

## DIPLOMA THESIS

# Neural Network based Electrocardiography Anomaly Detection

Submitted at the Faculty of Electrical Engineering and Information Technology,  
Vienna University of Technology  
in partial fulfillment of the requirements for the degree of  
Diplom-Ingenieur (equals Master of Sciences)

under supervision of

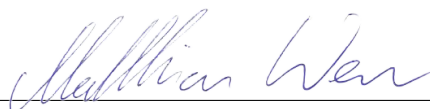
Univ. Prof. Dipl.-Ing. Dr. techn. Axel Jantsch  
Univ.Ass. Dipl.-Ing. Dr. techn. Sai Manoj Pudukotai Dinakarao

**Institut of Computertechnik Technology (E384)**  
Vienna University of Technology

by

Matthias Wess, BSc.  
Matr.Nr. 0926401  
Meranergasse 8, 2340 Mödling

6.1.2017

  
\_\_\_\_\_

## **Abstract**

Bio-signals such as Electrocardiography and Electroencephalography are the time variant signals representing the electrical outputs from the corresponding measurement instruments and are widely used to assess the health of patients. The objective of this thesis is to implement neural network based machine learning algorithm on FPGA to detect the anomalies in ECG signals, with a better performance and accuracy, compared to statistical models.

An overview of existing techniques for anomaly detection in ECG signals is presented along with their merits and demerits, implementation strategies for different aspects such as feature selection, feature reduction and classification. Based on the performed comprehensive analysis, few of the most efficient algorithms were selected for simulation in MATLAB for performance comparison. An implementation with principal component analysis for feature reduction and multi-layer perceptron for classification, proved superior to other algorithms.

For implementation on FPGA, effects of several parameters and simplification on performance, accuracy and power consumption were studied. Piecewise linear approximation for activation function and fixed point implementation were effective methods to reduce the amount of the needed resources. The resulting neural network with eight inputs and four neurons in the hidden layer, achieved in spite of the simplifications the same overall accuracy as in simulation. An accuracy of 97.5% was achieved on average for 42 records in MIT-BIH database.

These results suggest that the presented technique with several simplifications for FPGA implementation of neural networks is a valid technique with good performance but less power consumption, and area, and no significant loss of accuracy.

## Kurzfassung

Bio-Signale wie Elektrokardiographie und Elektroenzephalographie sind Signale, welche die elektrischen Ausgänge der entsprechenden Messinstrumente darstellen, und sind weit verbreitet, um die Gesundheit der Patienten zu beurteilen. Das Ziel dieser Arbeit ist die FPGA-Implementation eines auf neuronalem Netzwerk basierenden Algorithmus zur Detektion von Anomalien in EKG-Signalen, mit besserer Leistung und Genauigkeit im Vergleich zu statistischen Modellen.

Neben einem Überblick über vorhandene Techniken werden deren Vor- und Nachteile, Strategien zur Implementierung von zusätzlich nötigen Komponenten wie Merkmalsauswahl, Merkmalsreduktion und Klassifizierung präsentiert. Basierend auf der durchgeführten Analyse wurden einige der effizientesten Algorithmen für die Simulation in MATLAB zum Vergleich ausgewählt. Eine Implementierung mit Hauptkomponentenanalyse zur Merkmalsreduzierung und mehrlagigem Perzeptron zur Klassifizierung erweist sich überlegen gegenüber anderen Algorithmen.

Für die Implementierung auf FPGA wurden die Effekte von mehreren Parametern und Vereinfachungen auf Leistung, Genauigkeit und Leistungsaufnahme untersucht. Stückweise lineare Approximation für die Aktivierungsfunktion und Implementierung von Fixpunktarithmetik waren wirksame Methoden, um die Menge der benötigten Ressourcen zu reduzieren. Das resultierende künstliche neuronale Netz mit acht Eingängen und vier Neuronen in der verdeckten Schicht erreichte trotz der Vereinfachungen die gleiche Gesamtgenauigkeit wie bei der Simulation. Für 42 Datensätze der MIT-BIH-Datenbank erzielt das System eine Genauigkeit von 97,5%.

Diese Ergebnisse deuten darauf hin, dass die vorgestellte Technik mit mehreren Vereinfachungen für die FPGA-Implementierung von künstlichen neuronalen Netzwerken eine adäquate Methode mit guter Leistung, aber weniger Strom- und Flächenverbrauch und ohne signifikantem Verlust an Genauigkeit ist.

# Table of Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Computer Aided Diagnosis of Heart Diseases	1
1.1.1	Heart Diseases	2
1.1.2	Challenges	2
1.2	Artificial Neural Networks	3
1.3	Objective	3
<b>2</b>	<b>State of the Art Analysis</b>	<b>4</b>
2.1	Performance Metrics	4
2.2	Beat Detection and Feature Extraction	5
2.3	Anomaly Detection and Classification	6
2.3.1	Neural Network-based Anomaly Detection and Classification	6
2.3.2	FPGA Implementations of Neural Network-based Anomaly Detection	7
2.3.3	Comparison of ECG Anomaly Classification Systems	8
<b>3</b>	<b>ECG Anomaly Detection</b>	<b>10</b>
3.1	Importing MIT-BIH Arrhythmia Database	10
3.2	Beat Detection and Feature Extraction	11
3.2.1	Pan Tompkins Algorithm	12
3.2.2	Wavelet Transform based Feature Extraction	14
3.3	Feature Reduction	18
3.3.1	Principal Component Analysis	18
3.3.2	Fuzzy C-Means Clustering	19
3.4	Classification of ECG Anomalies with Neural Networks	19
3.4.1	Multi-Layer Perceptron	20
3.4.2	Selection of optimal Feature Set	21
3.4.3	Neural Network Topology	22
3.4.4	Training Algorithms	25
3.5	Results	27
<b>4</b>	<b>FPGA-accelerated ECG Classification</b>	<b>28</b>
4.1	System Architecture	28
4.2	MATLAB GUI and ECG Classification Process Control	31
4.3	Processing System	32
4.3.1	Software Architecture	35

4.3.2	Communication via Lightweight IP . . . . .	36
4.3.3	Interfacing ANN . . . . .	37
4.3.4	Training and Testing . . . . .	38
4.4	Neural Network FPGA Implementation . . . . .	39
4.4.1	Code Structure . . . . .	40
4.4.2	Interfaces . . . . .	41
4.4.3	Pipelining . . . . .	41
4.4.4	Fixed Point Datatypes . . . . .	41
4.4.5	Activation Functions . . . . .	44
4.5	Results . . . . .	48
4.5.1	Classification Accuracy . . . . .	48
4.5.2	Required Resources and Power Consumption . . . . .	49
4.5.3	Time Measurements . . . . .	50
<b>5</b>	<b>Conclusion</b>	<b>52</b>
	<b>Literature</b>	<b>56</b>

# Abbreviations

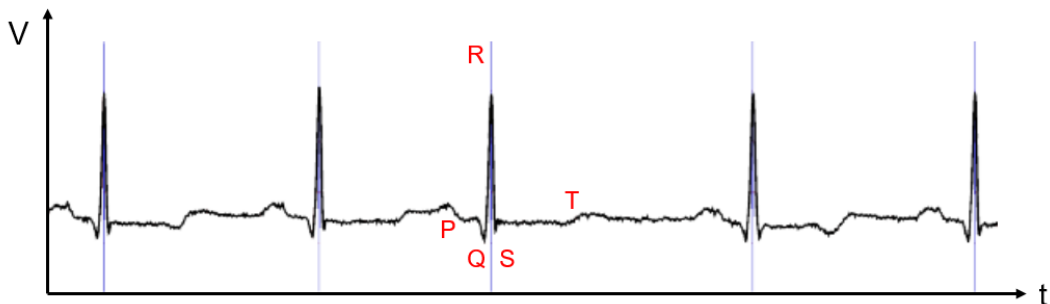
ECG	Electrocardiogram
EEG	Electroencephalogram
FPGA	Field Programmable Gate Array
ANN	Artificial Neural Network
LDA	Linear Discriminant Analysis
SVM	Support Vector Machine
VEB	Ventricular Ectopic Beats
MLP	Multi-Layer Perceptron
MNN	Modular Neural Networks
GFFNN	General Feed Forward Neural Networks
RBFNN	Radial Basis Function Neural Networks
PNN	Probabilistic Neural Networks
RNA	Reusable Neuron Architecture
BbNN	Block-based Neural Networks
EC-CGA	Elitism-Based Cellular Compact Genetic Algorithm
PCA	Principal Component Analysis
FCM	Fuzzy C-Means
PLA	Piecewise Linear Approximation
LM	Levenberg Marquardt
RPROP	Resilient Backpropagation
CG	Conjugate Gradient
IP	Intellectual Property
PS	Processing System
PL	Programmable Logic
BRAM	Block Random Access Memory
RTL	Register Transfer Level
HLS	High Level Synthesis
lwTCP	light weight Transmission Control Protocol
LUT	Look-Up-Table
FF	Flip-Flop
DSP	Digital Signal Processor

# 1 Introduction

With the heart being one of the most vital organs of the human body, assessment of its functionality is one of the crucial tasks in modern medicine. Electrocardiography (ECG) is a powerful, non-invasive tool widely used for cardiac monitoring helping cardiologists with the diagnosis of various heart diseases. With the improvements in technology, wearable devices allowing long-term measurements can be used for diagnosis of sporadic heartbeat anomalies. Due to the large amount of data, automatic detection and classification of cardiac diseases are necessary. Anomaly detection in ECG cannot be carried out solely based on amplitude of the signal. To successfully classify heart diseases the system needs to learn the characteristics of the signal and based on the learned data anomaly detection can be carried out. This chapter provides the necessary information on electrocardiography and adaptive systems, for the design of a system capable of automatically detecting ECG anomalies.

## 1.1 Computer Aided Diagnosis of Heart Diseases

ECG is the illustration of the electrical activity of the heart with time. It is recorded by placing electrodes on the patients thorax. The standard ECG consists of 12 leads which are determined by the placement and orientation of the electrodes on the body. The use of multiple leads provides additional information on particular regions of the heart [Tha10]. Figure 1.1 shows lead two of a twelve lead ECG record, with annotations for the typical waves. Each of the waves illustrates



**Figure 1.1:** A typical rhythm strip, with annotations at the typical P,Q,R,S and T-waves

activity of a particular region in the heart muscle. Higher amplitude of a wave suggests that the activated region consists of more muscle mass, creating greater electrical potential [Tha10]. The most typical part in the ECG signal is the QRS complex which consists of the Q, R and S-waves.

### 1.1.1 Heart Diseases

As most heart diseases influence the electrical activity in the heart, analysis of ECG allows cardiologists to accurately detect and classify heart diseases. Contemplating ECG from a non medical point of view, heart diseases can be distinguished based on different characteristics. The regularity defines whether a certain heart disease can be observed in every single heartbeat, or if the anomaly only occurs sporadically. Secondly, most heart diseases either lead to a change of rhythm of the heart beat or a different waveform in the ECG signal.

#### Arrhythmias

The resting heart of an average human being beats 60 to 100 times per minute. When observing a resting heart, every deviation of this usual rhythm is called arrhythmia, including disturbances in the rate, regularity or conduction of the cardiac electrical impulse [Tha10]. The term arrhythmia therefore not only describes a single aberrant beat, but also increase and decrease of the usual pulse frequency. As the normal heart rate depends on various factors, not every type of arrhythmia is dangerous to the human being. Many arrhythmias, however, require immediate therapy to prevent sudden death. Diagnosis of arrhythmia is usually carried out by analysis of ECG signals. Thereby the arrhythmia can be identified by measuring the distances between QRS complexes. Figure 1.2 shows a sporadic arrhythmia.

#### Other Anomalies

All other kinds of heart diseases mainly lead to a change of shape in the waveform. Whereas hypertrophy and enlargement of certain regions of the heart muscle lead to a regular change of shape, abnormalities of conduction can also occur sporadically. Other abnormalities like myocardial ischemia and infarction are common reasons for chest pain and can lead to sudden death [Tha10]. These kinds of anomalies can be diagnosed based on several indicators in the ECG signal. Inverted or delayed waves, for example, can indicate a certain kind of disease [Tha10]. Figure 1.2 also contains an inverted T wave.

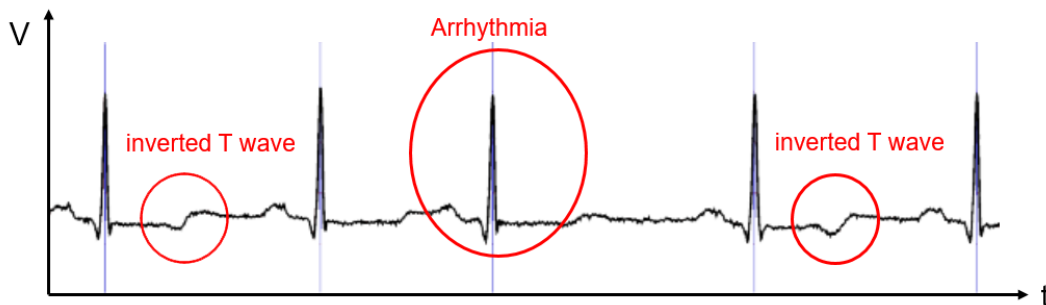


Figure 1.2: A typical rhythm strip containing an arrhythmia and an inverted T wave

### 1.1.2 Challenges

Due to the nature of certain heart diseases of only occurring sporadically, long-term measurements are necessary for diagnosis. The Holter monitor [GLG<sup>+</sup>83] offers the possibility of measuring the



heart activity for 24 to 48 hours or longer. Recent developments in modern technology also show a development of miniaturization of existing devices. Therefore it can be assumed, that in near future the recording of ECG can be performed by even smaller mobile devices. In spite of advances in technology, interpretation of the resulting ECG signals nowadays is usually performed by cardiologists assisted by devices, which automatically suggest possible anomalies. While the drawback of expert based diagnosis is the time consuming nature of the process, with automatic ECG diagnosis several new challenges have to be overcome.

- In terms of **signal quality**, **base band drift** and **signal noise** present two difficulties that have to be managed by the detection system.
- Anomaly detection can not be performed by statistical models, as ECG signals strongly vary between patients in terms of **heart rate** and **waveform**. Hence, the system needs to learn the characteristics of the signal and further based on the learned/trained data, the anomaly detection can be performed.

## 1.2 Artificial Neural Networks

Due to the previously mentioned characteristics of ECG signals, the task of ECG classification is best performed by a classification function able to adapt to the characteristics of the signal under observation. Artificial neural networks (ANN) offer a machine learning approach of making a system adaptable based on the operating environment, inputs and the information it has learned. As classification function, ANN are tolerant of some imprecision if plenty of training data is available [DR13]. If there are enough training data and sufficient computing resources for an artificial neural network, ANN can be used to estimate the output of sequence and based on the difference of estimated and the actual signal, anomalies can be detected.

