Implementation of Shallow Neural Networks for ECG Analysis and Heartbeat Classification

Constantinos D. Balaskas

Supervising Professor: Kostantinos Siozios

Thessaloniki, June 2018
Short abstract

Heart diseases are unfortunately a more common cause of death than any other disease in the world, according to the World Health Organization. Therefore, it is imperative for patients who suffer from such conditions to be constantly monitored through procedures like electrocardiography, which produces the ECG signal, the diagram representing the heart’s variating electrical voltage through time. For this to happen, a quick and accurate diagnosis has to take place. Machine learning algorithms are well suited to aid in diagnosis because of their automated nature. In this thesis we explore the implementation of a specific category of machine learning algorithm, the artificial neural network, for analyzing an ECG signal and detecting the heart beats of a patient that indicate a healthy condition. Generally, a neural network is consisted of processing units called neurons through which a signal propagates, depending on the connections between those units. For such a network to perform accurately, those connections (or weights) are finely tuned during the training process of the network, after which it produces the desired output. In our case, that output was the classification process of every heart beat that was presented as input into two distinct classes: normal or abnormal beats. Firstly, we attempted to create multiple networks, each one with a different set of parameters (elements per sample in the input data and number of neurons in the hidden layer) and evaluated the results. The networks were able to minimize their classification error down to 2%-3%, and functioned very well on assigning the heart beats to their true class. We concluded however that the network performance did not variate as expected, when those parameters altered, even when increasing the size of the input data set, because we dealt with the heart beats as raw data. For that reason, discrete wavelet transform was applied to the data as a feature extraction unit before the network was trained, giving in that way a unique identity to each pulse. Now that the signal was processed correctly, the results of training a network with a large data set appeared encouraging: the network seemed to respond better to data with more elements per sample and it produced more accurate classifications as the hidden layer neurons increased, achieving classification accuracy up to 96% on all the available data. We were able to determine that the optimal network for classifying heart beats into those two classes had to have 14 neurons in the hidden layer and the signal that it received had to be processed in a way that each heart beat was separated by 128 samples at both sides of its peak, in order to receive more information about it. After we settled the optimal configuration for a network, we tested its
accuracy. First, we saw that it responded excellently to even small data sets after being trained, providing accuracy on the level of 99%. Also, it seemed to increase its accuracy when the amount of data that were used for its training increased. Finally, we exported the structure of the neural network to a C program that emulated its function and could be used in a processing unit for accurate classification of heart beats from an ECG signal.
Extended abstract

Cardiovascular diseases are unfortunately the main cause of sudden deaths in the world, according to the World Health Organization. They are a group of diseases that concern the heart or the blood vessels and they include various diseases of the arteries, most commonly the cardiac arrest of the coronary artery. In order to prevent sudden deaths that are caused by CVD’s, suffering patients need to be diagnosed in a quick automated and accurate manner. Following the diagnosis, the need for constant monitoring of those patients is also imminent. A great and efficient way to achieve that is by using electrocardiography, a simple and non-invasive procedure that gives information about the condition of the heart. By applying electrodes to the patients’ chest to the appropriate places, a physician is able to monitor the depolarization and repolarization cycle of the heart muscle, which is depicted in the signal that this cycle produces: the electrocardiogram. Easy to identify, harder to analyze, the electrocardiogram is one of the most useful signals in our possession for understanding the condition of the heart. As a result, its analysis can be of much aid to doctors for diagnosing heart diseases. An accurate way of diagnosing such diseases is by using machine learning algorithms.

Machine learning algorithms are computational models that can process large amounts of data and learn a certain procedure with high accuracy without the need to be reprogrammed. The concept of learning is key in those algorithms and it entails the procedure of training, through which the algorithm is provided with input examples and their corresponding desired outputs called targets. By knowing what its output should be, the model can check the difference between its actual output and the target and then proceed to improve its structure so that the difference can be minimized. The algorithm that was used in this thesis is the artificial neural network. Taking inspiration from the neural system of living organisms, an artificial neural network has the ability to propagate information through neurons, the fundamental processing unit of a NN. Neurons are organized in groups called layers, starting from the input layer to the output layer. What determine the output of a network are actually the connections between neurons of different layers called weights, which is also the parameters the network tries to optimize during the training procedure. A finely tuned network has well defined values for every weight.

The nature of machine learning algorithms, and neural networks specifically, bonds well with the need for automated and highly accurate diagnosis of heart diseases through the electrocardiogram.
signals. Around those lines was the goal of this thesis: we wanted to create a neural network that would be trained to receive the heart beats of the ECG from unknown patients and find those heart beats that indicated an abnormal condition, separating them from normal heart beats. The methodology that was originally proposed for the creation of such a network follows below:

1. **Read the Signals and Annotations**
2. **Create the Input and Target Vectors**
3. **Configure the Network**
4. **Train the Network**
5. **Evaluating the network performance**

Primarily, we had to find data that could be used for the training procedure. The MIT-BIH Arrhythmia Database was used to provide us with ECG signals from 40 different patients, along with the annotation label that indicated the condition of every heartbeat. Those conditions were summarized by the learner into two classes: normal and abnormal. The network would be able to classify unknown heart beats into those two classes. Every beat was a different input sample and had to be separated from the next. So we introduced a window size variable that determined the number of samples that would consist a single heart beat on both sides of the R-peak of the beat.

Before the network was trained, we had to set some parameters concerning its structure and the training procedure itself. Most importantly, we had to determine the amount of neurons in the hidden layer (the layer between the input and the output layer) as well as function for transferring information through the neurons and data division. For every patient we trained multiple networks for a set of parameters (window size and neurons in the hidden layer) and evaluated the classification accuracy of the network concerning the two classes. The results that were received indicated that although we achieved a low amount of wrongly classified beats (around 2.6%), the performance of the network did not alter as the parameters of its configuration changed. We attempted to correct that problem by enlarging the data set of the training procedure, meaning that we organized all the patients’ information into a single matrix. That way, there was less chance for the network to be unable to generalize the results for unknown data and we could eliminate the patient parameter from our exploration. Still we did not receive the behavior that we wanted from the network, even if the larger data set was a step towards the right direction. Indicatively, the variation of error percentage of classification between networks was 3.5% (from 5.4% to 1.9%).
The fault lied within our method of dealing with the heartbeats as nothing more than raw data. We needed to give a unique identity to each heart beat and feed that to the network for its training. As a result, a feature extraction unit was inserted into our methodology as can be seen below.

The feature extraction unit was implemented using wavelet functions and discrete wavelet transform as a result. That choice can be justified by the advantages wavelets offer for the representation of a signal with abrupt oscillations in both the time and frequency domain. The approximation coefficient of the 4th level of deconstruction of the signal was used as the training input, with our exploration still focused on determining the network’s behavior while its configuration changed. The results from this method showed us the behavior we were searching for: networks with more neurons in the hidden layer and more importantly window sizes proved to have higher classification accuracy. The variation of error percentages this time was much higher, fluctuating between 13.5% and 4%. The optimum configuration with the highest classification accuracy (error percentage of 4.027%) had 14 neurons in the hidden layer and the signal that it received had to be processed in a way that each heart beat was separated by 128 samples at both sides of its peak, in order to receive more information about it. We put the behavior of the network to the test, checking its response to retraining with a small set of unknown data as well as its behavior while increasing the amount of data used for training per patient. In both testing procedures the network behaved as expected, ensuring the validity of our feature extraction method. Finally, we emulated the function of our network by creating a C program that could be used as software in a processing unit as part of an embedded platform. That was possible since we could extract the weights from our trained network and were therefore able to calculate mathematically the output to any unknown input sample. The architectural structure would have to contain sensors that received, digitized and processed the ECG signal accordingly, but the result would be the same as that of the network: an accurate classifier to indicate the abnormal heart beats and separate them from the normal ones.
Εκτεταμένη Περίληψη

Οι καρδιοαγγειακές παθήσεις αποτελούν δυστυχώς την κυριότερη αιτία αιφνίδιων θανάτων, σύμφωνα με τον Παγκόσμιο Οργανισμό Υγείας. Περιλαμβάνουν ασθένειες της καρδιάς ή των αιμοφόρων αγγείων, με χαρακτηριστικό παράδειγμα την καρδιακή προσβολή, που είναι πάθηση της στεφανιαίας αρτηρίας. Για την πρόληψη των αιφνίδιων θανάτων που προκαλούν οι ασθένειες αυτές, είναι επιτακτική ανάγκη οι πάσχοντες να διαγνωσθούν εγκαίρως, με έναν ακριβή και αυτοματοποιημένο τρόπο. Η διάγνωση, εφόσον ανιχνευτεί κάποια ασθένεια, πρέπει να ακολουθείται από συνεχή παρακολούθηση, ώστε να αποφευχθούν τα χειρότερα συμπτώματα. Η ηλεκτροκαρδιογράφηση προσφέρει μία απλή μέθοδο για την λήψη πληροφοριών σχετικά με την κατάσταση της καρδιάς. Εφαρμόζοντας ηλεκτρόδια στο στήθος του ασθενούς, χωρίς να εισαχθεί κάποιο όργανο στο εσωτερικό, ένας γιατρός είναι σε θέση να παρακολουθήσει τον κύκλο αποπόλωσης και επαναπόλωσης της καρδιάς μέσω του σήματος που αυτή παράγει, του ηλεκτροκαρδιογραφήματος. Η καρδιογράφηση είναι εύκολη να αναγνωριστεί, δύσκολο να αναγνωριστεί και σίγουρα αποτελεί τη κύρια πηγή άντλησης πληροφοριών από την καρδιά. Η ανάλυση του λοιπόν, είναι πολύ βοηθητική προς την αιτική κουνότητα και μάλιστα υπάρχουν πολλοί τρόποι για να πραγματοποιηθεί. Ένας από τους πιο ακριβείς και εύχρηστους τρόπους είναι η χρήση αλγορίθμων μηχανικής μάθησης.

Οι αλγόριθμοι μηχανικής μάθησης είναι υπολογιστικά μοντέλα που έχουν την ικανότητα να επεξεργάζονται μεγάλο όγκο δεδομένων με στόχο να βελτιώσουν την απόδοσή τους σε μία συγκεκριμένη λειτουργία, χωρίς να χρειάζεται να επαναπρογραμματιστούν. Η έννοια της μάθησης είναι κομβική και συνυφασμένη με τη διαδικασία της εκπαίδευσης, όπου ο αλγόριθμος δέχεται παραδείγματα δεδομένων στην είσοδο του και προσπαθεί να βρει την κατάλληλη έξοδο, ώστε τελικά να εκτελεί τη διαδικασία αυτή σε άγνωστα δεδομένα. Στην πτυχιακή εργασία, ο αλγόριθμος μηχανικής μάθησης που χρησιμοποιήθηκε είναι τα τεχνητά νευρικά δίκτυα. Εμπνευσμένα από τα ζωντανά νευρικά συστήματα, ένα νευρικό δίκτυο δομείται από τρεις στάδια, ξεκινώντας από το επίπεδο εισόδου, που χρησιμοποιεί τη χρησιμοποιημένη έξοδο του δικτύου και αλλάζει κατά την εκπαίδευση και μεταξύ των επιπέδων με χρήση αλγορίθμων μηχανικής μάθησης.
ανάλογα με τη διαφορά της πραγματικής από την επιθυμητή έξοδο (στόχο), μέχρι να κατασταλάξουν στην τελική τους τιμή.

Ο στόχος της εργασίας βασίστηκε στα πλαίσια της φύσης των αλγορίθμων μηχανικής μάθησης και τα πλεονεκτήματα που προσφέρουν στην αυτοματοποιημένη διάγνωση μέσω ανάλυσης του ηλεκτροκαρδιογράφηματος. Στοχεύσαμε στην κατασκευή ενός δικτύου που θα δεχόταν ως είσοδο τους καρδιακούς παλμούς από άγνωστους ασθενείς και θα έδινε στην έξοδο πληροφορίες για το ποιοι παλμοί έδειχναν μη φυσιολογικοί, ξεχωρίζοντας τους από τους φυσιολογικούς. Η μεθοδολογία που αρχικά προτάθηκε για τη διαδικασία αυτή αποτυπώνεται παρακάτω.

