);
ATRTIME=[];
ANNOT=[];
sa=size(A);
saa=sa(1);
i=1;
while i<=saa
annoth=bitshift(A(i,2),-2);
if annoth==59
ANNOT=[ANNOT;bitshift(A(i+3,2),-2)];
ATRTIME=[ATRTIME;A(i+2,1)+bitshift(A(i+2,2),8)+...
bitshift(A(i+1,1),16)+bitshift(A(i+1,2),24)];
i=i+3;
elseif annoth==60
% nothing to do!
elseif annoth==61
% nothing to do!
elseif annoth==62
% nothing to do!
elseif annoth==63
hilfe=bitshift(bitand(A(i,2),3),8)+A(i,1);
hilfe=hilfe+mod(hilfe,2);
i=i+hilfe/2;
```

```
else
ATRTIME=[ATRTIME;bitshift(bitand(A(i,2),3),8)+A(i,1)];
ANNOT=[ANNOT;bitshift(A(i,2),-2)];
end;
i=i+1;
end;
ANNOT(length(ANNOT))=[]; % last line = EOF (=0)
ATRTIME(length(ATRTIME))=[]; % last line = EOF
clear A;
ATRTIME= (cumsum(ATRTIME))/sfreq;
ind= find(ATRTIME <= TIME(end));
ATRTIMED= ATRTIME(ind);
ANNOT=round(ANNOT);
ANNOTD= ANNOT(ind);


      %—— DISPLAY DATA ————————————————
figure(1); clf, box on, hold on
plot(TIME, M(:,1),’r’);
if nosig==2
plot(TIME, M(:,2),’b’);
end;
for k=1:length(ATRTIMED)
text(ATRTIMED(k),0,num2str(ANNOTD(k)));
end;
xlim([TIME(1), TIME(end)]);
xlabel(’Time / s’); ylabel(’Voltage / mV’);
string=[’ECG signal ’,DATAFILE];
title(string);
fprintf(1,’> DISPLAYING DATA FINISHED’);


      % —————————————————————-
fprintf(1,’> ALL FINISHED’);
```

APPENDIX C. Statistical Summary of the Signals

The statistical summary of the records used in this study can be found online at [14]. One example for the record number 100 is shown in Table C.1. In this table, there are 367 annotated Normal beats in the first 5 minutes of the recording, and 1872 beats after the first 5 minutes. Likewise, there are 4 annotated Atrial Premature beats in the first 5 minutes and 29 during the rest of the recording. There are 2273 annotated beats in the recording #100 in total. The details of the remaining recordings can be found online at [14].

Table C.1
Details of Record #100

Beats	**Before 5:00**	**After 5:00**	**Total**
Normal	367	1872	2239
APC	4	29	33
PVC	-	1	1
Total	371	1902	2273

C.1 Symbols Used in Annotation

An expanded and updated version of the table C.2 can be found online at [14].

Table C.2
Symbols Used in Annotation

Symbol	***Meaning***
. or N	Normal beat
L	Left bundle branch block beat
R	Right bundle branch block beat
A	Atrial premature beat
a	Aberrated atrial premature beat
J	Nodal (junctional) premature beat
S	Supraventricular premature beat
V	Premature ventricular contraction
F	Fusion of ventricular and normal beat
[	Start of ventricular flutter/fibrillation
!	Ventricular flutter wave
]	End of ventricular flutter/fibrillation
e	Atrial escape beat
j	Nodal (junctional) escape beat
E	Ventricular escape beat
/	Paced beat
f	Fusion of paced and normal beat
x	Non-conducted P-wave (blocked APB)
Q	Unclassifiable beat
|	Isolated QRS-like artifact