Learning in human brains happens with the aid of neurons, which fire based on the input and perform simple operations. Each neuron is further connected to several hundreds of other neurons creating a neural network. Inspired by this concept, artificial neural networks make use of this structure for learning applications. Numerous variants of artificial neural networks are available in literature such as probabilistic neural networks, block based neural networks, deep learning neural networks and cognitive neural networks. The advantage of neural networks over other methods for anomaly detection, is the simple nature of one neuron, and therefore the possibility to easily resize the neural network, and therefore adapt it to the complexity of the task to solve. Furthermore, neural networks are actually able to learn the characteristics of the signal which makes them fit for the task of anomaly detection.

## 1.3 Objective

The advances in technology have made the implementation of algorithms and entire systems on low power devices possible. This thesis focuses on the study of optimization strategies for implementation of artificial neural networks on FPGA. The objective is to implement a neural network based machine learning algorithm on FPGA to detect the anomalies in ECG signals, with a better performance, accuracy and low power consumption.

## 2 State of the Art Analysis

The following chapter presents a state-of-the-art analysis for FPGA based ECG anomaly detection. The survey is separated into three parts, namely: description of performance metrics, beat detection and feature extraction, anomaly detection and classification.

### 2.1 Performance Metrics

Due to the high variety of metrics, databases, and the different focus of the publications, at first, standardized performance metrics are necessary to compare different implementations. Parameters describing the ability of a system to correctly classify ECG anomalies are: Accuracy ( $Acc$ ), Sensitivity ( $Se$ ), Specificity ( $Sp$ ) and Positive Predictivity ( $Pp$ ) ([ZKR10]). The most widely used metric for comparison of overall system performance is accuracy. Accuracy is defined in equation 2.1:

$$Acc = \frac{N_T - N_E}{N_T} * 100 \quad (2.1)$$

The variables  $N_T$  and  $N_E$  represent the total number of heartbeats within the record and the amount of classification errors, respectively. To also distinguish between false positives and false negatives, additional metrics are necessary. Sensitivity, the ratio of correctly detected events,  $T_P$  (true positives) to the total number of events is given by equation 2.2 [ZKR10]:

$$Se = \frac{T_P}{T_P + F_N} * 100 \quad (2.2)$$

$F_N$  (false negatives) is the number of missed events. Specificity is the ratio of  $T_N$  (true negatives), the number of correctly rejected non-events, to the total number of non-events. It is given by 2.3 [ZKR10]:

$$Sp = \frac{T_N}{T_N + F_P} * 100 \quad (2.3)$$

$F_P$  (false positives) is the number of falsely detected events. Lastly, positive predictivity, the ratio of number of correctly detected events to the total number of detected events is given by equation 2.4:

$$Pp = \frac{T_P}{T_P + F_P} * 100 \quad (2.4)$$

## 2.2 Beat Detection and Feature Extraction

To successfully classify or detect ECG anomalies, information on every heartbeat has to be gathered. Before extracting additional information each heartbeat has to be detected automatically. Heartbeats are usually detected with help of QRS complexes, as they have the most remarkable waveform. Extraction of additional features helps to acquired a higher level of information which can be made use of in the later classification process.

### Pan Tompkins Algorithm

Pan Tompkins algorithm is a QRS complex detection algorithm, based upon analysis of slope, amplitude, and width of ECG signals. To reduce false detections, a specially designed bandpass filter is used, which additionally allows low thresholds leading to increased sensitivity. For the standard 24 hour MIT-BIH arrhythmia database, the algorithm QRS complexes with 99.1% accuracy. [PT85] Pan Tompkins algorithm not only allows the detection of peaks but also gives additional parameters, and therefore most state-of-the-art methods are based on the differential method e.g. [ZFX11], or as in [VV13] improve the algorithm at certain stages.

### Waveform Segmentation for Feature Extraction

In addition to detection of R peaks, in [ESRCF10] extraction of further features such as P, Q, S and T-amplitudes and duration is achieved by segmentation of the waveform into R-R cycles. Minimum and maximum of Q and S waves are searched separately from the others as they are based on the R wave only. For the location of the P and T-wave a previous detection of support points is performed. With these support points the waveform is segmented to defined areas where to expect the points of interest of the ECG wave. The algorithm was implemented in *MATLAB*. To validate the method the MIT-BIH arrhythmia database was used and compared with the manual annotation, resulting in an average error of 4-6 samples at 360 Hz sample rate, depending on the level of difficulty of the sample.

### Using Wavelet Transforms for ECG Characterization

Several works make use of wavelet transforms to detect the position of certain turning points. Hence, wavelet transformation allows extraction of additional features. The algorithm presented by Sahambi et al. in [STB97] uses the first derivative of a Gaussian smoothing function as wavelet for wavelet transform to locate via zero crossings the location of the QRS complex. QRS width is then determined by location the onset and offset of the QRS. In addition the P and T waves and their features are determined. The algorithm was implemented on an optimally designed DSP hosted inside a PC. For verification the MIT-BIH arrhythmia database was used. With an accuracy of 98.8% the algorithm shows, also compared to newer publications e.g. [MAO<sup>+</sup>04] a state-of-the-art accuracy. In [PB10] B. Mazomenos et al. introduce a low complexity algorithm for the extraction of the fiducial points from ECG aiming for low-power devices. Employing the discrete wavelet transform with the Haar function being the mother wavelet an approximation of P, Q, R, S and T-turning points and duration can be extracted. The algorithm was tested on the several databases and achieved close to state-of-the-art performance while reducing the computational complexity, making it an ideal candidate for implementation on low-power systems.

## Comparison of Feature Sets

De Chazal et al. in [DCDR04] proposed an approach for classification of ECG signals into five classes. Besides heartbeat segmentation, they used morphologies of QRS and T-complexes as features for linear discriminant analysis (LDA). In the study the heartbeat fiducial points were determined manually and twelve different feature sets using multi and single-lead configurations were compared and the best configuration chosen for a comparison with existing algorithms. For performance assessment of the configuration the MIT-BIH was used and a ventricular ectopic beat (VEB) positive predictability of 81.9% was achieved. The study shows that the use of morphologies for classification eliminates the error of other feature extraction methods, and shifts the complexity of the preprocessing to the reduction of the high amount of features.

While research suggests Pan Tompkins algorithm as most effective for QRS complex detection, wavelet transform is recommended for detection of additional turning points. The segmentation of waveforms is an adequate method to increase detection accuracy of further turning points, when employing wavelet transform. Furthermore the use of morphologies as features for classification presents an alternative approach, eliminating the necessity and inaccuracy of additional feature extraction techniques.

## 2.3 Anomaly Detection and Classification

For the task of ECG arrhythmia classification, several methods are available. The results in [MK10] indicate that support vector machines (SVM) in comparison to artificial neural networks are much faster in training stage, but the generalization ability in terms of mean square error is more than three times less. Taking into account the results of [DCDR04], artificial neural networks also perform superior to linear discriminant analysis. Therefore it was considered sufficient studying the different classes and types of artificial networks. The following paragraphs show several implementations of ANNs for classification and detection of ECG anomalies. Most implementations employ several methods of preprocessing to reduce the amount of inputs of the ANN and therefore the size of the artificial neural network.

### 2.3.1 Neural Network-based Anomaly Detection and Classification

The most common neural network architecture for ECG classification is multi-layer perceptron (MLP) which was implemented and proved as efficient in several works ([CÖ07], [DR13], [ABI+03], [JNG10c], [ZKR10]). It consists of at least three layers of neurons that are fully connected amongst the neighbouring layers. The implementation by Dubey and Richariya ([DR13]) uses mostly morphological ECG features, such as R-R interval, P-R interval or R wave amplitude, as input parameters for MLP. The best results were achieved with 25 inputs and 5 neurons on each, output and hidden layer. In this constellation, the neural network classifies the records of the MIT-BIH database with 91.8% accuracy. Implementations with fewer inputs, depending on the preprocessing and feature extraction methods, achieve lower accuracies. In [ABI+03], Acharyaa et al. implemented a MLP with only four inputs and eight neurons in the hidden layer, still achieving an accuracy of 85.0%.

To compare different feature reduction methods for the later classification of ECG signals with multi-layered perceptrons (MLP) Ceylan and Özbay implemented in [CÖ07] four different algorithms. Principal component analysis (PCA), fuzzy c-means clustering (FCM) [ÖCK06], wavelet transform and a combination of PCA and FCM reduce the input data for the neural network also for the training phase. As input data, 200 samples in the intervals R-R were chosen. In the suggested FCM-PCA-MLP algorithm the first stage reduces the size of the training set, the PCA stage can be used to reduce the amount of input neurons [SAN<sup>+</sup>12] and the artificial neural network classifies the input segments into 10 classes with 99.0% accuracy rate on the MIT-BIH database.

Three different neural network approaches are compared by Jadhav et al., in [JNG11], for the binary classification into normal and diseased beats. Using 12 lead ECGs as signal source, multi-layered perceptron (MLP) [JNG10c] turns out to be the superior architecture over modular neural networks (MNN) [JNG10a] and general feed forward neural networks (GFFNN) [JNG10b], when classifying MIT-BIH arrhythmia database. In comparison to MLP, MNN consists of modules within the neural network, without communication in between themselves, and only interconnecting the outputs of the modules with the output-layer of the neural network. GFFNN are a generalization of MLP such that connections can jump over a random number of layers giving the possibility to solve certain problems more efficiently than MLP.

Another work by Zadeh, Khazaei and Ranaee ([ZKR10]) compares radial basis function neural networks (RBFNN) and probabilistic neural networks (PNN) with a MLP implementation. The RBFNN, also implemented in [RTSD12] and [HF07], uses a three layer structure with input, hidden and output layer. The hidden neurons implement the Gaussian transfer function. In PNN the middle layer is called pattern layer and has the same amount of units as there are training data. On the first layer the transfer function is a radial basis function, for the second layer it is a competitive function. Furthermore, PNN has the advantage of a simple training mechanism [ZKR10]. Results show a 95.0% accuracy for the three architectures on the MIT-BIH database. Mohamad et al. proposed in [SAN<sup>+</sup>12] the implementation of Elman neural network for detection of abnormal heart beats. Elman neural networks therefore usually extend the MLP architecture by a context layer, working as a loop back layer and consequently allowing the state of the hidden layer to be fed back as a delayed input. The work was extended, in [MMAJ<sup>+</sup>13], by adding a PCA stage to reduce the input features, improving the performance while reducing the size of the neural network, with both network settings achieving accuracies of more than 95.0%.

In previous works artificial neural networks proved as an efficient tool for ECG classification and anomaly detection. For the named tasks, MLP is the most widely used and studied ANN architecture. Most implementations make use of feature reduction techniques before the final classification stage. In particular FCM and PCA are suggested by as suitable by several authors.

### 2.3.2 FPGA Implementations of Neural Network-based Anomaly Detection

The advantages of implementing artificial neural network in FPGA lie in reduction of power consumption and size while increasing speed, since the hardware is specifically designed for the desired task.

In [SC12], Sun and Cheng propose a reusable neuron architecture (RNA) for the ANN implementation on FPGA to reduce the area and power consumption. In RNA the all layers use the same hardware implementation of the neurons. In their work they performed ECG classification, with a three-layer 51-30-12 MLP with back propagation. Compared with the flat design, the RNA

design saves 98.7% of the area and 99.1% of dynamic power, with only a 30-fold extension of the time delay.

Jiang et al. used in their work ([JKP05]) the for FPGA implementation optimized block-based neural networks (BbNN) for ECG classification. BbNN is a two dimensional reconfigurable array of blocks able to represent a ANN. As Features they used R-period and period interpolated samples. Reduced with PCA to three principal components and additionally the R-period the resulting four dimensional vector is used as an input for the BbNN. On the MIT-BIH database this configuration produced an average of 98.0% classification accuracy. In [JK07] the algorithm was improved by using additionally, Hermite characterization and high-order cumulants as features. Classifying into 12 abnormal types this system achieved an accuracy of 95.9%.

In [JC10] Jewajinda and Chongstitvatana used a BbNN architecture alongside with Elitism-based cellular compact genetic algorithm (EC-CGA) for classification. Using mainly morphological features, they achieved an accuracy on the MIT-BIH of 95.0%. To compare 32-bit floating and 16-bit fixed point architectures in terms of size, power consumption, speed and accuracy, in [ÖD15], Özdemir and Danişman realized both implementations. The algorithm classified four selected two lead ECG data records from Physionet MIT-BIH into four different classes. On the first stage the R peaks of the records where identified and 181 samples of the beat area selected. On stage two the size of the dataset is reduced with PCA to 8 principal components. With the first two stages being implemented in *MATLAB*, also different sized MLPs with back propagation were simulated. The final FPGA implementations with a 8-2-1 ANN architecture showed an accuracy over 96.0%. With the fixed point implementation being only one percent less accurate than the floating point architecture, this makes it very suitable for mobile devices.

### 2.3.3 Comparison of ECG Anomaly Classification Systems

Several circumstances complicate the comparison of different classification system, mainly, discrepancies in test conditions. Namely, the use of different databases or only excerpts of databases, result in an optimization of the system under test for a certain task, but not in good generalization. In classification the number of output classes also influences the resulting accuracy of the system. Other factors that influence overall accuracy, e.g. percentage of training data, additionally complicate the comparison. In spite of those difficulties, table 2.1 illustrates the advantages and disadvantages of several implementations. The table consists of two parts: software and

**Table 2.1:** Comparison of several ECG classification implementations

Software Implementations				
	Methods	ANN Size	Database	Accuracy
Dubey2013[DR13]	MLP	25-5-5	MIT-BIH Excerpts	91.8%
Ceylan2007[CÖ07]	FCM-PCA-MLP	10-30-10	MIT-BIH Excerpts	93.2-99.0%
ABI03[CÖ07]	MLP	4-8-4	-	85.0-95.0%
Zadeh2010[ZKR10]	MLP	13-45-3	MIT-BIH (12 Records)	95.7%
Shukri2012[SAN <sup>+</sup> 12]	Elmann NN	n-30-3	PTB Diagnostic ECG Db	99.2%
Rai2012[RTSD12]	RBFNN	n-m-3	-	90.1%
FPGA Implementations				
	Methods	ANN Size	Database	Accuracy
Sun2012[SC12]	MLP	51-30-12	-	-
Özdemir2015[ÖD15]	PCA-MLP	8-2-1	MIT-BIH (4 records)	96.0%
Jiang2005[JKP05]	PCA-BbNN	51-25-2	MIT-BIH (10 records)	98.0%

FPGA-implementations. While the software implementations mainly focus on the choice of optimal feature sets and preprocessing algorithms, the FPGA implementations are focused on the reduction of resources and power consumption. In general systems with effective preprocessing and feature reduction algorithms achieved a higher accuracy. Principal component was used in most of the works.