Για την εκπαίδευση του δικτύου χρησιμοποιήθηκε η MIT-BIH Arrhythmia Database, που παρέχει τα καρδιογραφήματα 40 ασθενών μαζί με τις ταμπέλες – στόχους που έδειχναν την κατάσταση του κάθε παλμού για κάθε ασθενή (φυσιολογικός ή μη φυσιολογικός παλμός). Για να γίνει η αντιστοίχιση στόχου και παλμού, έπρεπε ο κάθε παλμός να διαχωριστεί από τους υπολοίπους με μία μεταβλητή παραθύρου που θα απέκοπτε συγκεκριμένο αριθμό δειγμάτων εκατέρωθεν της κορυφής R του παλμού. Συνδυάζοντας τους παλμούς σε πίνακες εισόδου και στόχων για την εκπαίδευση, έπρεπε να καθοριστούν μερικές παράμετροι που αφορούσαν τη δομή του δικτύου όσο και τον τρόπο εκτέλεσης της εκπαίδευσης. Ορίσαμε λοιπόν συναρτήσεις που αφορούσαν τη μεταφορά της πληροφορίας στο εσωτερικό του δικτύου, τον διαχωρισμό των δεδομένων και τη συνάρτηση εκπαίδευσης. Υπό διερεύνηση τέθηκε ο αριθμός των νευρώνων στο κρυφό επίπεδο (επίπεδο μεταξύ εισόδου και εξόδου). Οπότε προσπαθήσαμε να μελετήσουμε τη συμπεριφορά του δικτύου, εκπαιδεύοντας πολλαπλά δίκτυα με διαφορετικές μορφοποιήσεις όσον αφορά το μήκος του παραθύρου και τον αριθμό των νευρώνων στο κρυφό επίπεδο, για κάθε ασθενή. Τα αποτελέσματα στην ταξινόμηση των παλμών στις δύο κλάσεις (φυσιολογικοί, μη φυσιολογικοί) φάνηκαν εξαιρετικά, με την ακρίβεια να φτάνει το 97%. Όμως, τα αποτελέσματα στην απόδοση του δικτύου δεν άλλαζαν όσο διαφοροποιούσαν η μορφοποίηση του δικτύου, δείχνοντας κάποιο λάθος στη μεθοδολογία μας. Παρόμοια σταθερή συμπεριφορά (διακύμανση λανθασμένων ταξινομήσεων 3.5%) παρατηρήσαμε και αυξάνοντας τα δεδομένα εισόδου στην εκπαίδευση, εξαλείφοντας τον διαχωρισμό των πληροφοριών κάθε ασθενούς. Διορθώσαμε τη μεθοδολογία μας,
όπως φαίνεται παρακάτω, εισάγοντας μία μονάδα εξαγωγής χαρακτηριστικών, που θα αντιμετώπιζε τους καρδιακούς παλμούς όχι ως ωμά δεδομένα, αλλά ως οντότητες με ξεχωριστή ταυτότητα την οποία θα μετέφερε στο δίκτυο.

Για την υλοποίηση της μονάδας αυτής χρησιμοποιήθηκαν οι συναρτήσεις wavelet, με το discrete wavelet transform να δίνει τα διανύσματα χαρακτηριστικών που αναζητούσαμε. Η επιλογή μας αυτή δικαιολογείται με τα πλεονεκτήματα που προσφέρουν τα wavelets στην αποσύνθεση και αναπαράσταση σημάτων με απότομες αλλαγές, τόσο στο χώρο του χρόνου όσο και στο χώρο των συχνοτήτων. Επαναλαμβάνοντας την ίδια διαδικασία εκπαίδευσης, χρησιμοποιήσαμε τον προσεγγιστικό συντελεστή του 4ου επιπέδου αποσύνθεσης του σήματος, με την προσπάθεια μας να συγκεντρώνεται στην εύρεση δικτύου που θα έδινε την μεγαλύτερη ακρίβεια σχετικά με τη μορφοποίησή του. Αυτή τη φορά η συμπεριφορά των δικτύων ακολουθούσε τη λογική, με την διακύμανση του ποσοστού λανθασμένων ταξινομήσεων να είναι της τάξης 9.5%. Σε μεγαλύτερα μήκη παραθύρου αντιστοιχούσαν δίκτυα με μεγαλύτερη ακρίβεια ταξινόμησης, καθώς υπήρχε μεγαλύτερη πιθανότητα να συμπεριληφθεί ολόκληρος ο παλμός στο διάνυσμα εισόδου. Η αύξηση του αριθμού νευρώνων στο κρυφό επίπεδο επηρέαζε σε μικρό βαθμό τα αποτελέσματα ταξινόμησης, δίνοντας μεγαλύτερη ακρίβεια σε δίκτυα με περισσότερους νευρώνες. Τελικά, το δίκτυο που ψάχναμε εμφάνισε ποσοστό λανθασμένων ταξινομήσεων 4.027% και μορφοποιήθηκε με 14 νευρώνες στο κρυφό επίπεδο, ενώ ο κάθε παλμός διαχωρίστηκε με 128 δείγματα δεδομένων εκατέρωθεν της κορυφής του. Για να ελέγξουμε τη συμπεριφορά του δικτύου αυτού, εφαρμόζαμε δύο μεθοδολογίες ελέγχου όπου αναλύσαμε τη συμπεριφορά ενός ήδη εκπαιδευμένου δικτύου στην επανεκπαίδευση του με λιγότερα άγνωστα δεδομένα και την απόκριση του σε άγνωστα δεδομένα. Η συμπεριφορά του δικτύου ήταν η αναμενόμενη και έδειξε πως ανταποκρίνεται εξαιρετικά και στις δύο περιπτώσεις. Το τελικό στάδιο της εργασίας περιελάβε την προσομοίωση της λειτουργίας του δικτύου σε ένα πρόγραμμα γραμμένο σε C ως λογισμικό για μία μονάδα επεξεργασίας παλμών. Η δυνατότητα υλοποίησης του προγράμματος βασίστηκε στις καλά καθορισμένες τιμές των βαρών του δικτύου, τις οποίες αν εξάγαμε κατάλληλα μπορούσαμε να υπολογίσουμε μαθηματικά την έξοδο του δικτύου για
οποιοδήποτε πρότυπο εισόδου. Η μονάδα επεξεργασίας θα ήταν μέρος μιας αρχιτεκτονικής δομής και θα τροφοδοτούταν με πληροφορίες του σήματος, αφού αυτές είχαν συλλεχθεί από αισθητήρες, είχαν ψηφιοποιηθεί και επεξεργαστεί κατάλληλα με εξαγωγή χαρακτηριστικών. Στόχος της αρχιτεκτονικής αυτής θα ήταν παρόμοιος με τη λειτουργία του δικτύου: ένας ακριβής ταξινομητής που θα διαχωρίζει τους μη φυσιολογικούς από τους φυσιολογικούς καρδιακούς παλμούς ενός ασθενή.
CONTENTS

INTRODUCTION ................................................................................................................. 12
  1.1 Problem Statement ................................................................................................. 12
  1.2 Chapter Review ..................................................................................................... 13

1. THEORETICAL BACKGROUND .................................................................................. 15
  1.1 Machine Learning ................................................................................................. 15
  1.2 Neural Networks .................................................................................................. 17
  1.3 The ECG Signal ..................................................................................................... 21
  1.4 MIT-BIH Arrhythmia Database ............................................................................. 24
  1.5 Wavelets and Discrete Wavelet Transform .......................................................... 25

2. NETWORK CREATION FOR DETECTING NORMAL BEATS ........................................ 28
  2.1 Preparation and Algorithmic Work Flow .............................................................. 28
  2.2 ECG Analysis and Creating the Vectors for the Network ..................................... 29
  2.3 Configuring the Neural Network and its Parameters ........................................... 30
  2.4 Training the Network and Creating Performance/Error Matrix ....................... 32
  2.5 Results and Improvements .................................................................................. 35
  2.6 Experimenting with Larger Input Data ................................................................. 45
  2.7 Results and Difficulties Exploration .................................................................... 48

3. FEATURE EXTRACTION .............................................................................................. 52
  3.1 Algorithmic Work Flow ....................................................................................... 52
  3.2 Applying Discrete Wavelet Transform and Extracting Coefficients ................. 52
  3.3 Creating the Network – Configuration Exploration ............................................. 53
  3.4 Results .................................................................................................................. 54

4. TESTING THE NETWORK AND EMULATING ITS FUNCTION IN C .......................... 59
  4.1 Retraining the Network with Unknown Patients .................................................. 59
  4.2 Checking the Networks Behavior with Single Patient Data ................................ 60
  4.3 Implementation in a C script ................................................................................ 63

5. CONCLUSIONS AND FUTURE WORK .................................................................. 66
  5.1 Conclusions ......................................................................................................... 66
  5.2 Future Work ......................................................................................................... 67

ACKNOWLEDGEMENTS ................................................................................................. 68
REFERENCES .................................................................................................................. 69
Introduction

1.1 Problem Statement

Since the beginning of the usage of electronic devices for medicinal purposes, there have only been few biomedical signals that could be described as more fundamental and useful than the Electrocardiogram (ECG). An electrocardiogram is a graphic product from an electrocardiograph, which displays the overall rhythm of the heart, and weaknesses in different parts of the heart muscle. It is the best way to measure and diagnose abnormal rhythms of the heart. It is evident that the ECG provides us with an easy and clear-cut method to understand and assess a patient’s heart condition.

Therefore, the analysis and digital deconstruction of the signal is imperative for understanding and studying the anomalies of the heart, aiming for a more accurate effort in detecting a faulty condition, preventing it or even curing it. For that reason, there is a need to automate the procedure of analyzing the series of heartbeats an electrocardiogram provides by using machines to generate the results. The problem that occurs however, lies with the complexity of the ECG or even the heart itself. There is no clear information that a machine can understand by analyzing the contents of the signal directly, since every pulse has its own range of properties and characteristics. A form of platform is required that allows the machine to understand the signal it is given, and eventually provide us with useful information about what that signal actually shows. That can be achieved with machine learning techniques, such as neural networks, which are able to process the large amounts of data that an ECG produces.

An artificial neural network (either shallow or deep) has the ability to train, using a large data set, in order to perform a certain function. Then it can generalize a result based on the data it is given as input, as long as its input can be categorized within the type of data it was trained with. So it is understandable that a neural network can be used to analyze a series of cardiac pulses and give useful information about the patient’s condition, without there being a distinct correlation between each pulse and what results from it. The only condition is that the network has to have been trained on a set of ECG pulses in order to be able to provide accurate results, such as the diagnosis
classification that is explored in this particular thesis. Generally, in classification, inputs are divided into two or more classes, and the learner must produce a model that assigns unseen inputs to one or more (multi-label classification) of these classes. This applies to ECG pulses perfectly, as each pulse can be labeled in a specific class (i.e. normal beat, Atrial premature beat etc.) during the training process. Later on, that class will be used to classify all the unknown to the network beats, according to the model built. Arrhythmia recognition is indeed important for computer assisted automatic diagnosis of cardiac diseases, which can be regarded as a problem of classification of ECG beats.

The use of neural networks in similar problems is justified by the variety of advantages it can offer. As mentioned above, a machine learning algorithm is capable of coping with large amounts of data which is perfect for studying the physiology of the heart in its complexity. The signals that indicate an anomaly (arrhythmia) show at random times in an electrocardiogram, if they even appear at all. Consequently, the ECG should be carried out during large periods of time so that it is possible to capture irregularities at a significant amount. Also, neural networks are structured in an easy manner for the learner to understand and adjust to the issue at hand. Using the neuron as their fundamental block, neural networks could be modified to have various layers and neurons and as a result are eligible to tackle a wide range of non-linear problems. Finally, a machine learning technique has the unique ability to recognize data it has never seen before, after going through the training procedure. The hidden layers inside the network are organized in a manner that allows them to recognize the important features in an input signal and to create an internal representation of the incoming patterns. That provides them with the foundation to compare these patterns with any unknown incoming pattern and be able to process them, without having seen them before. In ECG analysis, that is undoubtedly a very useful feature.

1.2 Chapter Review

In the first chapter, the theoretical background on which this thesis was based on is presented. Initially, some information about machine learning appears, as for example a definition of machine learning and the types of machine learning methods that can be used in a specific algorithm. Following, there can be found a few paragraphs about neural networks and specifically the way a signal is processed and propagates through its layers and neurons, in mathematical and non-mathematical terms. Some information about the ECG signal are presented next, concerning the
procedure as well as the way the signal is produced, through depolarization and repolarization of heartbeats. Finally, there is a brief mention of the MIT-BIH Arrhythmia Database, the database that was used to explore the ECG signals and train the neural networks.