Summing up, for the implementation of a system automatically detecting ECG anomalies, at least three stages are necessary: beat detection, feature extraction and classification. For beat detection Pan Tompkins algorithm and wavelet transform are effective methods. Wavelet transforms also facilitate extraction of additional features. For classification, there are several types of artificial neural networks available. Other methods for classification are linear discriminant analysis and support vector machines. In addition to the required steps, feature reduction via principal component analysis or fuzzy layer clustering, improves accuracy of the system, while reducing the amount of inputs. As several works show, ANN are fit for implementation on FPGA decreasing the total power consumption of the classification system.

## 3 ECG Anomaly Detection

In order to achieve an optimal implementation on FPGA, at first, the techniques considered for beat detection, feature extraction, feature reduction and classification were implemented *MATLAB* and compared in terms of accuracy and effectiveness. This not only helped to analyse the merits and demerits of all algorithms, but also allowed select the best fitting methods for hardware optimized solution. This chapter describes, in detail, the used components and techniques, their implementation, and the resulting accuracies. In spite of several different techniques for each step of the process, the general flow of data always follows the pattern illustrated in figure 3.1.

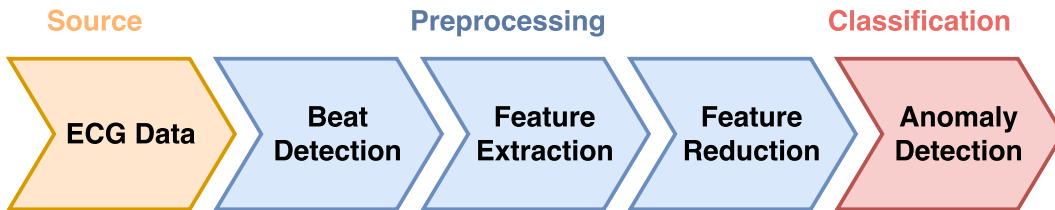


Figure 3.1: Dataflow for ECG anomaly detection

After the import of ECG data into the workspace, in preprocessing the selected data is prepared for classification. The preprocessing consists of the steps: beat detection, feature extraction and feature reduction. Beat detection determines the exact position of the heartbeats. Subsequently additional information, such as interval lengths, slopes and amplitudes, is extracted from the ECG signal, forming a vector of features for every detected beat. Reduction of redundancy contained in those feature sets is performed by feature reduction algorithms. A smaller feature vector not only reduces the amount of inputs into the classification stage, but also can increase classification accuracy.

### 3.1 Importing MIT-BIH Arrhythmia Database

The MIT-BIH arrhythmia database [GAG<sup>+</sup>00] is the standard for classification practice, which makes it easier to compare the results to other works. The database consists of 48 half-hour two lead ECG-recordings. The records, publicly available at <http://physionet.org>, are digitalized at 360 samples per channel at 11-bit resolution. For each subject there are three associated files available:



1. **Header** files contain information on the sample rate, amount of samples, information about the patient and additional notes on the ECG.
2. **Data** files consist of two lead ECG data. Since one sample of one beat is saved in 12-bits, the two bits of a time step stored continuously need 3 byte storage. According to the length of the records of 650.000 samples, every data file has the size of 1.950.000 byte.
3. **Attribute** files carry the annotations made by at least two different cardiologists. They mark the exact timing of the r-peak and contain additional information on the heart beat. The excerpt of the annotation table shown in 3.1 demonstrates that there are several comment annotations that have to be filtered for implementation.

**Table 3.1:** Excerpt of the Annotations Table

Encoding	Annotation
1	Normal Beat
2	Left Bundle Branch Block Beat
3	Right Bundle Branch Block Beat
4	Aberrated Atrial Premature Beat
5	Premature Ventricular Contraction
12	Paced Beat
14	Signal Quality Change
18	ST Change
19	T Wave Change
22	Comment Annotation
23	Measurement Annotation
28	Rhythm Change

For import into *MATLAB*, a simple function is implemented in order to facilitate the conversion of the binary data into matrices. The script is base on the implementation available alongside with the database on <https://www.physionet.org/physiotools/matlab/rddata.m>. For each record, all of the three files share the same file name, only distinguished by their file extensions (\*.hea, \*.dat, \*.atr). As the entire database is stored in the same folder, with a simple script, selecting just one file type the entire available records are listed. This procedure is used throughout the entire thesis.

## 3.2 Beat Detection and Feature Extraction

For the resulting classification accuracy of the entire system, exact beat detection is required, as the entire classification process is based on the extracted data. Missed or falsely detected beats cannot be classified correctly. In addition to detection of the heartbeats the signal processing performed on the ECG signals allows to extract additional information on the detected beat. That is why beat detection and feature extraction is best performed in one step.

### Beat Detection

First component during the classification process is beat detection. There are several different ways to determine the exact position of the heartbeats. Two methods and their implementation

delivering sufficient accuracies for further classification algorithms are discussed in this section. Beat detection itself describes the correct determination of the exact position of heartbeats. Depending on the convention this is best done by marking R peaks, which is, under normal circumstances, the highest maximum in the ECG wave, and therefore the easiest to detect. Incorrect and correct markings of R peaks can be divided into four categories:

1. **False positive** errors, imply a marking on an non-event sample. In beat detection it therefore describes a detected beat, while on the real signal there appears no R peak. This failure is likely to occur during signal disturbances, or sudden changes of signal offset.
2. **False negative** errors are missed events, hence non-detected heartbeats. Depending on the deployed detection method and the sensitivity, the cause origins from an abnormal beat with a low R maximum or other deviations.
3. **True positives** are correctly marked events. In beat detection they describe a marking on the exact position of the heartbeat.
4. **True negatives** describe all correctly non-marked samples. Considering the sampling frequency of 360 Hz and the pulse rate of an average human being, around 99.5% of the samples should be true negatives.

These definitions later on can also be used for classification, although the interpretation slightly differs.

## Feature Extraction

As heart anomalies cannot be detected by comparing the intervals between the detected beats, additional characteristics describing the morphologies of the beat are necessary. Those characteristics, in an optimal case, have to be reliable distinguishing marks for one ore more classes of anomalies.

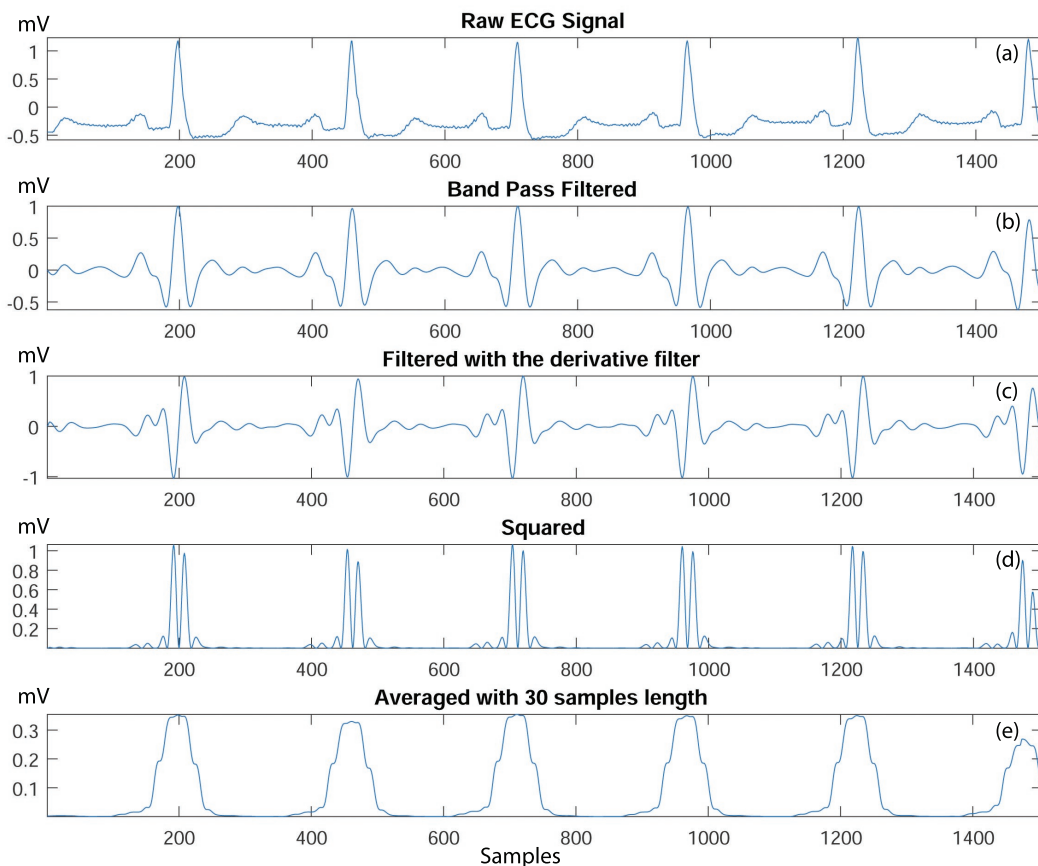
The following two sections describe algorithms performing beat detection and feature extraction.

### 3.2.1 Pan Tompkins Algorithm

As stated in section 2.2 the algorithm presented by Pan and Tompkins in [PT85], recognizes QRS complexes of ECG signals. To reliable detect QRS complexes, the algorithm implements a dual threshold technique. This avoids false positive and false negative decisions, as in case of no detection in 166.6% of the average R-R period, it starts a search-back with the lower threshold. The decision stage is based on slope, amplitude and width of R waves. For the standard MIT-BIH arrhythmia database, the algorithm detects QRS complexes with 99.1% accuracy. The implementation in *MATLAB* of the algorithm is based on the work of H. Sedghamiz [Sed14]. The single steps of the algorithm are illustrated in figure 3.2. As differentiation during the detection function, increases the impact of higher frequency noise on the accuracy, a preprocessing stage is implemented filtering noise and baseband drift.

## Preprocessing

In the first stage, the raw ECG signal (a) is bandpass filtered (b) to reduce noise and baseband wander. Next, the signal is differentiated to highlight QRS complexes (c). The following squaring stage intensifies the slope and therefore helps to distinguish between R waves and T peaks with higher than usual energy (d). With the following moving window integrator additional feature information about slope and width of the complex is obtained (e). As a last stage of the preprocessing, the algorithm marks local maxima of the average moving window output.



**Figure 3.2:** Illustration of the Pan Tompkins algorithm: raw ECG signal (a), bandpass filter to remove baseband drift and high frequency noise (b), derivation to highlight the QRS complex (c), squaring to distinguish between T and R peaks (d), average moving window (e)

## Decision stage

Finally the decision stage, which implements two thresholds in an adaptive manner, adjusts itself to the highly diverse ECG signals and therefore matches also to abnormal beats within one signal. Furthermore, the several stages and resulting intermediate signals, contain different interesting features of interest in ECG anomaly detection, such as energy, QRS width and slope.

The pulse shaped output signal of the preprocessor not only marks QRS complexes, but also high sloped T-Peaks and noise artefacts. To eliminate false detection several rules are implemented. The decision rules are outlined in the following list:

1. **Fiducial Mark:** The pulse signal first is weighted by the maxima values of the signal. A minimum of 40 samples between each R-wave is considered, since an distance between two R-peaks, shorter than 200 milliseconds is physiologically impossible [Tha10].
2. **Thresholding:** The algorithm uses two threshold values that adapt to changes of the ECG signal quality. Signal threshold identifies values above as *QRS complex candidates*, whereas the noise threshold estimates the current noise level lowering the signal threshold due to poor signal quality.
3. **Searchback:** After expiration of a timer, set to the 1.66 folds of the current R-R interval, a searchback routine is enabled to avoid false negative errors. In that case the highest peak between the signal and noise threshold is supposed to be the missing QRS complex.
4. **Elimination of Multiple Detection:** In this step the algorithm eliminates detections occurred within a 200 Milliseconds time window, to reduce false positives detections.
5. **T Wave Discrimination:** Lastly, in the interval from 200 to 360 Milliseconds after the detected QRS complex, an abnormally prominent T wave can be falsely interpreted as a R peak. To avoid false interpretation, the decision stage determines based on the mean slope of the waveform at that position, whether to mark the occurring peak or not.

The implementation in *MATLAB* is performs with an achieved accuracy of 98.0% slightly worse than the in [PT85] mentioned accuracy of 99.1% on the MIT-BIH database. Table 3.2 shows the performance metrics for the implementation of Pan Tompkins algorithm. As for additional

**Table 3.2:** Performance parameters describing the detection performance of Pan Tompkins algorithm on the MIT-BIH database.

False Positives	0.0038%
False Negatives	0.0032%
True Positives	0.3501%
True Negatives	99.6429%
Accuracy	98.0183%
Sensitivity	99.1072%
Specificity	99.9961%
Positive Predictivity	98.9133%

feature extraction, due to the highlighting of QRS complexes in the preprocessing it was easy to extract parameters, characterizing the R-Peak. Maximum, maximal slope, width and energy of every QRS complex were extracted for later classification.

### 3.2.2 Wavelet Transform based Feature Extraction

The second implemented method for beat detection and feature extraction is based on the Discrete Wavelet Transform. The method of Saurabh Pal and Madhuchhanda Mitra presented in [PM10] uses the decomposed signals to isolate ECG maxima and minima. A wavelet is a waveform

with limited duration and an average value of zero. Whereas, Fourier series analysis decomposes periodical signals efficiently, wavelet analysis advantages lie, due to the wavelets limitation in time, in analysis of transient signals. The wavelet transform is the convolution operation of the signal under observation  $s(t)$  and the wavelet function  $\psi(t)$ . The discrete wavelet transform is given as 3.1

$$X_{(a,b)} = \int_{-\infty}^{\infty} s(t)\psi_{a,b}(t) dt \quad (3.1)$$

with  $a$  as scale and  $b$  as location. In style of fast Fourier transform, fast wavelet transform can be used for efficient computation of wavelet coefficients and separation into approximation and detail signals. In filter banks, as illustrated in figure 3.3, with the recursive use of decomposition, higher level detail signals can be retained.

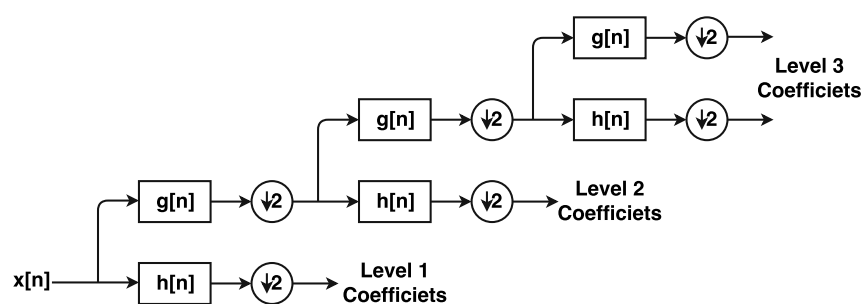


Figure 3.3: Filter bank

In the proposed algorithm, transformation is performed with Daubechies 6 wavelet, due to its similarities with the QRS complex. The signal is decomposed up to level eight. The detail signals are illustrated in figure 3.4, which already gives a good idea on the effectiveness for beat detection.

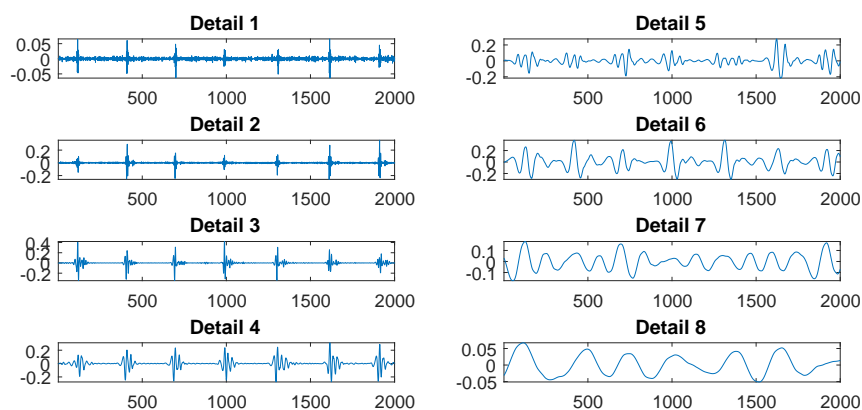


Figure 3.4: Details 1-8 of wavelet transform, QRS complexes are prominent in d2, d3, d4 and d5

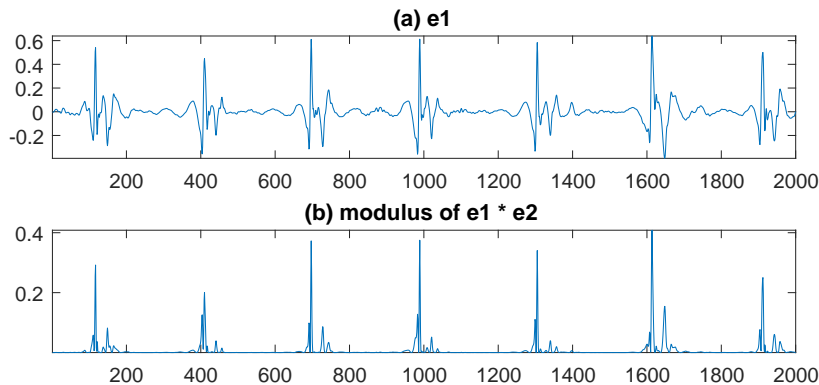
Considering the level of noise on the other signals, only  $d3, d4$  and  $d5$  are taken into account and summed in signal  $e1$  3.2.

$$e1 = d3 + d4 + d5 \quad (3.2)$$

Due to the shape (fig. 3.5 (a)) of signal  $e1$ , it is hard to identify the exact position of the R peak. To improve the process a second signal  $e2$  is defined 3.3, with  $n$  being the level of decomposition.

$$e2 = \frac{d4 * (d3 + d5)}{2^n} \quad (3.3)$$

The modulus of  $e1 * e2$ , illustrated in figure 3.5 (b), shows that in the resulting waveform contains the information of QRS complexes minus eliminated noise. In the detection stage, simple thresholding was used, as noise reduction and other mechanisms are not necessary.



**Figure 3.5:**  $e1$  containing the main information (a),  $e1 * e2$  contains information of QRS complexes (b)

In comparison to Pan Tompkins algorithm, with wavelet transformation based preprocessing the focus lies not mainly on the R peak detection, but with combinations of detail-signals further peaks and characteristics can be extracted. In [PM10] the detection methods of the additional peaks are described. The basic idea is to take the marked R peaks as a reference point and look for the other peaks relatively to it. As the Q and S-minima are part of the QRS complex and lie close the R peak, they are the simplest to detect. With signal  $e3$  defined as

$$e3 = d2 + d3 + d4 + d5 \quad (3.4)$$

they can be found, after differentiation, as the closest zero slope points right and left from the R peak. As for the T and P-wave, they are mainly found in the detail signals  $d6$  and  $d7$ . Hence  $e4$  is defined in 3.5 as

$$e4 = d6 + d7 \quad (3.5)$$

and used for P and T-peak detection. P peak is the peak on signal  $e4$  before the QRS complex and therefore can be found with a searchback mechanism. The T maximum presents one difficulty, as certain diseases lead to an inverted T wave. As a solution the algorithm at first detects the type of T wave, considering the magnitudes within the usual T wave interval.

Table 3.3 shows that all performance parameters suggest better performance of Pan Tompkins algorithm in beat detection. As the MIT-BIH database does not include annotations of the position of P, Q, S and T-peaks, accuracy for additional turning point detection was not measured. Summarizing the results, Pan Tompkins algorithm performs slightly better than wavelet transform based QRS detection. In terms of additional features, with Wavelet Transforma, it is possible to detect P, Q, S and T-wave with an average inaccuracy of around 4.0% [PM10], whereas the intermediate signals of Pan Tompkins algorithm allow to additionally extract QRS

**Table 3.3:** Performance parameters comparing QRS detection performance of Pan Tompkins algorithm and wavelet transform

	Pan Tompkins	Wavelet Transform
False Positives	0.0038%	0.0043%
False Negatives	0.0032%	0.0047%
True Positives	0.3501%	0.3485%
True Negatives	99.6429%	99.6425%
Accuracy	98.0183%	97.4354%
Sensitivity	99.1072%	98.6601%
Specificity	99.9961%	99.9957%
Positive Predictivity	98.9133%	98.7738%

characteristics.

For the feature vector<sup>1</sup> the R-R intervals relatively to the heartbeat before and after the detected QRS complex were taken into account. In addition the interval lengths from the R peak to the other extrema are computed. Furthermore the proposed feature sets under survey in [DCDR04] suggest the extraction of plain morphologies to the list of features. Due to the different types of occurring heartbeat anomalies, each characteristic has varying significance for each anomaly. As mentioned in chapter 1, arrhythmias mainly influence R-R interval lengths.

A problem that occurs in classification is, that arrhythmic heartbeats also change the extracted interval lengths of the preceding and succeeding heartbeat. For classification algorithms, this effect can lead to incorrect training. To optimize the results of training, a short algorithm was implemented, that preselects obvious arrhythmical beats and marks them. Furthermore the short piece of code sets the pre R-R interval to the average of the last five beats for the following QRS complex, to prevent false training. This mechanism increased the obtained classification accuracies significantly. Table 3.4 lists all extracted features.

**Table 3.4:** List of all extracted features

Nr. of Features	Feature Type
R Intervals	<ul style="list-style-type: none"> <li>•Pre R-R Interval Length</li> <li>•Post R-R Interval Length</li> </ul>
Morphology	<ul style="list-style-type: none"> <li>•Pre-QRS (80 Samples)</li> <li>•Post-QRS (99 Samples)</li> </ul>
Heartbeat Intervals	<ul style="list-style-type: none"> <li>•P-R</li> <li>•Q-R</li> <li>•R-S</li> <li>•R-T</li> </ul>
QRS Characteristics	<ul style="list-style-type: none"> <li>•Maximum Value</li> <li>•Width</li> <li>•Energy</li> <li>•Maximum Slope</li> </ul>
Additional	<ul style="list-style-type: none"> <li>•Arrhythmia Parameter</li> </ul>

<sup>1</sup>input vector for the neural network

### 3.3 Feature Reduction

The extracted characteristics of the ECG waves, all together, contain a high amount of redundancy. Aiming to reduce the size of the input vector for the classification algorithm two techniques for feature reduction were used and compared. To achieve an efficient workflow, at first different classification methods (section 3.4) were used with various feature sets as inputs, and later on compared with the results without feature reduction.

#### 3.3.1 Principal Component Analysis

Principal component analysis (PCA) is a common technique for feature extraction and dimensionality reduction, assuming that most information about the classes is contained alongside the directions in which the largest variation can be found. The number of principal component is lower or equals the number of original variables. The first principal component has the highest possible variance. Each succeeding principal component is defined in such a manner, that it is orthogonal to the preceding component and at the same time has the largest possible variance. Many state-of-the-art works in classification implement PCA as a feature reduction method e.g. [CÖ07, ÖD15, JK07].

In a supervised learning environment PCA is applied on the training data set. This not only leads to a reduced dimensionality of the dataset, but also sorts the output data in terms of variance. Assuming that the most information is contained in the subspace spanned by the first principal axes, no significant data will be lost [CÖ07]. In the later testing period, the retained transformation matrix consisting of the  $m$  principal axes  $T_1, T_2, \dots, T_m$ , can be used with  $[T_1^T, T_2^T, \dots, T_m^T]$  to reduce the dimension of the test data as well.

In *MATLAB*, principal component analysis is implemented as a single function, that returns the transformed data, alongside with the transformation matrix and percentage of total variance explained by each principal component. Use of principal component analysis on the morphologies seems the most logical step since they contain most redundancy, because a short series of samples usually carries almost the same information. Table 3.5 shows the percentage of the total variance and table 3.6 the accumulated total variance of ten principal components computed for the first five records of the MIT-BIH arrhythmia database.

**Table 3.5:** Percentage of total variance for the first ten principal components for records of the MIT-BIH, for some records the first principal component already contains more than 75.0% of total variance information

Record	Percentage of the Total Variance									
100	53.47	25.51	7.50	3.80	2.79	1.35	0.85	0.80	0.47	0.39
101	82.92	6.23	2.80	2.27	1.49	1.43	0.48	0.40	0.28	0.14
102	34.09	25.29	13.55	9.39	4.08	3.42	2.36	1.88	1.14	0.63
103	78.52	7.03	5.43	4.19	2.05	0.72	0.50	0.33	0.19	0.12
104	54.27	20.56	7.08	3.56	2.30	1.64	1.22	0.85	0.77	0.67

Interesting to point out is that the accumulated percentage for the first ten records occasionally already holds more than 99.0% of the variance information. On the contrary, for record 104 it only contains roughly 92.9%.



**Table 3.6:** Accumulated percentage of total variance for the first ten principal components for records of the MIT-BIH

Record	Accumulated Percentage of the Total Variance									
100	53.47	78.99	86.49	90.30	93.10	94.46	95.31	96.11	96.59	96.98
101	82.92	89.15	91.96	94.24	95.73	97.17	97.66	98.06	98.34	98.49
102	34.09	59.38	72.94	82.33	86.41	89.83	92.20	94.08	95.22	95.85
103	78.52	85.55	90.97	95.16	97.21	97.93	98.43	98.76	98.95	99.07
104	54.27	74.83	81.91	85.47	87.76	89.41	90.62	91.48	92.25	92.92

### 3.3.2 Fuzzy C-Means Clustering

In clustering, data is divided into a fixed number of classes. Whereas in non-fuzzy clustering, each data point can only be part of one cluster, in fuzzy clustering, data can belong to multiple clusters. The membership of the data grades the distance to the cluster centres. Iteratively updating the cluster centres and the membership grade, the position of the cluster centres is corrected. The update works based on an objective function, which represents the distance from each data point to the cluster center weighted by the membership grade. Fuzzy c-means clustering is also implemented in MATLAB.

## 3.4 Classification of ECG Anomalies with Neural Networks

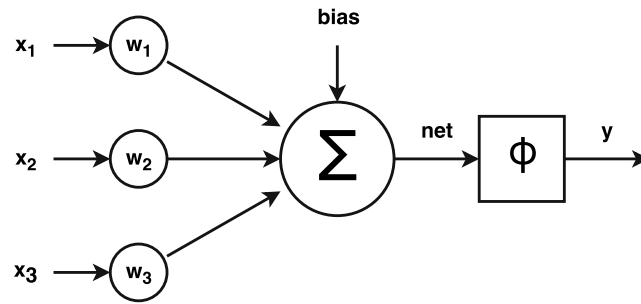
One of the most considered mechanisms to successfully classify and therefore detect ECG anomalies is artificial neural networks. Artificial neural network is a network based on the topology of biological neural networks. In case of this work they are used to correctly classify heartbeats. Chapter 2 lists several existing ANN topologies, and presents implementations comparing their accuracies for ECG anomaly detection. In general artificial neural networks can be characterized by three properties:

- **Architecture** of the neural network, describes the topology, size and connections.
- **Activation Functions** make up the decision rules whether a neuron is activated or not. Therefore usually sigmoid functions are being used.
- **Learning Algorithms** specify the way of training the neural network and adapt the weights and biases to the training data.

The basic element of artificial neural networks are the artificial neurons. Akin to the biological neuron, the artificial neurons main task is to forward the incoming stimuli, depending on their intensity. This behaviour, modelled by the structure displayed in figure 3.6, results in equation 3.6.

$$y(x_1, \dots, x_n) = \phi\left(\sum_{m=1}^n (x_m * w_m) + bias\right) \quad (3.6)$$

For the implementation this means that one neural unit has to perform several additions and multiplications, to accumulate the with  $w_1, w_2, \dots, w_n$  weighted inputs  $x_1, x_2, \dots, x_n$ , with  $n$  being the number of input signals, and the *bias*. The last function block applies the activation function

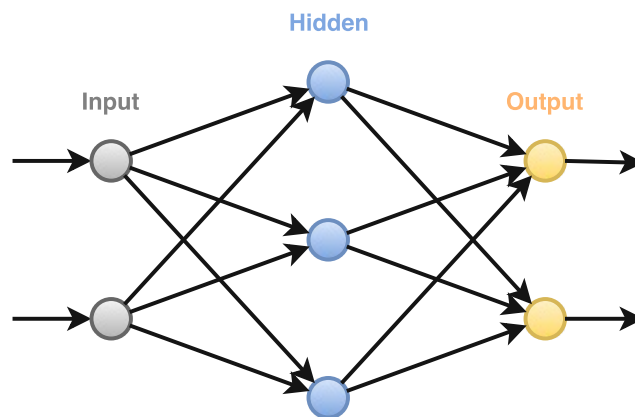


**Figure 3.6:** Topology of an **artificial neuron**: inputs  $x_1$  to  $x_3$  are multiplied with the corresponding weights  $w_1$  to  $w_3$  and accumulated with the bias in  $\Sigma$  resulting in  $net$ , activation function  $\phi$  applied on the  $net$  signal determines the output  $y$  of the neuron

on the resulting  $net$  signal. Due to the usual sigmoid characteristics of the activation function the resulting output signal  $y$  is limited to set boundaries. A connection of multiple artificial neurons in several layers, results in an artificial neural network. Based on the prior research, in this work a multi-layer perceptron is implemented due to its simple and effective structure, while achieving good accuracy classifying heartbeats.

### 3.4.1 Multi-Layer Perceptron

For classification, Multi-Layer Perceptrons (MLPs) have proven a mighty tool, with several advantages over statistical methods. MLPs are feedforward neural networks, meaning that the data flows, without loops, from input to output layer. There is at least one hidden layer between the input, and the output layer, enabling the MLP to solve non-linear problems [Zha00]. Therefore the training of MLPs is done with backpropagation algorithms performing supervised learning.