The second chapter contains the algorithm that was used to create a network that can identify whether a heartbeat was normal or abnormal. Each step of the algorithm is explained in detail: the analysis of the ECG, the matching of each heartbeat with its corresponding annotation label, the configuration of the network regarding its size and its input vector, the training of the network and the evaluation of the results. After the results were evaluated and deductions were made, an exploration of the difficulties followed in an attempt to improve the network model by enlarging the input data.

In the third chapter discrete wavelet transform wavelet transform was used in an effort to deal with heart beats differently than raw data. The network’s input was modified accordingly and a thorough exploration was made to discover the validity of this method. The results, both numerical and graphical are explained in detail.

In the fourth chapter we present some crucial testing methods to check the performance of the network, in cases where it was pre-trained or not. To conclude the experimental part of the thesis, the function of the network was emulated in a C program, as a demonstration of a possible processing unit concerning a patient’s heart beats.

The fifth and final chapter contains the conclusions that the learner came to after completing this particular thesis. Also some ideas and directions regarding future work on the same topic are discussed.
1. Theoretical Background

1.1 Machine Learning

What is Machine Learning?

Learning is a fundamental capacity of the human conscious behavior. In the framework of Artificial Intelligence, it became a necessity to create computational systems that are able to learn by themselves, to achieve in other words what is called Machine Learning (ML).

“Machine Learning is the ability of a computational system to improve its performance while executing a specific task without having the need to be reprogrammed.” [9]

The foundation of Machine Learning is to create a machine that uses previous knowledge and experience to improve its accuracy while executing a procedure from a predefined class of operations [11]. So the work on ML is primarily focused on developing algorithms that improve their response to a certain process that has been assigned to them using their experience.

To achieve that, large amounts of data have to be used. Before the data are processed by the model, they have to be prepared and manipulated correctly, so that they are compatible to the model. A part of that data is used as training for the algorithm, in order for the model to cultivate the ability to correct its response according to its input. By evaluating the results during training and testing the model at the same time, the system corrects itself by inclining towards the better response. The concepts of training and correction through evaluation are fundamental in any machine learning algorithm [10]. A typical machine learning workflow can be seen in Figure 1.

![Figure 1: Example of a Machine Learning Workflow](image-url)
Types of Machine Learning

The field of Machine Learning can be broken down to three different methods of learning: Supervised Learning, Unsupervised Learning and Reinforcement Learning. For every problem in ML there is a suitable method of learning, and vice versa, as for every learning technique there exist at least one suitable algorithm that fits the problem [13].

1) Supervised Learning is the process during which the algorithm constructs a function that relates known inputs to desirable outputs, aiming to generalize that function for inputs whose corresponsive outputs are unknown. This method can be used in regression or of course, classification problems.

2) In Unsupervised Learning the algorithm constructs a model for an ensemble of data and has to discover and form correlations or groups within that data by creating patterns of unknown number and identity. UL can be used for association analysis or clustering problems.

3) Reinforcement Learning’s algorithms learn a process by directly interacting with their environment. They aim to maximize a certain reward function and have to decide by themselves what actions lead to better results. Application of RL include controlling robot movement, gaming AI etc.

In this particular thesis, we explore a classification problem using Supervised Learning. Therefore, we can expand on that subject. On SL models, along with the training data set, labels are provided that indicate the desired response of the model on every index of the dataset. These are called targets and are used by the algorithm to check if the output it produced matches the target, which is essentially the desired output. By calculating the error between the desired and given output, the model can then correct itself towards minimizing that error [11]. That is the basis on which most supervised learning models are built.

While regression problems focus on creating models for predicting numeric values, classification tasks target towards models that predict distinct classes, i.e. different blood types. So in that case, the targets will consist of the class that every data index belongs to. The more classes a model has, the more complicated its structure will be.

A specific and very useful machine learning algorithm is the neural network. Its application can vary, from medical diagnosis (as explored in this thesis) to image recognition.
1.2 **Neural Networks**

- **What are Artificial Neural Networks?**

While studying the capabilities of the human brain, a question of importance came to the surface: are machines capable of recreating the functions of the neural system of the brain? So, artificial neural networks (ANN) were created, structures built to emulate the neural system of living organisms.

To understand the way ANN work, one has to look no further than the biological neural networks. The neural system of organisms consists of many different neural networks, and each one of them has the neuron as its building block. The neuron is simply the smallest and independent unit of a network; it constantly processes information, receiving and sending electrical signals to other neurons. However, the similarities in structure do not extend for long, as a machine (a computer) cannot operate with the same ease as a human brain. For example, voice or image recognitions are done by the brain with higher success than a computer. Of course, the problem isn’t with speed. Computers can process information thousands times faster than a human can. The problems lie in the distinct differences in internal construction between the two and also on the fact that a computer relies heavily on predetermined rules. [1]

---

A simple model of an artificial neural network is shown above, in Figure 2. Neurons are represented as circles and receive a number of signals \( x \), which is the input of the network. The neuron has only one output, that depends on the input it receives. The network above has only two layers: the input...
layer and the output layer. However, in many cases there will be hidden layers in-between, allowing for a more complex computation. The numbers of neurons each hidden layer has, depends on the specific problem the network faces and is open for investigation.

➢ Signal propagation through a Neural Network

As it can be seen above, every signal that is transferred from one neuron to the other through the network is connected with a weight value \( w \), which indicates how tightly are two neurons connected [1]. Essentially, the weight shows exactly how important is the contribution of each signal to the structure of the network, for the two neurons it links together (the higher the weight, the more significant the contribution). After every input is multiplied with its corresponding weight at the neurons, they are all gathered at the neuron of the output layer and are summed. The sum is then applied as input to a transfer function, which produces the final output, as described in [1]. That signal then can be transferred to other neurons and act as their input, if more hidden layers were to follow. Neural networks with one or two hidden layers between the input and the output are called Shallow Neural Networks, and those with more, Deep or Convoluted Neural Networks.

The simplest transfer function is the binary one. Having received all of its input signals, the neuron then compares the sum \( x \) of those signals to a threshold value \( t \) that is set while initializing the network. If \( x \) is bigger than the threshold, the neuron is inactive and propagates no signal to the rest of the network. Otherwise, it offers that sum value as input to the next neuron. In that way, the neuron acts as a binary element, and therefore the transfer function has the form shown below:

\[
f(x) = \begin{cases} 
1, & x < t \\
0, & x \geq t 
\end{cases}
\]  

(1)

Most transfers functions are not binary though and as a result there is no threshold value involved. In many cases, the value \( x \) is computed using the neuron’s inputs and weights and a numerical value is given as output by the transfer function. For example:

\[
f(x) = \frac{1}{1 + e^{-x}}
\]  

(2)

The transfer above is called a sigmoid function and is the function that was used in the networks used in this thesis. It is obvious that functions as the sigmoid are non-linear by design, because in
neural networks there cannot be an output directly proportionate to the input, as is the case in the linear approach.

➢ Training an ANN using error backpropagation

Being a typical machine learning algorithm, neural networks have to be trained properly in order to be able to perform certain operations. The error backpropagation is the most popular method for training networks with one or more hidden layers, each one with multiple neurons. The central idea of the error backpropagation method is simple: a network begins the process of learning from random values of its weights. If the answer it produces is wrong, then the weights are corrected in a way that minimizes the error (the difference between the output and the target), aiming to make the error small enough to be tolerable [1]. That procedure is repeated several times until it is fair to declare that the network has learned the examples with the desired accuracy.

The training procedure can be summarized in 6 steps: [4]

- Taking an input instance and giving it to the network as input.
- Calculating the output by using the sigmoid transfer function.
- Forwarding each layer’s output to the next layer (input to hidden, hidden to output).
- Calculating the error at the final layer.
- Altering the weights, one by one, starting from the last layer towards the first, depending on the error value.
- Continuing to the next instance and repeating the same steps.

The way the value of the weights is changed depends on whether the neurons are linear or non-linear. Since only non-linear neurons are going to be used here, the backpropagation technique that will be explained refers to those networks only. It is assumed that the network is a feedforward one, which means that the information is passed from the input layer towards the output layer. The processing of the information is done layer by layer and neuron by neuron, where each neuron receives the output of the previous layer and gives to the next one the calculated value, according to the transfer function.

In table 1 there are the symbols that will be used for mathematically explaining the backpropagation error training for those types of networks.
Table 1: Symbols used for explaining backpropagation training and their meaning

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_{ij}$</td>
<td>The weight that connects neurons i and j</td>
</tr>
<tr>
<td>$\Delta_p w_{ij}$</td>
<td>The difference between the weights of neurons i and j after p instance of the input vector</td>
</tr>
<tr>
<td>$E_p$</td>
<td>The error (difference of input-output) of instance p of the input vector</td>
</tr>
<tr>
<td>$t_{pj}$</td>
<td>The target of neuron j for the instance p of the input vector</td>
</tr>
<tr>
<td>$o_{pj}$</td>
<td>The output of neuron j for the instance p of the input vector</td>
</tr>
<tr>
<td>$x_{pi}$</td>
<td>The input signal of neuron l for the instance p of the input vector</td>
</tr>
<tr>
<td>$\delta_{pj}$</td>
<td>The difference ($t_{pj} - o_{pj}$)</td>
</tr>
</tbody>
</table>

Mathematically, the desired result is for the derivative of the error as to every weight $w$ to be proportional to the variation of that weight value with negative constant of proportion. In the presence of hidden layers, the way the derivatives are calculated is not very obvious.

The training method can be summarized in three equations [1]. The first consists of the general Delta rule, a gradient descent learning rule which states that the $w$ in every layer is changed by a value that is proportionate to the error signal ($\delta_{pj}$) and also to the output $o_{pi}$:

$$\Delta_p w_{ij} = n \ast \delta_{pj} \ast o_{pj}$$ (3)

where $n$ is a constant. The other two equations determine the value of the error signal. For every neuron in the output layer, the error is calculated in (4):

$$\delta_{pj} = (t_{pj} - o_{pj}) \ast f'_j(S_{pj})$$ (4)

where $f'_j(S_{pj})$ is the first derivative of the transfer function. For the neurons in hidden layers, the mathematics are more complicated, but finally the error can be calculated by (5):

$$\delta_{pj} = f'_j(S_{pj}) \ast \sum_k \delta_{pk}w_{kj}$$ (5)

As a reminder:

$$S_{pj} = \sum_i w_{ji}o_{pi}$$ (6)
The equations (4), (5) and (6) construct a loop for the network training and can alter only one \( w \) value at a time, starting from the last layer. The system will repeat this process as many times as it needs for the neural network to be successfully trained.

### 1.3 The ECG Signal

As was mentioned in the introduction, electrocardiography involves placing electrodes upon the patient’s skin to record any electrical activity his heart produces over a period of time. The depolarization and repolarization of every heartbeat causes small but significant pattern of electrical changes that the electrodes have the ability to detect.

Most commonly, a 12-lead ECG is used where ten electrodes are placed on the patient's chest and limbs. The ten electrodes in a 12-lead ECG are listed in Table 2. Be aware that a lead is not the same as an electrode. Whereas an electrode is a conductive pad in contact with the body that makes an electrical circuit with the electrocardiograph, a lead is a connector to an electrode (multiple leads can share the same electrode). The measuring process of the heart muscle and its electrical activity is done from 12 different angles represented by each lead and then is recorded on a time frame of about 10 seconds. By applying this method, the entirety of the electrical depolarizing cycle of the heart can be captured at every single moment with information about direction and magnitude. The result of this non-invasive medical procedure is a diagram presenting the variation of the electrical voltage through time: the electrocardiogram, or ECG.