**Figure 3.7:** Structure of a 2-3-2 multi-layer perceptron The input layer of a MLP is in comparison to the other layers usually just for distribution of the signals and does not provide the entire functionality of a neuron layer.

For the implementation of the MLP in *MATLAB* the neural networks toolbox was used, as it provides a simple possibility to compare different network structures, sizes, activation functions and training algorithms. For the comparison of the entire preprocessing block and different

techniques, a standard MLP with one hidden layer, with the size of one to two thirds of the input layer, was defined. During this phase Levenberg-Marquardt-Algorithm was used, due to its higher robustness<sup>2</sup> compared to Gauss-Newton-Algorithm [Mor78]. The whole testing was done in all cases record by record, for all patients contained in the MIT-BIH database. The chosen training rate for the neural network was set to 70.0%. To find the optimal preprocessing techniques and size of the neural network for hardware implementations, several difficulties have to be overcome:

- Finding the optimal feature set for classification.
- The different preprocessing and feature reduction techniques always have their advantages and disadvantages. A goal for the hardware implementation is to minimize the size of the input vector, hence the optimal algorithm reduces the size of the vector without reducing the accuracy of the neural network.
- Once selected a preprocessing algorithm, different topologies<sup>3</sup> for the neural network have to be compared in terms of accuracy.
- A comparison of training algorithms is necessary to choose a robust and easy to implement algorithm.

### 3.4.2 Selection of optimal Feature Set

For comparison of feature sets, different combinations of features were directly fed to a neural network, without any further feature reduction. The considering the morphology from 80 samples before the R peak to 100 samples after it together with additional information about the R-R intervals, proved superior to peak characteristics. Furthermore adding the extracted peak information to the morphology feature set did not improve the accuracy of the entire system. Therefore the first mentioned feature set was recognized as the optimal set.

Two different techniques were considered for detection of arrhythmic beats. The first one includes the R-R intervals to the anterior and posterior R-peaks as additional features. A drawback of this method is that an arrhythmic beat influences both related R-R intervals and therefore also R-R intervals of the previous and succeeding beat. This can lead to false training of the neural network and has to be avoided. The problem is illustrated in table 3.7.

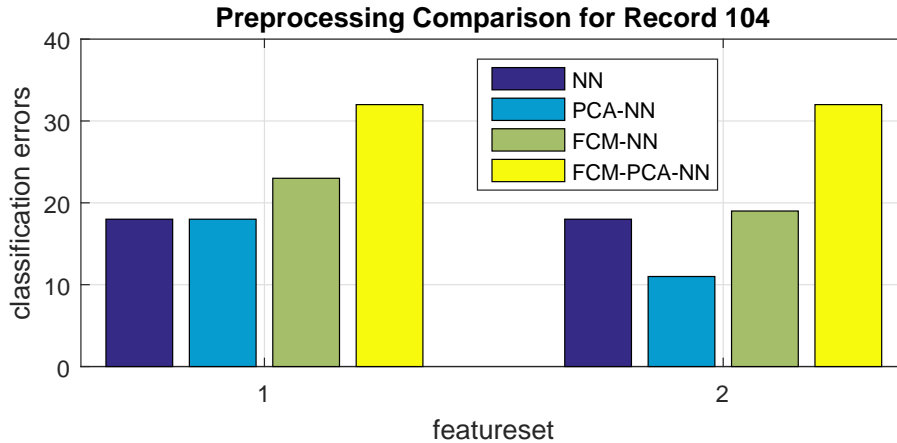
**Table 3.7:** Intervals measured in samples, the arrhythmia occurring in beat nr. 460 also influences the pre-RR interval of beat 461, the interval length (351) is corrected to 267.9 and the arrhythmia parameter is set to 1

Beat Nr.	RR-pre	RR-post	corrected RR	Arrhythmia
458	276	270	276	No
459	270	183	270	No
460	183	<b>351</b>	183	Yes
461	<b>351</b>	267	<b>267.9</b>	No
462	267	273	267	No

<sup>2</sup>meaning higher possibility of convergence even if it starts very far off the final minimum

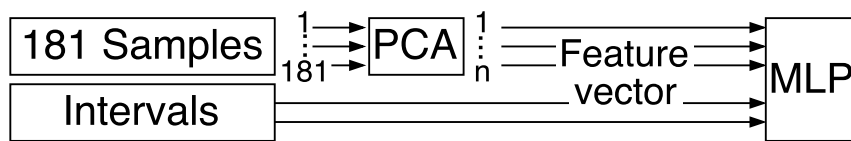
<sup>3</sup>amount and size of hidden layers can be varied

The second method implements the technique mentioned in section 3.2.2 and adds both, the corrected preceding R-R interval and the arrhythmia parameter to the feature set. For comparison three different preprocessing mechanisms (PCA-NN, FCM-NN, FCM-PCA-NN) were implemented and evaluated with the MIT-BIH database. For figure 3.8 the classification methods were applied on record 104. In that illustration PCA and FCM is still not used for feature reduction to avoid loss of information and ensure better comparability.



**Figure 3.8:** Feature set 1 is based on RR-pre and RR-post interval and the morphology, feature set 2 uses corrected RR-pre interval, arrhythmia parameter and the morphology

As a consequence of these measurements for the further work PCA-NN was chosen for implementation. An advantage of PCA is that a high percentage of variance and information is already contained in the first principal components. Hence to reduce the size of the neural network, it is possible to use fewer principal components as an input. A second advantage of PCA, is the fact that the transformation from the initial set of values to the principal components is an orthogonal transformation. As a consequence, it is possible to compute the transformation matrix for the training data and later on, transform the testing and verification data into the principal component space. Figure 3.9 illustrates the final classification architecture.



**Figure 3.9:** Feature set 1 is based on RR-pre and RR-post interval and the morphology, feature set 2 uses corrected RR-pre interval, arrhythmia parameter and the morphology

### 3.4.3 Neural Network Topology

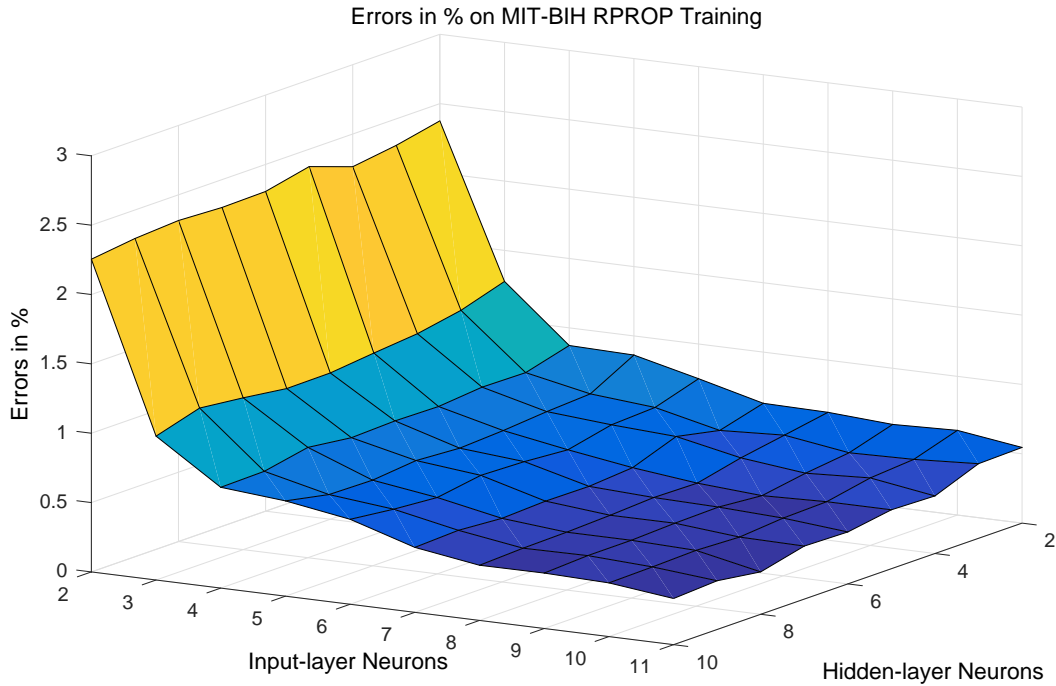
To fully comprehend the impact of the amount of neurons in each layer on the accuracy, classification for the entire MIT-BIH database was performed for different configurations of ANN. The parameters under investigation were primarily the amounts of neurons per layer. While the number of output neurons is defined by the amount of classes, additionally further hidden layers can be added to the neural network. The following parameters were studied:

- **Number of input layer neurons:** The input layer only distributes the inputs to all hidden layer neurons. Therefore the number of input layer neuron only stands for the number of inputs of the neural network. In the selected architecture 3.9 an increase of input layer neurons equals a use of more principal components.
- **Number of hidden layer neurons:** The hidden layer enables the neural network to solve non linear problems. Depending on the complexity of the problem, a certain number of hidden layer neurons is necessary. Choosing a number of neurons lower than the minimum required amount, underfitting may occur. With more than necessary neurons in the hidden layer overfitting may occur, meaning that the neural network will be adapted too exact to the training data and a good generalization can not be achieved. [Kar12]
- **Number of hidden layers:** The amount of needed hidden layers in neural networks is defined by the complexity of the problem. For linearly separable data, there is no need of a hidden layer at all. Increasing the number of hidden layers increases the complexity of the neural network more than increasing the number of neurons in a single hidden layer, since the training algorithm has to update the weights of all the layers and therefore the balancing stage also has to consider all the weights between the hidden layers.

Finding the best fitting topology and consequently size for the neural network is one of the most crucial tasks during the design. The study of S. Karsoliya in [Kar12] describes several tasks performed by neural networks, and how the optimal size of the layers varies from a few neurons to several thousand neurons in multiple hidden layer. As previously mentioned, overfitting and underfitting have to be avoided to achieve optimal results. Therefore, the size of the hidden layer(s), has to be fitting to the problem, and secondly the training algorithm has to deliver a good generalization of the problem. The paper also notes three rules of thumb to determine the number of neurons in the hidden nodes:

- The number of hidden layer neurons should be around 2/3 of the size of the input layer. If this is insufficient the number of output layer neurons can be added later on. [BG97]
- The number of hidden layer neurons should not exceed twice the number of neurons in input layer. [BL97]
- The number of neurons in hidden layer should lie between the input layer size and the output layer size. [Blu92]

Aiming at hardware implementation, the goal is to find a small but accurate architecture. Based on the previous results it can be assumed that by reducing the number of considered principal components to a certain extent, the accuracy will not drop significantly, as the most information lies in the first principal components. The number of inputs necessary for accurate classification was determined empirically. Depending on the patient and the types of anomalies contained in the record of MIT-BIH database, the number of inputs should lie somewhere between five and fifteen. A further increase of inputs does not improve the accuracy but only the complexity of the neural network. In order to determine the accuracy of several configurations, simulations for each architecture were performed. The three dimensional chart in figure 3.10 shows the dependency of classification accuracy on the amount of input neurons and hidden layer neurons. Classification was performed for all records of the MIT-BIH arrhythmia database, with separate training and testing phase for each patient. Training was performed with resilient backpropagation.



**Figure 3.10:** False classifications in dependency of the amount of input layer and hidden layer-neurons, the classification accuracy of the system can not be increased to 100% but has its limit around 99.8% depending on starting conditions and amount of neurons

Due to the random starting conditions of the weights and biases, every training of neural networks delivers different parameters. With the primary intention of rating the classification accuracy, each training and testing period, was performed multiple times for every neural network configuration, and the lowest error rate was selected. This eliminated the dependency of the results on the training algorithm and starting conditions, and helped to only rate the neural network configuration itself. Additionally with the multiple results for each neural network, it was possible to calculate the variance of the classification accuracy and scan for correlation with the sizes of hidden and input layer. With the sole increase of used neurons in the hidden layer and input layer, the overall accuracy of the neural networks could not be arbitrarily improved. After exceeding the number of 8 neurons in the hidden layer, the result of the training only depends on the starting conditions, the length of the training phase and the training algorithm itself. On the other hand increasing the number of input layer neurons doesn't improve accuracy of classification if there are not enough neurons in the hidden layer. In addition to that, more than 15 input neurons do not significantly improve classification accuracy. It can be concluded, that with PCA the number of inputs for this exact problem, can be reduced from 182 to 15, without significant loss of accuracy.

In accordance with [Kar12] the optimal number of hidden nodes in figure 3.10, depends not only on the number of inputs and outputs, but also on the complexity of the problem, and therefore does not scale with  $2/3$  of the input neurons. Furthermore already during simulation two hidden layers did not improve classification accuracy. Therefore further investigation of complexity and needed resources were not performed.

### 3.4.4 Training Algorithms

For training of neural networks there exist different methods and algorithms. During training of neural networks the weights and biases are set in a manner, that the neural network outputs, match the control pattern. Therefore, in supervised learning environments the results for the training dataset have to be known. For feed forward networks back-propagation is the most widely used technique. For training the dataset is divided into two blocks. The first of these blocks is used for training of the neural network. In batch training, at first the error and outputs for the entire training dataset is computed. Afterwards, based on the gradients of the activation functions, the weights are corrected and updated. The process of computing the outputs and updating weights and biases is called epoch. These steps are repeated until the end-condition is fulfilled. The second block, contains data for testing. During testing phase, the weights of the neural network are not updated. The basic problem of training a network, is to teach the network to correctly classify samples of the testing dataset. Two main problems can occur:

- **Overfitting** occurs, when the neural network is trained too well on the training dataset, will not generalize well and therefore will not correctly classify the test data. Overfitting occurs, if the number of hidden layer neurons is too high for the complexity of the problem, or the training algorithm does not fit the task.
- **Underfitting** occurs, when either the training phase is too short, or there are not enough neurons in the hidden layers and the neural network does not adapt well to the complexity if the classification task.

In general there is no training algorithm that fits every task. Problems of backpropagation techniques are a slow convergence, or the chance for the algorithm to terminate in a local minimum. Three training algorithms were compared in terms of performance and speed of convergence for the desired classification task. In [PS11] F.Paulin and A.Santhakumaran compare the performance of training algorithms for feed forward artificial neural networks for classification of breast cancer. Levenberg Marquardt (LM) performs slightly better than Resilient backpropagation (RPROP) and Conjugate Gradient (CG), in terms of accuracy of diagnosis. In [Kiş05] a comparison of RPROP, CG ad LM is performed for the tasks of stream-flow forecasting and determination of lateral stress in cohesionless soils. In the two case-studies RPROP outperforms the Levenberg-Marquardt algorithm in terms of accuracy during the testing phase. Even though the two studied tasks differ from ECG anomaly detection, the study demonstrates that RPROP offers better generalization ability.