An indicator of a healthy condition of the heart is, during each heartbeat, is the progression of depolarization which begins with pacemaker cells located in the SA (sinoatrial node) and continues in order, by spreading through the atrium, passing through the atroioventricular node down into the bundle of His and into the Purkinje fibers, and finally spreading down and to the left throughout the
ventricles. The recognizable tracing of the ECG is born by this orderly process. Any physician then is in a position to extract information about the heart rate and rhythm as well as blood flow to the ventricles (indirectly) by analyzing the tracing. [2]

Table 2: Electrode placement on a 12-lead ECG

<table>
<thead>
<tr>
<th>Electrode Name</th>
<th>Electrode Placement</th>
</tr>
</thead>
<tbody>
<tr>
<td>RA</td>
<td>On the right arm, avoiding thick muscle</td>
</tr>
<tr>
<td>LA</td>
<td>In the same location as the RA electrode, but in the left arm</td>
</tr>
<tr>
<td>RL</td>
<td>On the right leg, lower end of medical aspect of calf muscle</td>
</tr>
<tr>
<td>LL</td>
<td>In the same location as the RL electrode, but in the left arm</td>
</tr>
<tr>
<td>V₁</td>
<td>In the fourth intercostal space (between ribs 4 and 5) just to the right of the sternum (breastbone)</td>
</tr>
<tr>
<td>V₂</td>
<td>In the fourth intercostal space (between ribs 4 and 5) just to the left of the sternum</td>
</tr>
<tr>
<td>V₃</td>
<td>Between leads V2 and V4</td>
</tr>
<tr>
<td>V₄</td>
<td>In the fifth intercostal space (between ribs 5 and 6) in the mid-clavicular line</td>
</tr>
<tr>
<td>V₅</td>
<td>Horizontally even with V4, in the left anterior axillary line</td>
</tr>
<tr>
<td>V₆</td>
<td>Horizontally even with V4 and V5 in the midaxillary line</td>
</tr>
</tbody>
</table>

For one to provide interpretation of the electrocardiogram, the key is to understand the electrical conduction system of the heart. Normal conduction starts and propagates in a predictable pattern, and deviation from this pattern can be a normal variation or be pathological. The cardiac output of each heart rhythm can variate from bad to good, a decision that relies on experience. As any medical imaging method, the ECG is useful for the assessment of the mechanical function of the heart, and like any other physiological test, determining a normal condition relies on studies of the population. Any heart rate range that is between sixty and one hundred can be considered normal since data shows this to be the usual resting heart rate. Interpreting the ECG comes down to recognize certain patterns and having a spherical understanding of the signals the heart muscle produces. The foundation of these patterns is in electromagnetism and can be summarized with these four points below: [15]

1. depolarization of the heart toward the positive electrode produces a positive deflection
2. depolarization of the heart away from the positive electrode produces a negative deflection
3. repolarization of the heart toward the positive electrode produces a negative deflection
4. repolarization of the heart away from the positive electrode produces a positive deflection
Therefore, following the overall direction of depolarization and repolarization leads to a vector indication a deflection, whether positive or negative, on the ECG, that depends on the lead it is pointing. For example, the process of depolarization from right to left leads to the production of a positive deflection in lead I, as the two vectors are pointing towards the same direction.

As can be seen in Figure 3, within a normal ECG waveform, four distinct parts emerge. Those are the P wave, a QRS complex, a T wave, and a U wave (not represented in Figure 2), all of them easily discrete and with separate patterns of appearance.

According to [2], [4], the P wave represents the depolarization which spreads through the atria coming from the SA node. Its usual duration is 80 to 100 milliseconds. Following the P wave, a time period appears of zero voltage (isoelectric period) which represents the impulse travelling in the atrioventricular node and the bundle of His. Measuring the time difference between two P waves leads to the estimation of the atrial rate. A useful time interval, that indicates the period from the beginning of the atrial depolarization to the beginning of the ventricular depolarization, is the P-R interval. Its range varies from 120 milliseconds to 200 milliseconds. In the ECG, it is the time period between the beginning of the P wave to the start of the QRS complex.

The QRS complex (can be seen clearly in Figure 3) represents ventricular depolarization and has a standard duration of 60 milliseconds to 100 milliseconds, which is a short period of time. That would point towards the depolarization being a quick process. If it lasts longer than the timeframe above, the conduction within the ventricles appears deteriorated. This occurs when a ventricular hearth (abnormal pacemaker site) becomes the pacemaker driving the ventricle, in which case the result is usually the conduction over slower ventricle of the impulse. Thus the QRS complex’s duration is prolonged, as the time period of the depolarization is bigger. Measuring the time interval between two consecutive QRS complexes gives an estimated value of the ventricular rate. Shape wise, the QRS complex doesn’t always look like the one on Figure 3, because it depends on the electrodes that are being used during the recording. Also, it depends on the appearance of abnormal conduction of electrical impulses in the ventricles. Following the QRS complex is a period of zero voltage, when the ventricle gets depolarized and is called the ST segment. Its importance is evident in diagnosing cases of hypoxia or ventricular ischemia, since then the ST segment appears elevated or depressed accordingly. [10]
The T wave represents ventricular repolarization. Concerning its duration, conduction of the depolarization wave is faster than wave of repolarization, therefore is longer. The time period between the onset of the QRS complex and the T wave is called the Q-T interval and is an estimation of how long a ventricular action potential lasts. In fact, it represents the time cycle of ventricular depolarization and repolarization. Depending on the heart rate, a Q-T interval usually lasts from 0.2 to 0.4 seconds. The U wave represents papillary muscle repolarization and is not typically seen, so its absence is generally ignored.

1.4 MIT-BIH Arrhythmia Database

As was mentioned above, neural networks need a large amount of data in order to be trained properly. Therefore, a database was needed that provided the learner with many ECG recordings and their corresponding annotations. That was the MIT-BIH Arrhythmia Database [5], which consists of 48 ECG recordings, each with a duration of 30 minutes and recorded with two channels. Those were obtained by the BIH Arrhythmia laboratory from 47 different subjects, out of which 25 were men (ages from 32 to 89) and 22 were women (ages 23 to 89). Out of a whole of 4000 24-hour recordings taken from a sample of both inpatients and outpatients (60% and 40% respectively), 23 were chosen at random and the other 25 were chosen from that same set in order for more significant but less common arrhythmias (like junctional, complex ventricular or supraventricular arrhythmias) to be included fairly in the small random sample. For the majority of the records (45), the upper signal that was used is a modified limb lead II or MLII, which required for the electrodes to be placed on the chest. On these recordings, the lower signal that was used was a modified lead V1, or in some cases V2, V4 or V5. For the other 3 ECG recordings, the upper signal was a lead V5 and the lower one MLII, V1 or V2. Note that in the upper signal a normal QRS complex appears more prominently. The recordings had to be digitized, because they were obtained in analog form. A frequency of 360 samples/second per channel was used for that process, along with eleven-bit resolution over a range of 10 millivolts. Also, the ECGs were digitally band pass filtered at 0.1-100 Hz in order to emphasize the QRS complexes.

On every record, cardiologists produced beat labels for every heart beat using a slope-sensitive QRS detector. The scientists added labels for beats missed by the detector, signal quality labels, rhythm labels and comments to go along with the “normal beat” labels, that were initially detected. Any
disagreements were discussed and resolved by two cardiologists, for the final transcript of annotations to be produced. Before their release, the annotations were checked for inconsistencies by an auditing program, so that any possible missed or falsely detected beats would be identified. After many corrections through the years, approximately 110,000 annotations emerged in computer-readable form. Generally, the annotations appear at the R peak of the QRS complex and their location is described sufficiently accurately in the reference annotation files. The symbols used in the annotations are very useful for the classification of the heart beats, so they are presented in Table 3 along with their meaning.

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

1.5 **Wavelets and Discrete Wavelet Transform**

Signal processing can be facilitated by using a class of functions, which are well localized in time and space, in order to capture the abrupt oscillations of a signal, like the ECG. Wavelets fulfil these criteria. A wavelet is a rapidly decaying waveform of effectively limited duration (unlike sinusoid waveforms that extend to infinity) that has an average value of zero. Where sinusoids are smooth
and predictable, wavelets tend to be irregular and asymmetric. [19] The two most fundamental wavelet concepts are scaling and shifting. Scaling refers to the process of stretching or compressing it in time, according to the scaling factor $s$ which effects the wave’s frequency. Scaling a wave by 2 ($s = 2$) results in reducing its frequency by half. A stretched wavelet helps in capturing the slowly varying changes of a signal, while a compressed wavelet captures the more abrupt oscillations. Therefore, with scaling, a signal is represented accurately through its whole length. Shifting on the other hand means delaying (or hastening) the wavelet’s onset. Mathematically, delaying a function $f(t)$ by $k$ is represented by $f(t - k)$. Properties like scaling and shifting are utilized for creating a wavelet transform. While in Fourier analysis a signal is broken down to waveforms of various frequencies, in similar fashion wavelet analysis consists of the deconstruction of a signal into scaled and shifted versions of the original (mother) wavelet called coefficients. Coefficients are functions of scale and position which, when gathered, depict the original signal. To create those functions, it is necessary to compare the wavelet with the signal, while using every available scale and shifting it along its entire length. The results of the comparison, which depict of how similarly the signal is approached constitute the wavelet coefficients. Plainly, the coefficients constitute the results of a regression of the original signal performed on the wavelets.

![Figure 4: Graphic example of a four-level Discrete Wavelet Transform](image)

The two major transforms in wavelet analysis are Continuous and Discrete Wavelet Analysis. Unlike the DWT, the Continuous Wavelet Transform operates at every scale, even up to a maximum scale that is determined by the user, by losing details in the analysis for computational power. Regarding shifting, the wavelet in CWT is shifted smoothly over the full length of the analyzed signal. However, calculating all the wavelet coefficients at every scale available is a long process and produces vast
amounts of data. So in this thesis, Discrete Wavelet Transform was chosen, where only a subset of scales and positions are used to calculate the coefficients. According to [18], if scales and positions are based on powers of two (dyadic values), then our analysis will have the same accuracy with increased effectivity.

To implement a Discrete Wavelet Transform on a chosen signal, filtering techniques are often used through algorithms that produce coefficients rapidly. The signal passes through a series of high-pass and low-pass filters to be analyzed into high frequency and low frequency content respectively. Downsampling is performed on the signal after its propagation through the filters, where the sampling rate of the signal is reduced (or half of the samples are discarded, according to Nyquist’s rule) and the information remains intact. The coefficients that result from the filtering and downsampling are divided into approximation and detailed: the approximation coefficients are the high-scale and low-frequency components of the signal, whereas the detailed coefficients are the high-frequency and low-scale components [2]. By iterating the decomposition process described above, the signal is broken down into many lower resolution components, as illustrated in Figure 4. The original signal is shown to be deconstructed to four levels of coefficients by applying a low pass filter to every approximation coefficient. The decomposition of the signal into different frequency bands is obtained by consecutively filtering the time domain signal through high pass and low pass filters. Practically, a suitable number of levels must be selected based on the nature of the signal, the purpose of its processing or on a suitable criterion. [19]
2. Network Creation for Detecting Normal Beats

2.1 Preparation and Algorithmic Work Flow

Before the graduate process of the conducted work is described, we need to mention some prerequisites that had to be done in advance. First of all, the majority of the study was implemented with Matlab, and specifically with two available toolboxes on that environment: the Neural Network Toolbox and the WFDB toolbox. As its name indicates, the Neural Network Toolbox [4] provides a useful GUI as well as an advanced script (which is available for modification of course) of many types of Networks, depending on the application. Those types were a Fitting Neural Network, a Pattern Recognition Neural Network, a Clustering Network and a Time-Series Network. Along with the Toolbox, a manual was provided to the learner in order to understand and assimilate the ways the Toolbox works in Matlab. Regarding the WFDB (waveform database) Toolbox [5], it consists of a series of functions for both reading, writing and most importantly processing signals that have been obtained from Physiobank's Database [5] (the database used here is the MIT-BIH Arrhythmia Database, also from Physionet). Basically, it provided the environment for the patients’ records stored in waveform to be processed easily in Matlab, along with the annotation files that accompanied each record. The functions that were used from the toolbox will be presented and explained later on. Finally, to end the preparation stage, we had to download the entire database locally, meaning that all the ECG signals, along with header files (notes from cardiologists) and annotation files had to be downloaded.

The whole process of extracting information from an electrocardiogram to end up classifying each heartbeat can be broken down to discrete stages. The work flow is demonstrated in Figure 5 below. In the next units, those steps will be explained in detail.

![Figure 5: Work flow for ECG analysis and Network creation](image)
2.2 **ECG Analysis and Creating the Vectors for the Network**

First, we had to obtain the ECG signals from the database and read them in a way that they can be processed by Matlab. The function that was used for that was `rdsamp()`, which takes the record of a patient as input and produces a vector with the ECG signal’s voltage values in specific time intervals, along with those time intervals on which the signal was sampled. As mentioned above, the sampling frequency during the digitization of the signal was 360 Hz and as a result the time intervals were $\frac{1}{360} = 2.778$ milliseconds apart. Also, since we would focus only on the upper lead signal of every ECG signal, the second channel would be discarded. To proceed, we had to isolate the heartbeats that were characterized with an annotation label, and therefore had an identity of what they represented. To do that, we read the annotation files using the function `rdann()`. Its inputs were both the WFDB record and the annotator and it returned the position of every labeled beat in the record with its corresponding symbol, and its meaning as well (as shown in Table 3). So at this point we have extracted all the information needed from the ECG signals.

Next, we would have to process the information from the signal and transform it into vectors that can be used by the neural network. But in order for that to take place, a decision must be made regarding the type of network that would be used. After much consideration, the pattern recognition net was utilized, as it offers the possibility to use targets (the clustering network makes associations using only the input data) and the ability to classify the data into specific and discrete classes (the fitting network would produce results in a way that they fit with a linear model). Once that was decided, we knew that the vectors a pattern recognition network would demand were an input vector and a target vector.

- The target vector was the easiest to create, because of the fact that the Matlab Neural Network Toolbox requires a very specific form of target matrixes. The number of classes we would aim to classify the data into, translated to the number of rows the vector contained. Every column represented a single heart beat that was annotated and in each column of the vector only one class was correct (or active), indicating the class that the heartbeat should be classified into. There lies the first approach that we considered: since the majority of the heartbeats in the database were labeled as “N”, meaning normal beat (about 84%), we decided for that to be the first available data class. The second class would contain every other heartbeat aiming to indicate a faulty condition. So in the first class all the beats would
be classified as normal, and in the second class as abnormal. By applying this approach, we were able to create a target vector for every WFDB record.

- The input matrix would consist of the signal values that were indicated by the positions of the annotation. Since every patient’s record was different, one input matrix would correspond to a single patient. However, in order to include the complete information about the pulses that were annotated, we needed to take into consideration the signal values around the position of the beat in the annotation. Therefore, at that point arose a necessity to create a parameter, a window size, that would define the distance on either side of the beat that would be included in order to not lose information about the heartbeat. For a specific window size, every column of the input matrix would consist of the signal around the annotated, and labeled, beat, and the number of columns corresponded to the amount of labeled heartbeats that a patient’s record contained. Another problem that was addressed was the possibility of samples from two beats to interconnect with each other. Pulses like that were discarded as they were too close to successive beats, especially when a large window size was applied.

In that way, both the input and the target vectors were ready to be used by the network. The way the target vector was created obviously indicates that we would create a neural network that could identify with accuracy the normal or healthy heart beats within the ECG signal.

2.3 Configuring the Neural Network and its parameters

Before training the network, all of its parameters would have to be determined. The only parameter concerning the data was the window size, as was discussed above. Altering the window size changed the dimensions of the input matrix. For example, if we select a window size of 16, we would keep 64 samples before and 64 samples after the annotated heartbeat and therefore the input vector would consist of 129 rows, which can be seen in Figure 6 (as a reminder, each column represents a different beat series). The annotation index characterizes the whole column of the matrix as one beat of specific behavior (i.e. normal beat).

Before initializing the network’s other parameters, it is necessary to divide the data. Part of a typical multilayered network flow is dividing the data into three subsets. The training set is used to update the weights of the neurons and calculate the gradient and its descent. The second is the validation
set, which is responsible for finalizing the values of weights and ending the training process. This occurs whenever the validation error reaches its minimum values. Monitoring the validation error may be proved useful for avoiding overfitting, as it increases once the network begins to overfit the data. Essentially the validation data set helps to fine-tune the network so that it works as efficiently as possible. The third and final subset is the test set, which is used to test the network after the training has finished.

As for the network parameters, the most important one concerning its structure was the amount of neurons of the hidden layer of the neural network (let’s call it hidden layer size). Generally, a large amount of neurons in the hidden layer allows for a higher level of computational process by the network, even though the duration of the computing process would increase. The main peril with increasing the hidden layer size is overfitting, meaning that the network might learn to only respond to its input data and simultaneously would lose the ability to generalize, if other data were presented. Consequently, an optimum amount of neurons would be required, which is a subject of exploration, as will be seen later on. In Figure 6 a network with 10 neurons in the hidden layer is demonstrated. The output layer has only two neurons, as there are only two classes available for classification.

![Diagram of a network with defined hidden layer size](image)

Figure 6: Overview of a network with defined hidden layer size

The other parameters that define a network are irrelevant to its structure and concern its functionality. These parameters are presented below, along with their selected value/name:

- **Training function**: The training function defines the way the network is trained and specifically how the weights of every layer are updated and corrected, as every indicated data point is being iterated through. The chosen function here is the scaled conjugate gradient backpropagation function (“trainscg”), whose memory requirements are relatively small, and yet is much faster than standard gradient descent algorithms.
Input and Output Processing Functions: Input processing functions can transform user input data to a more efficient form for a network. On the other hand, output processing functions are used to transform the target vectors the user provides for network use. Two functions were used here: “removeconstantrows” and “mapminmax”. The first removes any inputs or targets that are constant (and therefore are discarded) and the second normalizes the input and target elements to fall in the range of [-1, 1].

Data Dividing Function: The division of the data into the three subsets (training, validation and test) can be done in four different ways. Most commonly, the data is divided randomly by using the dividing function “dividerand”, which is the function that was used in this thesis.

Data Dividing Ratio: We need to determine the amount of data that would be categorized into each data set by defining the division parameters “trainRatio”, “valRatio” and “testRatio”. A 70% fraction of the data goes to the training set and 15% go to both the validation and the test set. These are also the default values of the parameters.

Performance Function: The performance function is clearly used to measure the network’s performance. The function that was used was “crossentropy”, which is appropriate for pattern recognition and classification problems and calculates a network’s performance with a measure of the cross-entropy of estimated and actual class memberships.

After all the weights were initialized and the parameters were set, the network was ready to be trained. The type of network used was a Pattern Recognition Network (Matlab’s patternnet function), for the reasons discussed in [14].

2.4 Training the Network and Creating Performance/Error Matrix

For the training of a single network, Matlab’s train() function was used and batch training was implemented, meaning that the weights were updated only after all the inputs were presented. Monitoring the training process is simplified by nntraintool, a GUI that provides the user with information about both the training and the network itself. The tool can be seen in Figure 8, where it is obvious that quantities like the epochs (number of times the network iterates through all the input data), training time or the plot of training state can be viewed. Regarding the ending of the training process, nntraintool offers four possible outcomes. The first is to stop the training manually (at the bottom of Figure 8), which is not recommended. Two of them include a variable reaching its
minimum value and these variables are the performance (minimum at zero, meaning no error between the outputs and the targets) and the gradient (default minimum value is set at $10^{-6}$) after the backpropagation error process. Last are the validation checks. If the validation error fails to increase in six consecutive iterations of the data, then the training is complete. Along with the progress part of the tool, the user is given 5 different plots to evaluate the training process along with the performance of the network and the classification it produced. The training state plot (Figure 7) gave the user a perspective of the descent of the gradient from all three data subsets and its minimum value, as well as the validation error reaching its minimum at a specific epoch. More information about the other plots will be explained in section 1.5.

It was evident that training one network would not be enough, considering the three different configuration variables each network had to process: the patient, the window size and the hidden layer size. Every patient of course had a different record and as a result, unique input and target vectors had to be created for each one. The window size would determine the size (the number of rows) of that input vector. However, there were no default values for neither the window size nor the hidden layer size. So, investigation was in order. We had to create a method that could determine which configuration of the network was the best, and repeat that process for every patient’s record. The variation of the parameters was selected as such: the window size would begin from 16, with a
step of 16 until 128 and the hidden layer size from 5 to 15 with a step of 1. It is understandable that for every configuration and patient corresponded one neural network. Out of the 48 records, 8 were deemed useless since they did not contain any normal beats and did not contribute to the statistical classification of the data. So in total we created and trained: \(40 \times 11 \times 8 = 3520\) neural networks! Understandably, that process took a very long time.

Aiming to find the optimum configuration of the neural network as mentioned above, we had to find a way to visualize how well each network performed. So we decided to measure the performance of the network generally and the performance of each data subset individually as well. The overall performance was easy to calculate with the Cross Entropy performance function, as mentioned above, with Matlab’s `perform()` function. In Cross Entropy, the function calculates a network performance (given targets and outputs), with optional performance weights and other parameters. It returns a result that heavily penalizes outputs that are extremely inaccurate, with very little penalty for fairly correct classifications. Minimizing cross-entropy leads to good classifiers and as a result, receiving minimum values gives as an estimation of good classification. To calculate the performance of the training, validation and test sets we would multiply the targets with a mask provided by the training structure for all three subsets, creating target data that corresponded to each set individually. Using those targets, the calculation of cross entropy was an easy task using `perform()`.

Along with the performance values, we calculated the error percentage of each network, which represented the percentage of the data that were put in the wrong class by the network. Error percentage was a useful measurement to calculate how well the networks executed the process that was assigned to them, because it is more easily understood as a value.

To gather all of those measurements together and be able to instantly recognize from which configuration they came from, we created a matrix (let’s call it final matrix) containing the record number (patient), the window size, the hidden layer size, the overall performance of the network, the training set performance, the validation set performance and the test set performance and finally the error percentage. The final matrix contained all of the information from the networks and their configuration so it was in a perfect form to numerically evaluate the classifications.
2.5 Results and Improvements

Classification Results

Evaluating a network’s classification can be done easily by exploiting the plots that are provided during the training process. The most useful plot was the Confusion Plot (plotconfusion), which shows specifically the amount of data points that were classified correctly or incorrectly, and on which class they were put in. One of the confusion plots that were obtained during the training of all the networks (using record 201, window size 64 and 10 neurons in the hidden layer) is shown in Figure 9.

Before presenting and discussing the results extracted from the final matrix and training all the necessary networks, the matrices in Figure 9 have to be explained. A Confusion Plot is essentially a matrix representing the targets (true labels) and the network outputs (predicted labels). Plot confusion shows four different matrices, one for each data subset and one for the overall data set. In a classification problem of n classes, the confusion matrices will have \((n + 1)\) rows, representing the output class and \((n + 1)\) columns which correspond to the target class. Every diagonal cell
indicates correctly classified data and the off-diagonal cells indicate incorrect classification, while every cell shows the percentage as well as the number of observations. More specifically, the cells of the far right column represent the percentage of all observations that have been predicted correctly or incorrectly to belong to each class. For example, in the All Confusion Matrix of Figure 9, the upper right cell indicates that 96.9% of all the beats that were classified as normal were in fact normal beats (1620 out of 1672) and 3.1% of all the beats that were classified as normal were actually abnormal (52 out of 1672). Cells of the bottom row show the percentages of all observations that belong to each class and were correctly or incorrectly classified. For example, the bottom left cell of the same matrix shows that out of 1624 beats that were normal, 99.8% (1620 beats) were correctly classified as normal and 0.2% (4 beats) were incorrectly classified as abnormal. The bottom right cell indicates the overall accuracy of the classification, along with the overall error percentage that was saved in the final matrix. Given those details, it is understandable that the confusion matrices give the user an excellent tool to check the accuracy of the classifications on each individual data set.

Concerning our work, the vast majority of networks that were trained produced Confusion Plots similar to those on Figure 9. The classification on the first Target Class (normal beats) was exceptional, never dropping below 98% of all examples. That result was produced for even the smallest number of neurons in the hidden layer, which was 5, and of course continued with the bigger layers as those networks had even more computational power. The most probable explanation for this result is that the majority of the labeled beats from WFDB records are normal beats, and therefore the network has a large dataset to train and test itself on those particular beats. Since it is more accustomed to that class, it would be expected that it performs classifications very well. Concerning the second class, the one of abnormal beats, it consists of a much smaller amount of heart beats and therefore, one would expect that incorrect classification would be more prominent there. Despite, the overall well-functioning networks, some irregularities occurred. A common occurrence during the training process was that many abnormal beats would be predicted to be normal by the network. This phenomenon intensified on networks with small hidden layer sizes, with the overall percentage of correct classifications in the second class dropping sometimes to 10%. In some instances, all of the data were put in the first class and none were classified as abnormal beats, as we saw on the confusion plot. After some investigation, we concluded that it was caused by the extremely low number of abnormal beats on a patient’s record. It is expected
from a network to perform poor classification on two different classes when one of them has little to no data that belong to it. One point that needs to be made is that classification accuracy is directly impacted by the data size of the second class. So the classification accuracy would vary from record to record, depending on how many beats were labeled as abnormal. Other than these irregularities, it is safe to say that the networks worked very well on classifying the beats correctly, reaching prediction accuracy of 98% in some configurations for the abnormal beat class and generally achieving about 97% of total accuracy.