#### Gradient Descent Training

Gradient descent algorithms update the weights of neural networks in the direction of the negative gradient of the performance function. To compute the weights and biases of output and hidden layers, at first, the errors of neural network outputs are calculated. Basic back propagation allows the estimation of new weights and biases. There are several drawbacks of gradient descent methods. Firstly the learning rate, which scales the derivative and therefore has a big influence on the time needed to reach convergence, is fixed. A too small learning rate will lead to a long time of training, while a large learning rate may lead to oscillation [RB93]. There are improvements of the normal gradient descent version that get rid of this problem with introducing a momentum parameter. However, the problem of choosing the right learning rate in that case just gets shifted to the selection of the optimal momentum parameter.

## Resilient Backpropagation

Resilient backpropagation, presented in [RB93], makes use of the characteristics of sigmoid activation functions in multi-layer neural networks. One problem of gradient descent algorithms is based on the fact that the sigmoid functions have a slope close to zero for large inputs. This causes networks trained with steepest descent to change weights and biases in only very small steps, even if they are still far from their optimal values. resilient backpropagation computes the direction of the weight update, based on the sign of the derivative. To determine the size of the weight change an update value is introduced. There is a separate update value for every weight and bias, which is increased by a fixed factor whenever the derivative of the performance function with respect to that weight has the same sign for two successive iterations. The update value decreases when the sign of the performance function with respect to that weight changes [Kis05]. This mechanism leads to, compared to gradient descent training, faster convergence and at the same time avoids oscillation.

## Levenberg-Marquardt Algorithm

Levenberg-Marquardt algorithm [Mor78] is a second-order training algorithm that avoids having to compute the Hessian matrix. The algorithm makes approximates the Hessian matrix as

$$H = J^T J \quad (3.7)$$

where  $J$  is the Jacobian Matrix. The gradient is computed as

$$g = J^T e \quad (3.8)$$

with  $e$  being the network output errors. The newton-like update function

$$x_{k+1} = x_k - [J^T J + \mu I]^{-1} J^T e \quad (3.9)$$

becomes Newton's method with  $\mu$  close to zero. With a larger  $\mu$  it becomes a gradient descent update. After a successful step  $\mu$  is decreased and increased only when a step would increase the performance function [Kis05]. Levenberg-Marquardt algorithm secures faster convergence than gradient descent techniques.

To pick the best fitting training algorithm for the given task, the three considered algorithms were compared in terms of complexity, speed of convergence and their performance during training and testing. Table 3.8 shows a summary of the ratings. Due to it's slow convergence, normal gradient

**Table 3.8:** Comparison of training algorithms

Algorithm	Complexity	Convergence	Training Performance	Testing Performance
Gradient Descend	Low	Very Slow	Medium	Medium
Resilient Backpropagation	Medium	Medium	Medium	Good
Levenberg-Marquardt	Medium	Fast	Good	Medium

descend backpropagation was not an option for this work. The advantage of Levenberg-Marquardt algorithm is the fast convergence and therefore short training time. As shown in [Kis05], for some tasks, RPROP outperforms Levenberg-Marquardt algorithm in terms of accuracy during the testing phase. With that in mind RPROP was selected as training algorithm for this work.



### **3.5 Results**

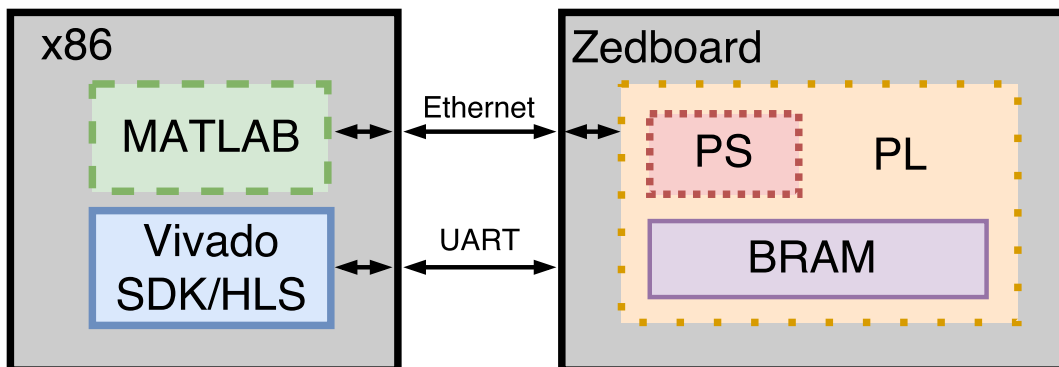
The MATLAB implementation shows that artificial neural networks can efficiently classify ECG anomalies. For beat detection Pan-Tompkins algorithm proved superior to wavelet transform based QRS detection. Best results in terms of classification accuracy were achieved with a feature set consisting of 181 samples, in the area of the detected R peak, in combination with the corrected pre-R-R interval and the newly defined arrhythmia parameter. PCA reduces efficiently the dimensions of the feature vector, which allows a lower number of required input neurons for the MLP. To determine the optimal number of neurons in each layer, an extensive analysis on the entire MIT-BIH database was performed. A minimum of five input layer and 3 hidden layer-neurons is required for effective classification. Classification accuracy can be increased by adding additional neurons to each layer. Highest accuracy reached was 99.7% with 20 inputs and 12 hidden layer neurons. For neural network training, RPROP proved as the most efficient algorithm.

## 4 FPGA-accelerated ECG Classification

The following chapter describes the implementation, of the previously selected classification methods on a hardware platform. Goal of this task was to reduce power consumption and computation time, while maintaining a high classification accuracy. Therefore, at first a fitting hardware platform was chosen and the previously developed algorithms mapped on different components.

### 4.1 System Architecture

The development of an efficient artificial network implementation on FPGA, was one of the main points in this thesis. To be able to individually implement the components, at first a hardware architecture has to be designed. For the further implementation all the components have to perform the desired tasks while facilitating a flexible design process. With the goal developing a flexible ANN structure, implementation of the artificial neural network was performed with **High Level Synthesis**. With *Vivado HLS*, Xilinx offers a development environment that translates C/C++ code into Intellectual Property (IP) components<sup>1</sup> that can be imported into Vivado block designs. Figure 4.1 illustrates the selected hardware architecture and interfaces. ECG data can be processed in MATLAB and transferred via Ethernet to the FPGA development board, which performs the classification task.



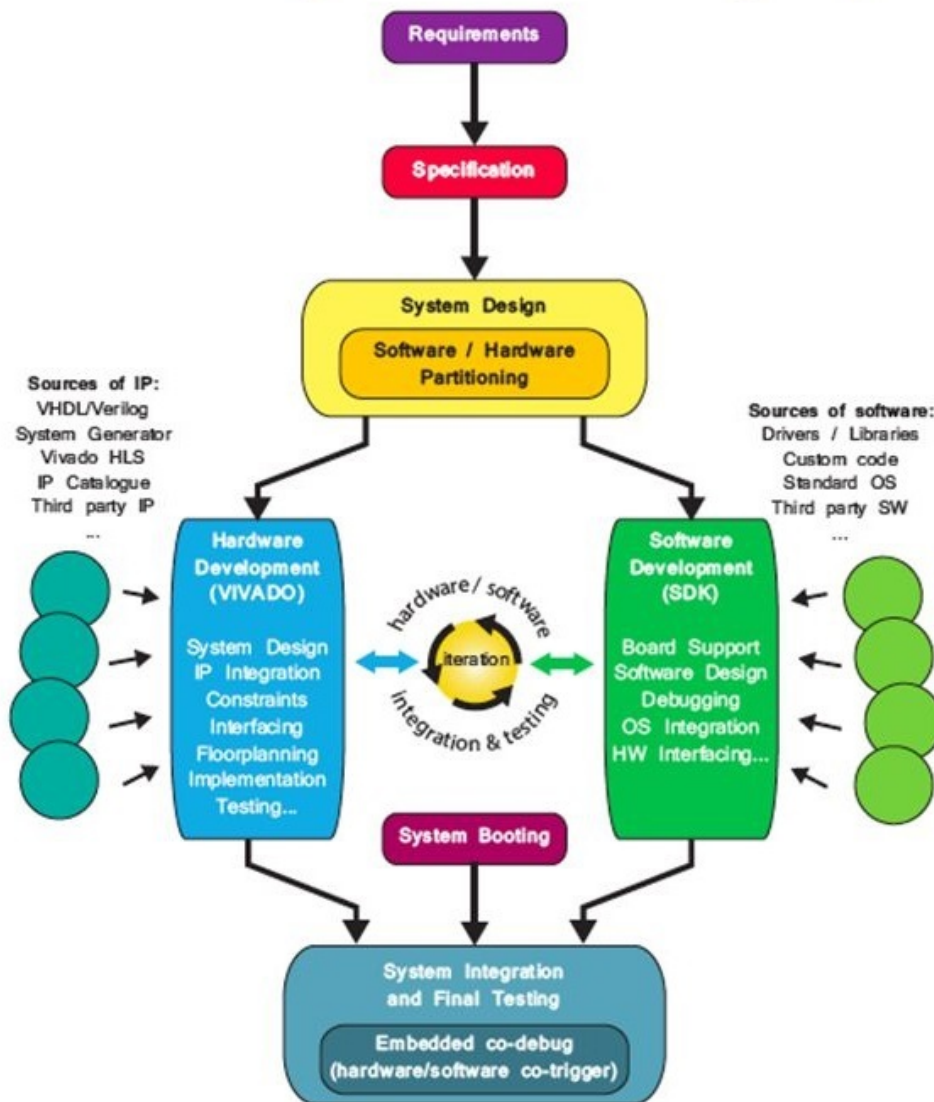
**Figure 4.1:** The **hardware architecture** shows the employed Zedboard, with Processing System (PS) and Programmable Logic (PL), Block RAM (BRAM), UART over USB is used for programming and debugging, Ethernet for Data I/O

<sup>1</sup>The IP components are encoded in *verilog*

To employ the created IP components directly on a Zynq-7000 SoC board, **Zedboard** was chosen as development kit. With all necessary interfaces on one board, it is fit for rapid prototyping and proof-of-concept development. In addition to the interfaces provided for communication between PC and the board, useful components as an OLED display for easy on board debugging are implemented. The first conceptual design contemplates a data input of from *MATLAB* over Ethernet to Zedboard. After classification on the Zedboard the results should display on either on the OLED screen or be transferred back to *MATLAB*.

Figure 4.2 illustrates the design flow for Zynq SoCs. After defining the requirements and specifications, the entire system is split into two parts. The most frequently used components are implemented in the programmable logic. Hardware design, floorplanning and implementation is performed in *Vivado*. More complex and flexible structures are implemented on PS.

## Basic Design Flow for Zynq SoC



**Figure 4.2:** Illustration of the design flow for Zynq SoCs, the design process is separated into hardware and software design (Source:[CEES14])

To map the in chapter 3 presented software design on hardware, the entire process was divided into sub-sequences. These sub-sequences were analyzed and rated by their complexity to determine the best mapping. As shown in figure 4.3 preprocessing and evaluation of ECG and classification data is performed in MATLAB. After transmitting the data via Ethernet to Zedboard, the processing system controls training and testing of the artificial neural network, implemented on the programmable logic.



**Figure 4.3:** Hardware/software mapping, preprocessing and evaluation are performed on a X86 platform, training and testing of the neural network are controlled by processing system, neural network itself is implemented on programmable logic

Preprocessing and evaluation are necessary steps for proof-of-concept, but only represent the testing environment. therefore a software based implementation was sufficient. Training and testing of the neural network is performed on the processing system as it offered flexibility during the prototyping phase and therefore allowed rapid evaluation of the implemented ANN. ANN is implemented on the programmable logic to accelerate computation speed and decrease power consumption.

Before selecting an ANN architecture and sizes of hidden and input layer, an estimation of the resulting required resources and further parameters was necessary. Therefore, working on a high abstraction level was crucial for the development of the neural network implementation. High level synthesis allows development of vhdl/verilog modules in high-level programming languages. *Vivado HLS* is a High Level Synthesis tool for translation of C/C++ code to verilog modules. A case study compared two alternative algorithms for K-means clustering, with hand written Register Transfer Level (RTL) code. The performance gap between the HLS-derived and hand-written RTL implementations is approximately a factor of two in terms of area-time product [WBC13]. While FPGA resource consummation of the HLS designs was approximately the same as of the hand-written design, latency is degraded by a factor of **30**. Optimizing the c-code for HLS implementation deploying parallelization and pipelining reduced the factor to **3.8**. Taking into account the reduced design time, *Vivado HLS* can be accepted as a powerful tool for prototyping. In *Vivado HLS*, the developed design can be synthesized and later on exported as a verilog IP. For debugging and testing reasons it is recommended to used C/RTL co-simulation for error calculation. In case of the ANN it was implemented in floating and fixed point data types.

Created IP packages can be imported into *Vivado Design Suite*. In *Vivado*, the design and interfacing of the entire system is defined. Third party IPs and the imported IPs can be combined with IPs from the Vivado IP catalog to a block design. The design is translated into a bitstream. For the software development Xilinx offers *Vivado SDK*, an eclipse based software development environment. In combination with the board support packages, C/C++ projects can access the hardware through the previously defined interfaces. Testing and training of ANN was mapped, alongside with interfacing the IPs and the handling of received TCP-packages, on the processing system. The following sections describe in detail the implementations all sub-sequences.

## 4.2 MATLAB GUI and ECG Classification Process Control

The entire process of ECG classification is controlled via MATLAB. Controlled by a Graphical User Interface (GUI), ECG data is fed to the classification hardware. For successful transmission and classification, the software on the Zedboard has to be running. The GUI, shown in figure 4.4, allows to select the record to classify and displays the classification results. In addition the factor of training data can be edited. Additional components can be easily added to the user interface.