These results can also be verified be the ROC Plot (Receiver Operating Characteristic), the other useful plot that *nntraintool* offers for classification problems (when using *patternnet* in general). A ROC plot is shown below in Figure 10, for which the record Number 203 was used, window size was 80 and hidden layer size was 7.

![Figure 10: Example of a ROC plot from a trained network](image)

The Receiver Operating Characteristic is used to check the quality of classifiers. For each class of a classifier, ROC applies threshold values across the interval [0,1] to outputs and for each one, the True Positive Ratio (TPR) and the False Positive Ratio (FPR) are calculated. For a particular class 1, TPR is the number of outputs whose actual and predicted class is class 1, divided by the number of
outputs whose predicted class is 1. FPR is the number of outputs whose actual class is not class 1, but the predicted class is, divided by the number of outputs whose predicted class is not class 1. You can visualize the results of this function with `plotroc`. The more each curve hugs the left and top edges of the plot, the better the classification. Figure 10 gives a great representation of what we saw during the training process of the networks. Both curves fit very well into the top left corners, despite of the fact that the curve that corresponds to class 2 reaches a flat state in a smaller TPR. That occurs due to the larger amount of data points, or heart beats that belong to the first class, since the majority of incorrect classifications belong to the abnormal beat class. Most ROC plots had an even better curve shape wise than those in Figure 10, reaching the point of being almost tangential with the axis. This proves and verifies how well the networks classified all the data.

- **Performance Results and Optimum Configuration**

![Overall Performance](image)

*Figure 11: Example of non-magnified plot of the overall performance with window size set at 16*

Finally, we had to check the performance and error numbers that were gathered in the final matrix, numbers that are the most important results that were extracted from the training process of the networks. It should be noted that both of those measurement represent equally the overall
performance of the networks on classifying the beats with the desired accuracy. Firstly, we will illustrate the performance results.

As a reminder, the performance values (for the network in whole as well as for every data subset) were produced by the `perform` function, according to the parameter `net.performFcn`, which was set as Cross Entropy. Since for each different network configuration, a discrete measurement number was given, we decided to plot those measurements in relation to the variations of the window size and the neurons of the hidden layer. A minimum point at these plots would indicate a configuration that leads to a lower performance number, meaning a smaller difference between the outputs and the targets as calculated by cross entropy.

In a first attempt to plot the network performance, we discovered the graph shown in Figure 11. It is obvious that the chaotic nature of this plot could not be analysed as seen. Few instances presented high performance values, but the vast majority were densely concentrated in the bottom part of the figure. So, in our effort to find a minimum and reach to a possible conclusion, we magnified the bottom region, taking a different plot for every window size and studying the variations of the hidden layer size. Plots like these are presented in Figures 12 (overall performance), 13, 14 and 15 (training, validation and test set respectively).

![Figure 12: Plots of network overall performance with set window sizes and varying hidden layer size](image-url)
It is visible that we have magnified these plots within the range of $10^{-6}$ and still there are no easily recognizable minimum points between them. When minima appear, they do so randomly, without consistency on the hidden layer size, from a window size value to the next and are sporadic. Same phenomena can be seen in the plots for every data set in Figures 13, 14 and 15. So we can easily
deduce that there is no effect of the window size on the performance of the network, with the way it was configured. Later we will tackle that issue by changing the way beats are presented to the network. All the performance values are too close to each other which means the network is unable to differentiate from one heart beat to another, even if it was trained to classify them correctly.

It was remarkable that the network produced almost identical numbers of validation performance and test performance (~0.29) for all network configurations. Again no minima have the consistent nature that we are searching for. That same result can be depicted in Figure 16, when the error percentage is taken as a measurement of the network performance. Identically as Figure 11, the error percentage did not provide us with any information concerning the minimum error for a specific amount of neurons. Each curve represents a different window size, but all of them reach their minimum at the same level (0% error).
Similar results were produced by studying the variation of window sizes with a set value to the hidden layer size (Figures 18-21). Beginning with the error percentage, the same chaotic plot as before was produced, as shown in Figure 17. Every different curve represents a specific number of neurons, and their curves are studied with varying numbers of window sizes. All of them reach a 0% error percentage, giving us no information about their minima. Concerning the performance values, even at a magnification of $10^{-5}$ of the original value (the original figure had the same incomprehensible form as Figure 11), we still were not able to distinguish clear minimum points in Figure 18, that would indicate an optimum configuration for the network, this time from the perspective of a varying hidden layer size. Those results are mirrored in the training, validation and test set performances as well.
Figure 17: Plot of error percentages for different hidden layer size and varying window size (x axis)

Figure 18: Plots of network overall performance with set hidden layer sizes and varying window size (x axis)
Figure 19: Plots of network training set performance with set hidden layer sizes and varying window size (x axis)

Figure 20: Plots of network validation set performance with set hidden layer sizes and varying window size (x axis)
To summarize, we saw that even though the classifications conducted by the network were excellent in accuracy, as was evident by the confusion and roc plots, the network performance and error percentage values remained almost intact while the configuration was changed. It was expected that larger window sizes would improve the information about the heartbeat and would result into better performance than on smaller values. Correspondingly, networks with smaller hidden layer were expected to produce poorer results, as they did not have the computational force to classify the beats as well as they actually did. Those indicated a faulty behavior of the network, in the sense that the heart beat was not captured in the input vector the way we wanted. But, in order to be sure of the fault in our model, we had to test it in extreme situations, before changing it completely.

2.6 Experimenting with Larger Input Data

Admittedly, there were some significant problems with our previous model of networks, especially with network performance. There was a need to see if the surprisingly fixed values of the
performance for the network generally and the data subsets would variate, if extreme conditions were to be applied.

Firstly, we needed to minimize the risk of the network overfitting the training data and be unable to generalize results for data different than those it was trained with. The easiest way to achieve that is to increase the amount of data that is inputted to the network. By doing so, we provide the network with more examples, which allows it to be trained better and more accurately. In addition, we could check again for an optimal configuration for the network, since the smaller datasets did not seem to produce good results.

Table 4: Different categories of heart beats in the data set and the amount of each one.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Amount of Beats</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;</td>
<td>436</td>
</tr>
<tr>
<td>+</td>
<td>1188</td>
</tr>
<tr>
<td>/</td>
<td>4946</td>
</tr>
<tr>
<td>A</td>
<td>959</td>
</tr>
<tr>
<td>E</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>794</td>
</tr>
<tr>
<td>J</td>
<td>54</td>
</tr>
<tr>
<td>N</td>
<td>75003</td>
</tr>
<tr>
<td>Q</td>
<td>31</td>
</tr>
<tr>
<td>R</td>
<td>3078</td>
</tr>
<tr>
<td>S</td>
<td>2</td>
</tr>
<tr>
<td>V</td>
<td>6607</td>
</tr>
<tr>
<td>a</td>
<td>150</td>
</tr>
<tr>
<td>e</td>
<td>16</td>
</tr>
<tr>
<td>f</td>
<td>982</td>
</tr>
<tr>
<td>j</td>
<td>223</td>
</tr>
<tr>
<td>x</td>
<td>183</td>
</tr>
<tr>
<td>l</td>
<td>125</td>
</tr>
<tr>
<td>~</td>
<td>534</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>95312</strong></td>
</tr>
</tbody>
</table>

The method that was followed demanded an input vector that contained all heart beats, along with their annotation labels, from every single record of the database. In order to adopt this method, we had to accept the assumption that a heartbeat extracted from the ECG of a patient does not differ from a similarly labeled heartbeat of a different patient. Consequently, the input vector was treated as homogeneous regarding its data. Of course, the records that did not contain any normal beats were discarded from the process, as was discussed above. The input vector was created in the same
way as in the previous model. Table 4 was created to illustrate the exact amount of heart beats per annotation label in the larger data set. In that way, we can visually inspect how much data would each class off the classification process contain.

Concerning the two classes however, we approached the problem in two different ways. The first was to use the same classes as before, one consisting of all normal beats (“N”) and the other every other beat in the data, categorized as abnormal. In the second method, we changed the contents of the second class to solely atrial premature beats (“A” annotation label, Table 3). Beats that did not belong in either of those classes were deemed irrelevant and were subtracted from both the input and the target vectors.

To initialize and prepare the network for training, we kept the parameters concerning its functionality unchanged. However, the two structural parameters were processed in a different way. In the spirit of exploration, we gave the window size parameter its minimum (16) and maximum value (128), as well as an intermediate value (64) and constructed a differently sized input vector for each one. Logically, a corresponding target vector was created using the same logic. This process was repeated for both approaches of the problem, creating a specific duo of input/target for a network to process. It is noticeable that for this exploration, only a single network was necessary to produce results, while before we utilized multiple networks. To ensure the validity of the results, each training process was repeated several times with the same duo of vectors.

The extreme conditions that were mentioned concern the hidden layer size. We experimented with a wide range of neurons, from a single-neuron-layer to a layer with 100 neurons, for every case. Small numbers of neuron in the hidden layer are expected to produce poor classification results, especially when the network has to deal with one or two neurons in the hidden layer. In cases with vastly large hidden layers, you might cause the problem to be under-characterized and the network must optimize more parameters than there are data vectors to constrain these parameters. So we expected the performance numbers that appeared previously to alter dramatically, according to the extreme structure of the network, even though the window size was constricted to its previous limits.

Finally, we checked again to see if there would appear a network configuration (window size and hidden layer size) that would result in a minimum of error percentage, now that a much larger data set was presented to the network as input.
2.7 Results and Difficulties Exploration

➢ Results from the First Method

We tested every combination of input/target vector (with the three window size values that were mentioned before) with number of neurons in the hidden layer. The results however were not the expected, because the networks managed to classify very well the data and produce excellent classification statistics and plots, as can be seen on Figure 22.

The confusion and roc plots that appear in the figure were extracted using input and target vectors of window size 128 and 100 neurons in the hidden layer. We can see that the network classified the data with 97.6 % accuracy (as shown in the confusion plot on the left), containing the error percentage to a small 2.4 % misclassified heart beats. The ROC plot shows that our classifier performed very well on putting the data to the correct class, for both classes, since both curves come close to the upper left corner. It is evident that nothing has changed effectively from the previous model, except from the quantity of the data. Wrong classifications are constricted to a small percentage and normally appear on the second class, as the abnormal beats represent a smaller number of the dataset. Looking at the performance numbers, they did not variate from their previous values, which was the best indicator for the failure of our model. The overall performance value was around 0.007 — 0.008 (very small deviations appeared on all values), the training value
was 0.09 — 0.1 and both validation and test performance values were around 0.29, exactly as they were before. The overall classification of heart beats would variate from 87% (mainly for small hidden layer sizes) to 98%, even in extreme configurations. The lowest scores correlated with low prediction accuracy for the abnormal beats and variated from 65% to 89% in some cases. Those kind of results are an accurate representation of what we saw during the exploration and after training each network, during the whole process.

Those surprisingly steady results also appeared in our effort to find the network with the minimum error percentage of the data classification, when changing the window size and hidden layer size. In Figure 23, the curves represent different amount of neurons in the hidden layers. We can see the curves slightly declining as the window size increases, but not in convincing magnitude. The curves seem to have a horizontal behavior that shows the non-variating error percentage between different networks. It is evident that the very steady values of the network performance are mirrored in the error percentage, when variating the window size. All of the networks seem to have an error percentage of about 4%, without that changing when increasing the window size.

![Figure 23: Error percentage with variating window size for a specific amount of neurons in the hidden layer](image.png)
Because we wanted to be more persistent, we checked for an absolute minimum on all configurations of the network, by designing Figure 24. In the x axis we put the error percentage, in the y axis the hidden layer size and in z the window size. We can obviously have a better overview of all the networks’ performances by examining them in three dimensions. In this case, we can see that networks with lower error percentages seem to correspond to higher window size values, as the data points incline to the upper left side of the graph. However, they are not spreaded well; all the networks appear to produce error percentages of around 5%, despite their configuration. In our search for minimum values, we see that some appear but they are not as clear-cut and separated as we wanted in order to reach to a specific conclusion. However, it appears that increasing the size of the input data was a step towards the right direction. The error percentage seems to “follow” the increase of the window size in a very subtle but distinctable manner, by producing lower error percentages, meaning networks with better classification accuracy.

![Error Percentage for Multiple Configurations](image)

*Figure 24: Error Percentage for every possible network configuration*
Results from the Second Method

For the second method, since every heart beat that was not normal nor atrial premature, the size of both the input and the target vectors was smaller than before. However, only a very small percentage of the total heart beats are actually labeled as “atrial premature beats” (1.01%) as can be seen in Table 4, so as an improvement method it is expected to produce fewer good classifications. The data were tested for all three window sizes (16, 64 and 128) and a variety of hidden layer size values, from 1 neuron to 50 and 100. The results that were produced were not different at all from those of our previous efforts. Even after lots of testing and parameter changing, the performance outputs had their standard values of 0.008 for the overall performance and 0.01, 0.29 and 0.29 for each data subset. The only difference between this method and the first was that at random moments the network would not classify any beats as abnormal (class 2), giving an output of only normal beats. Of course that is explained by the significantly less amount of “A”-labeled beats in comparison the beats of class 1. Generally, a classification accuracy of 98% was achieved, which however was the result of correctly predicted normal beats and was not a valid indication of good performance by the network. The abnormal beats class was classified with an accuracy of 45% or below.

Discussion for improvements

Taking into consideration all those results, it is obvious that our methods did not lead us to unambiguous results. On the contrary, whether we used one patient’s heart beats or all the patients’ data as network input, we did not reach a conclusive minimum error, that would point towards an optimum network configuration. It should be noted that using a larger data set, with all the patients’ information proved to be a step into the right direction. Also when we experimented with extreme numbers of neurons in the hidden layer, the network did not seem to alter its performance. Those indicate that there was a fault in our logical approach of the problem: we treated the heart beats as nothing more than raw data. We need to reconsider the way a heartbeat is presented to the network, aiming for each beat to have a unique identity, that will be transferred to the output of the network. In order to achieve such separation and attribute discrete characteristics to heartbeats, feature extraction on every pulse needs to be implemented.
3. Feature Extraction

3.1 Algorithmic Work Flow

For the continuation of this thesis, inserting a feature extraction unit was deemed necessary. The feature extraction unit would receive the raw input data and produce the appropriate vectors which contain relevant information about each heartbeat. Those vectors would exist as inputs to the network, along with the corresponding targets, and would be used during the training procedure to classify the data into the two classes. As a result of the insertion of the feature extraction unit, the algorithmic work flow had to be changed (Figure 25). The rest of the steps remained unchanged, as it was the input data that caused irregularities.

Figure 25: Updated Algorithmic Flow with Feature Extraction Unit

3.2 Applying Discrete Wavelet Transform and Extracting Coefficients

The feature extraction method we used was Discrete Wavelet Transform (DWT), as was proposed by Emran et al. [23]. In that way, the data samples that compose the ECG signal would be compressed into a few feature vectors (which characterize the behavior of the signal), as a result of the spectral analysis that DWT performs. Because of its great time and frequency localization ability, the DWT can reveal the local characteristics of the input signal [10].

The wavelet transform separates signal components into different frequency bands enabling a sparser representation of the signal [20]. More advantages of the DWT were mentioned in the first chapter.
Before applying the DWT on the data, we had to decide on the type of wavelet we would implement, as well as the maximum level the signal would be decomposed. As a result of the research from Azariadi et al. [2] the wavelet that was chosen was from the Daubechies wavelets and the 2\textsuperscript{nd} order. The signal was decomposed to four levels. Daubechies wavelets in general are compactly supported wavelets with extremal phase and highest number of vanishing moments for a given support width. Associated scaling filters are minimum-phase filters. The 2\textsuperscript{nd} order of those wavelets (we will refer to it as “db2”) resembles slightly the QRS complex of an ECG signal, offering a better representation of the signal. The db2 wavelet can be seen in Figure 26.

The input data of the feature extraction unit contained all the heart beats from every patient in the database. This method seemed to produce better results earlier. So, the DWT had to be performed to every single of those 95,000 beats individually, and extract the detail and approximation coefficients from each one. Since we used four levels of decomposition, 8 different coefficients were created for every time the wavelet analysis was executed. Primarily, we used Matlab’s \texttt{wavedec} function to perform the one-dimensional analysis on the signal and returned a structure that contained the decomposition information of the signal. Using \texttt{detcoef} and \texttt{appcoef} we extracted the detail and approximation coefficients from that structure, respectively. The coefficients contained all the features of the original signal, meaning the compressed characteristics of each heartbeat. In that way, the data could be transferred to the data intact, without losing their nature.

3.3 \textit{Creating the Network – Configuration Exploration}

Before setting the network parameters and functions, we had to determine the form of the input data. Primarily, we had to decide which combination of coefficients would serve as input to the network, since four detail and four approximation coefficients were produced after DWT was applied. According to [2], using only the approximation of the fourth level as input would provide us with great classification accuracy and minimal computational cost at the same time. So out of the eight vectors from the spectral analysis of the original signal, only the 4\textsuperscript{th} approximation would constitute the input vector of the network, along with its corresponding target vector. The target was created in the same way as before.

Of course, we had to investigate the possibility of finding a network configuration with the minimal error percentage, with specific values for the window size and the hidden layer size. So we created
different networks, one for each pair of those parameters. To do that we had to create different sets of coefficients for every window size and use them as input to networks with various amount of neurons in the hidden layer (5-15). In Table 5 we demonstrated the number of samples of each level of coefficients, depending on the window size of the original data, before the feature extraction. The amount of elements the fourth approximation coefficient had every time can be found in the last column.

*Table 5: Number of samples per coefficient for different window sizes*

<table>
<thead>
<tr>
<th>Window Size</th>
<th>Samples of 1st level Coefficients</th>
<th>Samples of 2nd level Coefficients</th>
<th>Samples of 3rd level Coefficients</th>
<th>Samples of 4th level Coefficients</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>18</td>
<td>10</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>32</td>
<td>34</td>
<td>18</td>
<td>10</td>
<td>6</td>
</tr>
<tr>
<td>48</td>
<td>50</td>
<td>26</td>
<td>14</td>
<td>8</td>
</tr>
<tr>
<td>64</td>
<td>66</td>
<td>34</td>
<td>18</td>
<td>10</td>
</tr>
<tr>
<td>80</td>
<td>82</td>
<td>42</td>
<td>22</td>
<td>12</td>
</tr>
<tr>
<td>96</td>
<td>98</td>
<td>50</td>
<td>26</td>
<td>14</td>
</tr>
<tr>
<td>112</td>
<td>114</td>
<td>58</td>
<td>30</td>
<td>16</td>
</tr>
<tr>
<td>128</td>
<td>130</td>
<td>66</td>
<td>34</td>
<td>18</td>
</tr>
</tbody>
</table>

The parameters of the network had to remain unchanged of course, so that any result could be comparable to our previous method. All of the functions mentioned in the last chapter, were set to the same values here as well. Because both the performance values and the error percentage depict the same result, how well the network performed the data classification, we used only the latter in this scenario. Error percentages offer an easily understandable quantity of the amount of data that the network classified in the wrong class and therefore of its overall accuracy. Consequently, we created a matrix that contained the error percentage of a network with a specific configuration. The contents of this matrix summarized the accuracy of every network that was trained during this procedure.

### 3.4 Results

As mentioned above, the results concerned the error percentages from all the networks after the training process. The classifications that were performed had very good to excellent accuracy. More specifically, in cases of small window sizes and hidden layer sizes, the networks achieved
percentages around 12%. Noticeably, networks with the lowest window size (16) could not classify the better than 89% (11% error). Networks with higher window sizes achieved gradually lower error percentages, dropping to around 8% for 32 and 48 window sizes, and reaching percentages of about 5% for bigger values. That behavior is clearly represented in Figure 27, where each curve represents a specific window size. The gradual decline of the error percentages is evident, something that indicates a more natural response to the data: as the window size increases, the probability that the pulse would be contained within that region of the signal increase, and the percentage of misclassified data decreases. For that reason, every curve appears lower in the graph (smaller error percentage) than the curve before. Notice that the curve corresponding to window size 128 produces the lower error percentages out of all.

![Error Percentage with Wavelets](image)

Figure 27: Error Percentage in relation to hidden layer size for a variety of window size values

If we alter the figure above by switching the constant variable from the window size to the hidden layer size, we get Figure 28. With this particular graph we are starting to see what we aimed for at the beginning. All of the curves start from higher error percentages and gradually descent towards lower values in the y axis. Those lower values represent larger window sizes, as was seen and in the
previous figure. Obviously, some peaks appear that indicate lower classification accuracy, however they do not affect the overall flow of the graph and can be treated as noise. Additionally, we were interested in a minimum error percentage and not a maximum. That minimum point seems to be located in the higher hidden layer sizes, on curves for 12-15 neurons. To summarize, the behavior of our networks appears to be as predicted, since lower error percentages correspond to curves with the highest values of window size.

![Error Percentage per Window size with Wavelets](image)

*Figure 28: Error percentage in relation to window size for various values of the hidden layer size*

In order to clarify the existence (or non-existence) of an absolute minimum error percentage, and therefore optimal configuration of a network, we created Figure 28. Similar to what we did before applying the discrete wavelet transform on the larger data set, we plotted the error percentage in relation to Window Size and Hidden Layer Size in three dimensions in Figure 29, for all possible configuration, thus creating a more detailed view of each configuration individually.

The conclusions from this figure are very clear and very encouraging. First of all, the way the data is represented has majorly improved from Figure 24: there is a clear tendency for networks with larger
window sizes (and less evidently, with larger hidden layer sizes) to produce smaller error percentages and the data are more scattered, according to the configuration of the network they were produced. The graph has around 8 points of high error percentage at its right side, which correspond mostly to low window size values and some higher misclassifications in larger window sizes (can be treated as noise). At the other side of the figure, appears an absolute minimum at the upper left (illustrated with a red cross). That network produced 4.027% of error and corresponds to 14 neurons with a window size of 128.

![Error Percentage for Multiple Configurations with Wavelets](image)

*Figure 29: Error percentage for all possible network configurations*

That configuration is the one we were searching for, which produced the higher accuracy when classifying the data into the two classes. Its appearance clearly states that using feature extraction with wavelets on the data was the correct decision. Noticeably, in Figure 29 there are five other local minima that correspond to networks with error percentages below 4.9%. Those pairs of values for window size and hidden layer size are [128,6], [128,7], [96,14], [112,12] and [96,10]. Values like
these validate the results from Figures 27 and 28, where it was shown that networks with bigger window sizes produces more accurate classifications.

Now that we pinpointed the optimal network from the process above, we need to test it for smaller data samples and generally check its behavior when facing unknown patients.
4. Testing the Network and Emulating its Function in C

4.1 Retraining the network with unknown patients

The next step after pinpointing the optimal configuration for a network is to test if it responds well to data from unknown patients after having been trained. In order to do that, we separated the original dataset into two groups, while preserving the properties of the original data set regarding its contents of normal and abnormal beats. Our aim was to check how well a pretrained network would respond when trained again with a smaller data sample. The steps for this procedure are explained below in detail.

- **Creating the data sets:** Two networks had to be created for the purpose of the testing procedure. Therefore, we had to divide our dataset, which contained 95312 heartbeats, into two separate data sets. Each one of those sets had to include the same percentage of normal and abnormal beats as the original, preserving in that way the initial properties about the two classes. Originally, the data set consisted of 78.7 % of normal beats and 21.3 % of abnormal beats. Those same percentages carried over to the two datasets with the only condition that the smaller dataset would consist of 10 % of the information of the bigger dataset. So the large dataset contained 86648 heartbeats and the small one 8664 heartbeats, each one with 78.7 % of normal and 21.3 % of abnormal beats. To put it into perspective, the smaller dataset contained the amount of information that would correspond to ECG signals from three patients.