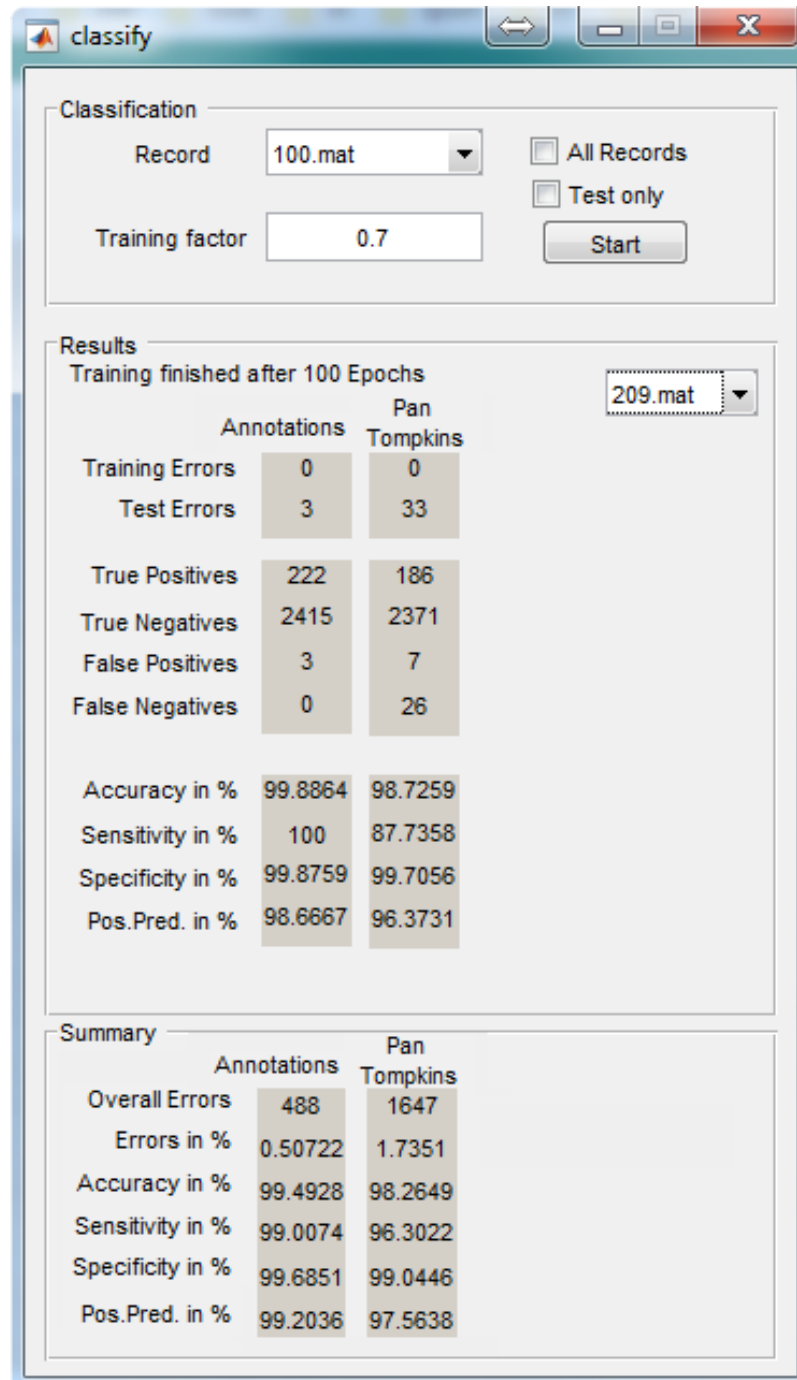


Figure 4.4: MATLAB graphical user interface

Preprocessing is based on the implementation in chapter 3. Hence, the feature vector consists of the R-R interval, arrhythmia parameter and  $n$  principal components. In addition to the input vectors, the desired outputs are transmitted. Training and testing can be performed based on the annotations or the beats detected by Pan Tompkins algorithm. To ensure reliable data transfer, the feature vectors are numbered serially, allowing the receiving part of the Zedboard to control for packet loss and other transmission errors. For the final TCP/IP transfer, 10 vectors are merged into one package before sending. With a total of 8 inputs, 2 outputs and 1 vector number in 32 bit resolution the size of one data package is  $(8 + 2 + 1) * 10 * 4 = 440$  Bytes. Additional inputs increase the size of the data packages by 40 Bytes. Before transfer of the testing data, the overall amount of datasets, the number of inputs and training-rate are sent to a fixed IP on port 7. During the sending process of ECG data, the *MATLAB* script waits for a response of the neural network after every sent package. This signals a successful transmission and the next package can be sent. This process ensures correct order of the data packages and was implemented in addition to TCP retransmission functionality. Although this method leads to slower data rates, the achieved average speed of 80kB/s still meets the requirements and avoids package loss.

After transmission of testing and training data, the script starts sending one ready message per second to the Zedboard, keeping the TCP connection alive. After performed classification, the Results are transmitted back to MATLAB. The response messages contain transmission and classification-statistics:

- Total number of transmitted datasets
- Number of transmission errors
- Number of training epochs
  
- False classifications during training
- False classifications during testing
- False Positives
- False Negatives
- True Positives
- True Negatives

Additional parameters can be easily added. The classification results are automatically displayed in the GUI. Besides the statistics for classification of every record, the user interface also displays the results for all classified results. Additionally, the in section 2.1 presented performance parameters (accuracy, sensitivity, selectivity, and positive predictivity) are computed and displayed.

### 4.3 Processing System

Apart from preprocessing and statistical evaluation, the entire process of training and testing the neural network is controlled by the processing system. As described previously, PS is receiving the ECG data from the PC and interfacing the neural network. To allow programming of the

FPGA, a Vivado Block design has to be created and synthesized. The resulting bitstream can be transferred onto the Zedboard. The block design defines the routing and configurations of all included blocks. In the developed block design, the processing system communicates with the neural network via Advanced eXtensible Interface (AXI) while the ECG data is stored in Block RAM (BRAM). The addresses of the variables stored in BRAM have to be defined in the Address Editor. The block design, illustrated in figure 4.5, contains the following components:

- **ZYNQ7 Processing System** implements the processing system into the block design. The clock of the processing system is set to 100MHz and the outputs are configured. The processing system communicates with the peripheral components via AXI Interconnect.
- **Processing System Reset** connects the processing system reset with the resets of the AXI Interface and further components.
- **AXI Interconnect** multiplexes the communication of the Zynq with further AXI components.
- **Artificial Neural Network** receives the control signals and weight and bias-updates via AXI-lite. The inputs and outputs of the neural network are stored in BRAM.
- **AXI BRAM Controller** controls the data transfer between the processing system and BRAM.
- **Block Memory** stores the inputs and output data of the artificial neural network.

Before synthesis the all modules are merged with a vdh1 or verilog wrapper into one module. Vivado creates this wrapper file automatically. To allow fast update of the neural network, TCL scripts allow to automate re-routing and updating of all components. Drawback of deployment of the Zynq processing system is its high maximum power consumption of 1.529 W.

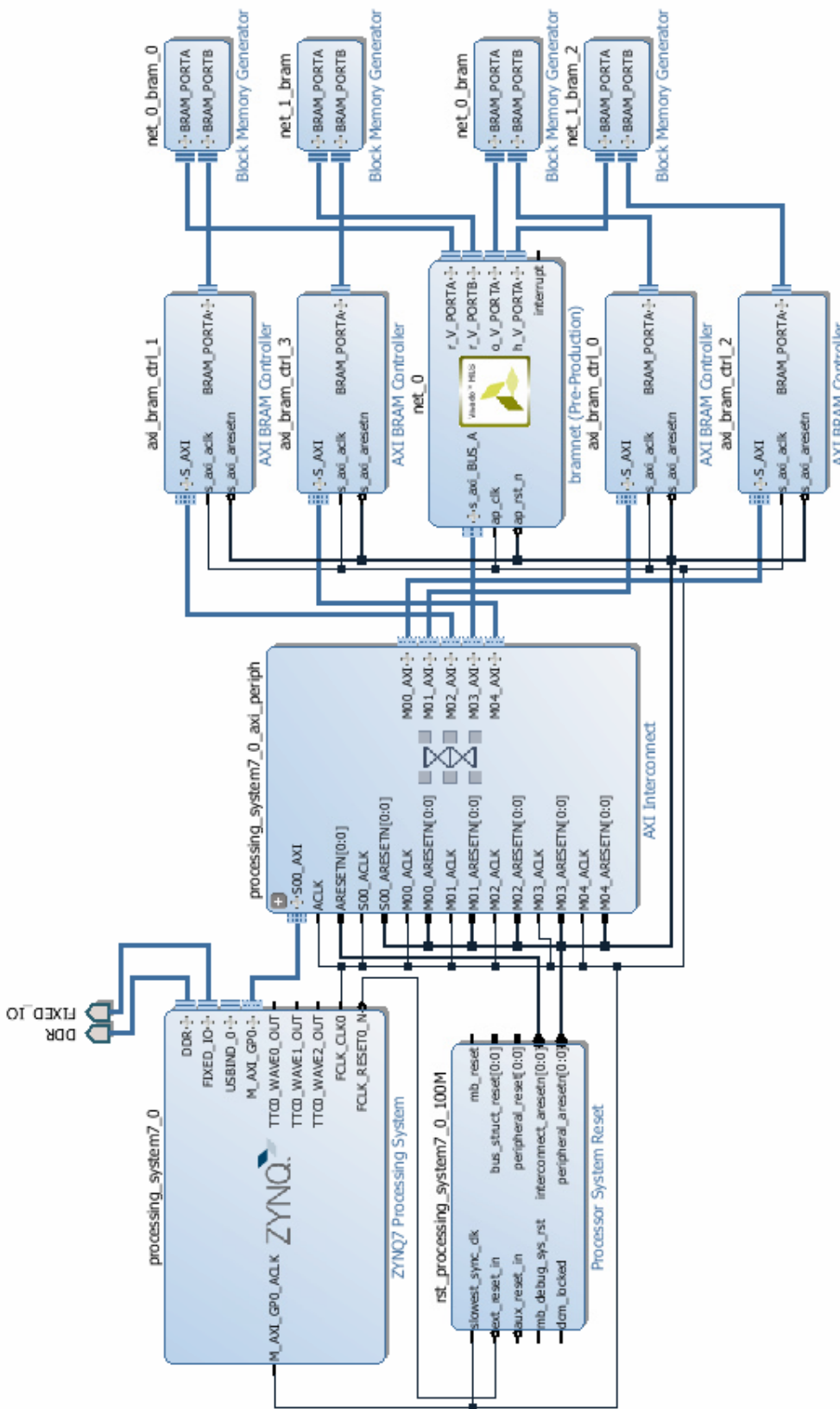


Figure 4.5: Vivado Block design: Processing system controls training and testing of the neural network via AXI-lite interface, input and output data is exchanged via BRAM.

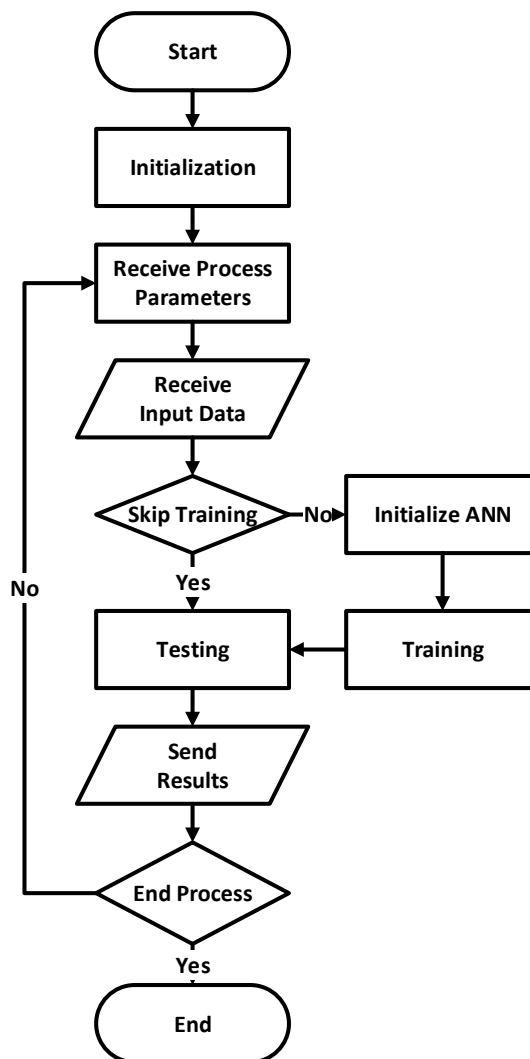


### 4.3.1 Software Architecture

After successful synthesis and implementation, the exported bit stream of the block design can be exported to Vivado SDK. This automatically invokes Vivado SDK and creates a so called board support package, containing the functions to interface the implemented hardware modules. During classification, the Zynq Processing System has four main functions:

- Controlling the TCP/IP Ethernet Interface (light weight IP stack)
- Interfacing the Artificial Neural Network
- Controlling training and testing

Figure 4.6 shows the flowchart of the ARM Software. Classification is controlled by the ARM



**Figure 4.6:** The ARM software consists of one main loop executing all the tasks for every record

software, based on the parameters and data received from MATLAB. As the commands for starting and ending the classification process are transmitted via Ethernet, the light weight IP stack (lwIP) has to be initialized. After initialization, parameters for the classification process are received. Next the data to classify is transferred alongside with the desired results. The received feature vectors are stored in arrays directly in BRAM. Before training is performed, the artificial neural network is initialized with random weights and biases. If desired the training can be skipped. This can be useful when comparing the performance of classification based on annotations and beat detection algorithms. For each epoch of the training phase the weights and biases are previously updated via the AXI-lite interface. Before invoking the neural network function, the input data is written into BRAM. After successful training, the weights and biases are stored and used for testing for the entire dataset. Lastly the results of the classification process are displayed via serial console and transmitted to the PC. The loop end here and a further record can be classified. Alternatively the connection can be shut down and the program end.

### 4.3.2 Communication via Lightweight IP

For the implementation of the TCP/IP protocol stack, lwIP was used. lwIP focuses on a reduction of memory usage and code size, making it a suitable choice for embedded systems [Dun01]. To limit the package size of the TCP packages and avoid retransmissions, the data was packaged already in MATLAB and sent one by one. After each sent package, the MATLAB script waits for an answer of the ARM processor, signalling that the previous package was received correctly. Although this mechanism slows down the data rate of the protocol to around 80kB/s streaming of two lead ECG data would be still possible, as one lead of ECG data contains only 495 bytes per second <sup>2</sup>. Therefore the data rate would not hinder implementation of the preprocessing algorithms in the ARM core.

The lwIP stack is initialized at the beginning of the program with a fixed IP-address (192.168.1.10). Executing the function `start_application()` the lwIP starts listening to TCP connections on port 7. Successful execution of the function terminates the initialization.

Before sending the ECG features, the number the amount of features, number of heartbeats and percentage of beats used for training are transmitted via Ethernet. Additionally the amount of hidden layers used for classification and further parameters can be send. Afterwards the input data for the artificial neural network is sent by the MATLAB script. On the receiving side the correct transmission is checked and in case of success, a response message sent back to the PC. Otherwise the response message signals that the data transmission has to be started again. Implementing this method the data was transmitted correctly with 100% success rate.

The receiving of TCP data is handled by a callback function bound to the receiving port 7. This function is invoked every time lwIP signals a received TCP package. The program enters a while loop that is left if the global state of the software is changed directly by the callback function. The callback function itself reads the input data and writes it into global variables. These variables are only set once for every received ECG record.