- **Training a network with a larger data set:** Firstly, we had to train a network using the configuration that produced the best classification accuracy from before: 14 neurons in the hidden layer and 128 window size. Of course, discrete wavelet transform was applied to the data set (the larger as well as the smaller one). We used the training parameters that were used for all the previous training procedures. After the training was completed, we took the error percentage as measurement for the performance of the network. The network produced an error percentage of 5.52 %, an excellent result for classifying the data. The fact that we did not reach the 4.027 % of error that we saw before is not surprising, since our large data set contains significantly less information than the original dataset.

- **Retraining the network with the small data set:** Now our network is trained and configured to respond well to the large dataset. So we wanted to check its response to a smaller data set during
training. The smaller data set contained 6818 normal and 1846 abnormal beats, mirroring the content of both the original and the larger data set. Those heartbeats were used to retrain the network along with the corresponding targets. The error percentage that we received was 0.53%, a number incredibly close to zero (100% accuracy), and around 10% less than the error the larger data set produced.

The conclusions that arise from this testing procedure is that our network is very well tuned with the configuration that we applied and responds excellently to unknown data from unknown patients, even if they represent as little as 10% of the information that was used to train the network originally. Since the network was trained the first time, the weights and biases were set to specific values that produce optimum results. When the input data changed by quantity, the alterations the network had to do to those weights during its second training were very small, since the nature of the problem was not changed. As a result, the network quickly was modified to respond very well to the new data and could easily classify the new heart beats with high accuracy.

4.2 Checking the networks behavior with single patient data

Now that we knew the network could respond very well to unknown data as long as it has been previously trained, we wanted to check its behavior when facing small training data sets. The main idea was to create a vector with all the heart beats of a single patient (similarly to what we did in the second chapter), then apply discrete wavelet transform and produce the feature vector which then would be the input data and would be used for training along with the target vector. However, when adjusting the training parameters of the network we would increase gradually the amount of data that would be used for training, and consequently the data for validation and testing. The iterations begin with 2% of the patient’s data acting as the training set and end at 98%. Also, the validation and testing set, as they represent equal percentages of the data, went from 49% to 1% through the iterations. For every iteration we would train a network with the available data and record the error percentages they produced. Regarding the patients themselves, since they represent a much smaller data set than what was previously used with feature extraction, we had to select those with an acceptable ratio of abnormal to normal beats in order for the classification to be performed with two classes of substantial data. The threshold that was set to that ratio was 10% and as a result only 18 out of the 40 patients were used. The results were plotted for every
patient; four of these plots appear below at Figure 30. We were interested in observing how the error percentage would behave when the amount of data that were used for training increased. To illustrate that behavior, we applied a polynomial fit of first degree (linear) to the plot, by using Matlab’s `polyfit()` and `polyval()` functions. The correlation coefficient $R^2$ was extracted with the function `corrcoef()` between the curve and the fitted line. Also, the equation of the linear fit is presented so that we have a clear method of knowing the slope of the line. The four patients that appear in Figure 30 were chosen randomly as a sample of all the other similar graphs.

![Figure 30: Error Percentage for increasing percentage of training data for four different patients](image)

The most important result from the figure is that the error percentage decreases as the percentage of data used for training increases gradually, as was logically expected. The network has more and more information to train with and is able to learn from more examples how to classify the
heartbeats. That way the amount of heart beats that would be classified in the wrong class was expected to decrease, as it did. This gives the reassurance that our network works according to plan. One would expect that networks with 10% available data for training would not be trained properly and many irregularities would appear, especially overfitting. However, those networks had a large amount of data that were used for validation and managed more easily to find the optimum model for the network that could learn the process better. It should be noticed that all of the patients that were used for testing showed this behavior. The peaks that appear in the graphs are not reason for concern, as they are nothing more than noise. For further confirmation we decided to plot the performance (calculated using the cross entropy performance function) for each subset individually (training, validation and test) for every patient. The results appear in Figure 31 for patient 223, which was chosen randomly.

Figure 31: Performance of each data subset with increasing percentage of training data (patient 223)
The graphs above confirm that our method of testing was correct, and that indeed the network learned to classify the data better while the percentage of training data increased. The upper left graph depicts the decline of the training set performance, meaning that according to cross entropy the data were used more efficiently for classification. Of course, since the validation and test data sets became smaller while the training set increased in size, the performance measurement for both of them is shown to increase, indication of larger errors.

This experimenting procedure would conclude the testing and checking of the performance and training of neural networks.

### 4.3 Implementation in a C script

After establishing that the network we configured and trained works with excellent accuracy and responds very well to new data, the next step was to export its structure and build it into a program using code in C, which would emulate its functions. This program could be the software of a possible processing unit that could do the same as our network, meaning classify the heart beats into the two classes. That could be achieved since the weights and biases were set to specific values and the network was finely tuned after the training procedure.

The first step was to obtain the values of weights and biases of the network we had trained (14 neurons in the hidden layer, 128 window size). First we received a vector containing information about every neuron, either in the hidden or in the output layer, by using Matlab’s `getwb()` function. We then proceeded to separate the weights and biases form this vector and organize them in weight arrays and bias vectors according to the level they represented. Matlab’s `separatewb()` was used for this action. The first weight array (IW) contained the weights between the input vector (or the input layer) and the hidden layer, whereas the second (LW) contained all the weights between the hidden layer neurons and the output layer. On the other hand, the first bias vector was consisted of the bias values of the hidden layer neurons and the second had the biases from the two output neurons. That information is summarized in Table 6, along with the description and dimentionions of each structure. As a reminder, the input vector contains 18 elements. Those values above are imperative for creating the processing unit, as they can be used to calculate its output for any input.
Before creating this unit, settlements had to be made concerning its function. As its input, our network would receive the data from the ECG in the form of a $18 \times N$ matrix, with $N$ representing the amount of heartbeats the signal contained. Every heartbeat would have been detected, then separated from the others and a wavelet transform would have been applied. Those tasks could be performed by a sensor (or a number of sensors), which would send that information to our unit. This is not a subject of study for this particular thesis. After the network within the processing unit receives the pre-processed information, it will be able to produce a vector of $N$ elements, which would indicate the condition of every heartbeat: Normal or Abnormal. The output can be produced with a strict mathematical computation, by using the structures of Table 6. Of course, the transfer functions of each layer had to be taken into consideration (chapter 1). The hidden layer neurons transmitted the signal with the $\text{tansig}(\cdot)$ function, which is the hyperbolic tangent sigmoid transfer function. According to the documentation [4], this function is mathematically equivalent to the hyperbolic tangent function $\text{tanh}(\cdot)$ and their only difference is the execution time in a Matlab code. As for the output layers, they used a $\text{softmax}(\cdot)$ transfer function, which is a commodity in all neural network classifying problems. The softmax layer and the classification layer compile to form the output layer. A neuron $j$ of the softmax layer converts the real values it receives to normalized class probabilities, by applying the function (7):

$$y_j(x) = \frac{\exp(a_j(x))}{\sum_{i=0}^{K} \exp(a_i(x))}$$  \hspace{1cm} (7)

where $a(x)$ is the input of the neuron, $y(x)$ its output and $k$ are the amount of classes of the classification problem (in this case $K = 2$). Of course, since the neurons output are normalized probabilities:
\[ 0 < y_j(x) < 1 \quad \text{and} \quad \sum_{i=0}^{K} y_i(x) = 1 \quad (8) \]

So, when an input vector of elements \( p_n \ (n = 1, \ldots, 18) \) is presented to our unit, the output for each class \( k \) (\( \text{out}_k \)) will be calculated according to (9) and (10):

\[ \text{out}_k = \text{softmax} \left( B2(k) + \sum_{j=1}^{14} LW_{j,k} \times a_j \right) \quad (9) \]

\[ \text{where} \quad a_j = \tanh \left( B1(j) + \sum_{n=1}^{18} IW_{n,j} \times a_n \right) \quad (10) \]

The class with the highest probability value, by a considerable amount, was the true class of the heart beat and would appear as the response of the unit to that specific heartbeat. That way the function of the network was accurately emulated by our unit.

Of course, the program was tested with the ECG’s from our database and the results were excellent. The target labels of the heart beats were in agreement with the program’s output, with a small margin of error, as was expected by the error percentage of classifications from the neural network.
5. Conclusions and Future Work

5.1 Conclusions

After the completion of our experimental work, we reached to the conclusions listed below.

- At first we created a single neural network for every patient, in order to classify their heart beats into two different classes. From an accuracy perspective, the networks managed excellent performance, managing in almost every case 97% correct classifications.

- Our method of dealing with the heart beats as simple values from an electrical signal, without altering or processing them in any way proved to be faulty. We faced some irregularities, as the network did not respond to alterations concerning its input and its structure. The performance of the network remained unchanged when the hidden layer increased in size, as well as when the elements of every sample increased.

- Even when tested with a larger data set and with an excessive amount of neurons on the hidden layer, the network performance remained unchanged. However, in cases when the hidden layer had a reasonable amount of neurons, the network showed improvement in error percentages when the window size increased, indicating that using a larger amount of data was a step towards the right direction.

- We applied discrete wavelet transform to the data, in order to preserve the identity of each pulse when providing them as input to the network. The results were very encouraging. When increasing the number of neurons in the hidden layer the percentage of misclassifications declined in a clear manner. Also the network performed better as more samples of each observation were used as input. That finally led us to the conclusion that the optimal network to use had to have 14 neurons in the hidden layer and the heartbeats had to be separated by 128 samples from each other.

- A network of the optimal configuration behaved excellently when new data sets were presented as input, even when their size was as small as 10% of the original data set it was trained with. Error percentages of about 0.53 % prove exactly that.

- When the network was configured and not yet trained, its response to unknown data during the training procedure was excellent. The network managed to decrease the percentage of
errors gradually, as more data were used for training and less simply for testing. That conclusion was expected, but was confirmed by the experimental procedure.

5.2 Future Work

The main focus of our work was to implement a neural network to classify heart beats as Normal or Abnormal. However, the data base that was used provided heart beats with various annotations, which indicated multiple characterizations for heart beats. In that way, a future study could expand our work by creating a network to classify heart beats in all those different classes, thus giving a deeper understanding of what a faulty (or abnormal) condition of a beat could be.

During our exploration for the optimum configuration of the neural network, the parameter that we used to reach to a conclusion was the classification accuracy. Even though that parameter is of crucial importance especially on application concerning medical conditions, other parameters can also be taken into consideration in an effort to determine the best form of input as well as the best structure for our network. Some of those could include the computational cost of the network implementation, the longevity of the training procedure if speed is of importance, sensitivity on classifying data to a specific class for applications more focused on certain conditions of the heart and many more.

Finally, we completed the experimental part of the thesis by implementing the function of our neural network to a programmable processing unit. This unit could be extended to contain sensors that are connected directly to the ECG recording stage, and therefore process the data from their raw form (detecting the heart beats, separating them, applying wavelet transform).

Whether the unit’s functionality is extended or not, it could be used on a platform based on Internet of Things, that could be wearable and would enable remote monitoring of patients without human interference and could trigger quick notifications in emergency situations.
Acknowledgements

For the completion of this thesis, I would like to thank Professor Konstantinos Siozios for his guidance, timely advice and encouragement that helped me so much with my first steps into research methodology.
References


[12] N. Belgacem, M.A. Chikh, F. Bereksi Reguig - “Supervised Classification of ECG Using Neural Networks” - Biomedical Engineering Laboratory, Department of Electronics, Science Engineering Faculty, Abou Bekr Belkaid University (2015)


“Effective ECG beat classification using higher order statistic features and genetic feature selection” - Biomedical Research Volume 28, Issue 17 (August 2017)


Václav Hlaváč – “Wavelets Transformation” Presentation

Bennet J, Ganapakasam ChA, Arputharaj K - “A Discrete Wavelet Based Feature Extraction and Hybrid Classification Technique for Microarray Data Analysis” - The Scientific World Journal Volume 2014 (2014)


Emran M. Tamil, Nor Hafeezah Kamarudin, Rosli Salleh, M. Yamani Idna Idris, Noorzaily M.Noor, and Azmi Mohd Tamil, “Heartbeat Electrocardiogram (ECG) Signal Feature Extraction Using Discrete Wavelet Transforms (DWT).”

A. Adib, M. Aynal Haque – “ECG beat classification using discrete wavelet coefficients” – Article (January 2010)