After training and testing the artificial neural network, lwIP is used again to transfer the classification results back to the PC. Therefore the callback function is invoked by the requests for result data received from MATLAB. The results are transmitted as one package. After successful data transmission the program displays a summary of the data transfer:

---

<sup>2</sup>11-bit resolution, 360 samples/s

```

TCP receiving @ port 7

-----HEADER-----
DATAWIDTH: 10
DATALENGTH: 2220
TRAINING FACTOR: 0.7

-----RECEIVING-----
Data Read took 41.2 s.
Speed: 18.5 kB/s

----TRANSMISSION----
Length: 2220
Correct: 2220
Errors: 0

```

**Listing 4.1:** Example summary of the data transmission

### 4.3.3 Interfacing ANN

Interfacing the artificial neural network is the most frequently executed task of the processing system. Therefore two efficient methods to transfer data onto the neural network were studied. Before data transfer can be started, the neural network has to be initialized. The functions for initialization and configuration are provided by the board support package. After instantiating variables with the by the board support package introduced data types **XNet** and **XNet\_Config**, the newly created variables can be connected to the hardware-implemented ANN. The board support package automatically provides the necessary device-ID as a constant. After successful initialization, a transition to the ready state is initiated.

There are two ways for data exchange between the neural network and the ARM core. Due to the selected training algorithm the most frequently performed instructions are: data input, execution of the neural network and reading output data. To ensure fast execution of these three steps, the inputs and outputs of the neural networks are stored in BRAM and can be directly accessed by the ARM processor. The update of weights and biases of the neural network is performed via AXI-lite, as this form of data transfer reduces the required resources since for the direct access of BRAM additional BRAM controllers are necessary.

The translation from floating point to the desired fixed point data type is executed directly before data transfer. The conversion is performed by two macros:

```

#define FLOAT2FIXED(x) ((int)((x) * (1 << FRACT_BITS_IN)))
#define FIXED2FLOAT(x) (((float)(x)) / (1 << FRACT_BITS_OUT))

```

**Listing 4.2:** Conversion from floating point to fixed point and vice versa

The measured latencies for transmission of 32 bit via AXI-lite and BRAM are  $0.72\mu s$  and  $0.507\mu s$ , respectively. To write data into the previously defined BRAM regions, variables pointing to the addresses are defined and accessed. Data transfer via AXI-lite is performed with functions defined by the board support package. Input parameters are a pointer to the instance of the neural

network, the offset, pointer to the data to transfer, and the data width. Both methods are shown in listing 4.3.

```
int *input = (int *)0x46000000;input[0] = 1;           // BRAM
XNet_Write_weights_V_Words(&Net,0,(int *)w,NUM_W);    // AXI-lite
```

**Listing 4.3:** Direct BRAM access

#### 4.3.4 Training and Testing

The neural network is trained with resilient backpropagation. Implementation of the in chapter 3 presented RPROP algorithm [RB93] is based on the implementation in .NET found on <https://visualstudiomagazine.com/Articles/2015/03/01/Resilient-Back-Propagation.aspx>. As previously described, in RPROP, backpropagation is performed based on sign change of the gradients. Therefore, for every weight and bias the current gradient has to be stored. To increase the step size close to minima, the algorithm makes use of delta values, defining the step size, that can be increased in case of no sign change. The delta values replace the fixed learning rate from other gradient descent algorithms. Delta values are separately stored for each weight and bias. Listing 4.4 shows the RPROP algorithm in pseudo code.

```
//Initialization
prev gradient = 0
prev delta = 0.1

while (epoch < maxEpochs && mean squar error > minError)
{
//compute gradients
  for each feature vector of training set {
    compute and accumulate gradient terms
  }

//update weights and biases
  for each weight (and bias) {
    if (no sign change for gradient) {
      increase delta
      update weight with increased delta
    } else if sign change for gradient {
      decrease delta
      reset weight to previous value
    }
    prev delta = new delta
    prev gradient = curr gradient
  }
  epoch++
}
return weights and bias values
```

**Listing 4.4:** The functionality of RPROP explained in pseudo code

For training all delta values are initialized with 0.1 and previous gradient values with 0. After initialization the algorithm enters a loop. The number training epochs is limited by the value **maxEpochs**. For the computation of gradients, the derivatives are approximated with  $f(x) = (1 - x) * (1 + x)$  for hidden layer neurons and  $g(x) = \frac{(1-x)*(1+x)}{2}$  for output layer neurons. For the backpropagation process, the hidden layer outputs are required. Therefore outputs of the hidden layer neurons are as well stored in BRAM. The calculated gradients are accumulated for the entire training set. Core of the RPROP algorithm is the weight update process. For each weight and bias, the gradients are checked for sign change. In case the newly computed gradient has the same sign as the previous gradient, the delta value for the gradient is increased and the weigh/bias is updated with the new value. New delta value is calculated by multiplication with 1.2 and limited to a maximum of 50. If the previous weight update led to a change of sign, the delta value is halved. The minimum values for delta is defined as  $10^{-6}$ . The successful update of all weights and biases concludes one training epoch. After exceeding the limit of maximal epochs. To avoid overfitting, after 100 training epochs a mean square error lower than  $10^{-6}$  also leads to end of training. For testing, the trained neural network performs classification for the entire set of feature vectors. After testing and training a summary of the classification results is displayed in the serial console (see listing 4.5).

```

-----TRAINING-----
Epochs:                100
Missclassifications:    0
Mean Squared Error:    0.000000
Training length:       1589

-----TEST-----
Missclassifications:    0
Mean Squared Error:    0.000624
Test length:           681

TP: 34
TN: 2236
FP: 0
FN: 0

```

**Listing 4.5:** Classification summary for record 100 of MIT-BIH arrhythmia database

## 4.4 Neural Network FPGA Implementation

Based on the studies performed in the previous chapters, a neural network is implemented. The hardware model is created with high level synthesis in Vivado HLS and exported as intellectual property package. Vivado HLS offers testing of the RTL-design alongside with C implementations and therefore allows to compare precision and results of hardware and software implementations. Furthermore, the use of high level synthesis offers flexibility during the design process, as most parameters can be varied easily. Activation function, fixed point precision and network size influence not only classification accuracy of artificial neural network but also the amount of required resources.

#### 4.4.1 Code Structure

The code structure of the neural network, shown in listing 4.6, can be divided into three steps: initialization, computation of the hidden layer outputs and calculation of the final neural network outputs. During Initialization, the interface types of inputs and outputs are defined. Computation of hidden layer outputs is performed by multiplication of the feature vector with the weight matrix. The biases are added to the resulting vector and the selected activation function is applied on each neuron. The similar process is performed for computation of the output layer outputs.

```
//Initialization
define interfaces

//Hidden Layer
for number of hidden layer neurons
{
  for number of inputs
  {
    multiply weight*input, accumulate in hidden_out vector
  }
  add biases to hidden_out
  apply activation function on hidden_out values
}

//Output Layer
for number of output neurons
{
  out[i] = 0;
  for number of hidden layer neurons
  {
    multiply weight*hidden_out, accumulate in out vector
  }
  add biases to out
  apply activation function on out values
}
```

**Listing 4.6:** Pseudo code of neural network

For high level synthesis, the neural network has to be implemented as a function in c. The particular function to synthesize is called by a testbench that contains the main function of the c program. The implementation of the neural network makes use of several constants that are defined in a header file containing information about size of the neural network and precision of the selected data type. Variation of these constants results in a different size of the neural network.

For best results of high level synthesis the code of the function to synthesize has to be optimized. In *Vivado HLS*, the hardware synthesis of the c code can be influenced by **pragmas**. Pragmas can be used to add additional information to the code. These pragmas can contain for example information on the type of interfaces to be implemented, or enrolling and pipelining of loops.

#### 4.4.2 Interfaces

Pragmas are used to define the interface for a function parameter. The pragmas for creation of the interfaces are definitions set at the beginning of the function. Listing 4.7 shows the creation of two interfaces for an example function.

```
void example(int ex_input, int ex_output)
{
    #pragma HLS INTERFACE s_axilite port=ex_input bundle=BUS_A
    #pragma HLS INTERFACE bram port=ex_output

    ex_output = ex_input + 5;
}
```

**Listing 4.7:** Example function with AXI-lite input and BRAM output

Based on the measurements in section 4.3.3 the types of inputs and outputs are selected for the neural network. Table 4.1 shows the inputs and outputs of the neural network alongside with the type of interface.

**Table 4.1:** Parameters for the artificial neural network function

Name	Interface	Size	Comment
input	BRAM	NUM_IN	Feature Vector
w1	AXI-lite	NUM_IN x NUM_HIDDEN	Input to Hidden Layer Weights
b1	AXI-lite	NUM_HIDDEN	Hidden Layer Biases
hidden	BRAM	NUM_HIDDEN	Hidden Layer Outputs
w2	AXI-lite	NUM_HIDDEN x NUM_OUT	Hidden to Output Layer Weights
b2	AXI-lite	NUM_OUT	Hidden Layer Biases
out	BRAM	NUM_OUT	Neural Network Outputs

#### 4.4.3 Pipelining

To reduce latency of the neural network, pipelining is enabled for both, hidden and output layer loops. The loops are pipelined by inserting the pragma *#pragma HLS PIPELINE* inside the for loop. For a neural network with eight input neurons and six hidden layer neurons, the latency was decrease by a factor of 3 by implementing pipelining. Drawback is that the number of DSP increases. While without pipelining the neural network requires only two DSPs, with the option enabled, one DSP for each hidden and output layer neuron is necessary. Therefore it can be said that when enabling pipelining in HLS additional components can be necessary to optimize the degree of capacity utilization of other resources. The degree of pipelining should be chosen based on the primary goal of the implementation (minimize latency or reduce power consumption)

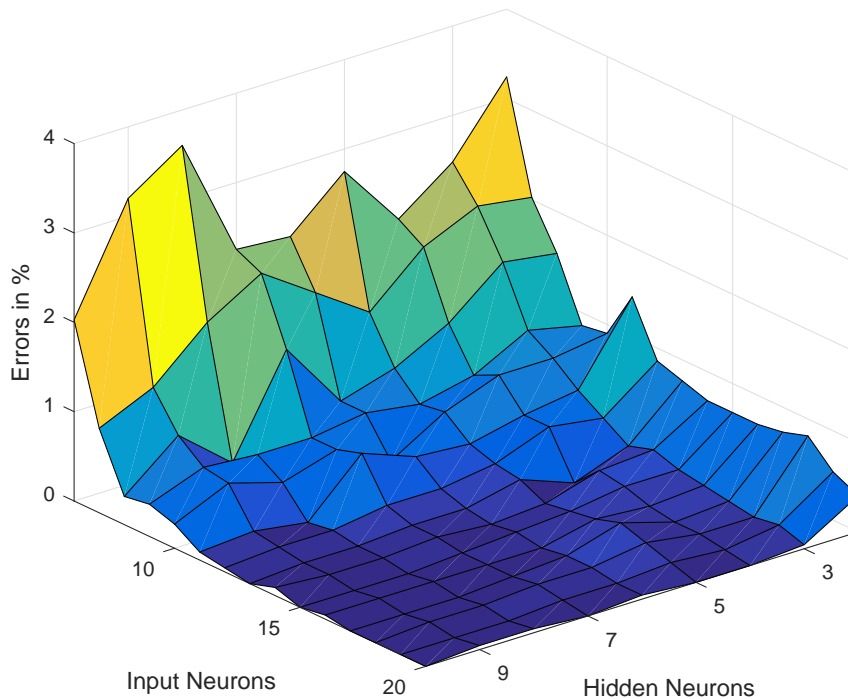
#### 4.4.4 Fixed Point Datatypes

To optimize the hardware implementation of the ANN, an extensive analysis of implementation with fixed point data types has been performed by varying the number of input neurons, hidden layers, and the bit width of the fixed point implementation. For implementation of fixed point

designs Vivado HLS offers a fixed point library. To make use of it, the *ap-fixed* library has to be included in the header file. Consequently fixed point data types can be defined. The command `typedef ap_fixed<32,16> fix32` is a type definition that allows to make use of the newly created `fix32` datatype. The first number in brackets sets the bitwidth of the data type, the second number defines the amounts of fraction bits. Different fixed point implementations were compared in terms of resulting accuracy and required resources.

### Influence on Accuracy

The amount of fraction bits for fixed point data was set to half of the bit width of the selected data type. Figures 4.7 and 4.8 illustrate the amount of false classifications for record 104 of the MIT-BIH database as functions of three parameters. This particular record was selected as it represents a task with typical complexity and allows a good demonstration of how classification accuracy is influenced by change of parameters. However, for some records 100% classification accuracy is not achievable. Training the fixed point implementation of neural network, the training algorithm adapts weights and bias according to the selected precision. Whereas training the neural network in a floating point implementation and transferring weights and biases later onto a fixed point implementation increases false classifications, since weights and biases are trained for floating point precision. In figure 4.7 it can be seen that independent of the amount of inputs, more than three hidden layer neurons do not increase accuracy. For input neurons this boundary lies between ten and thirteen, depending on the amount of hidden layer neurons. For figure 4.8 the number of

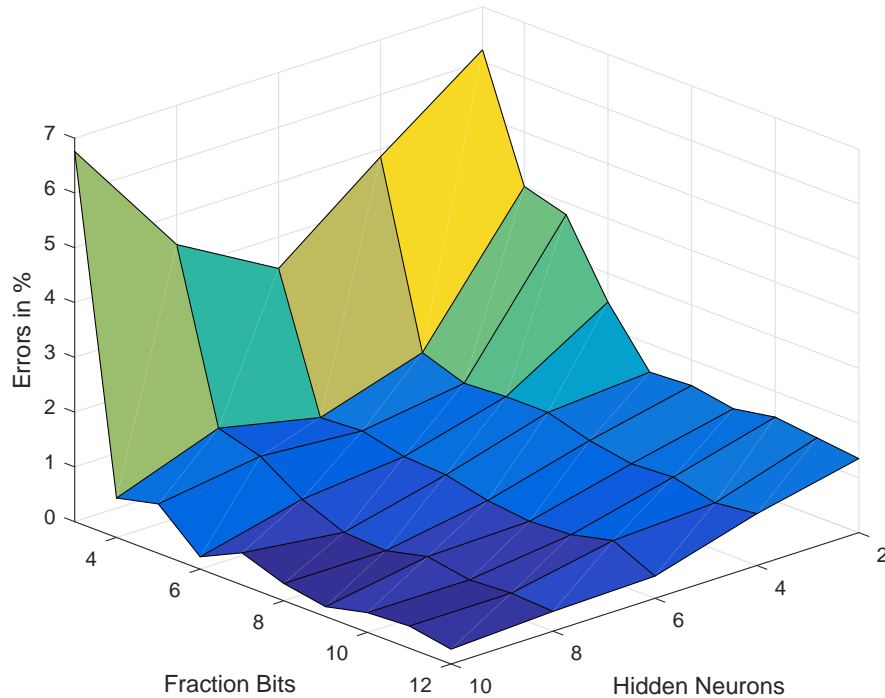


**Figure 4.7:** Accuracy with 16-bit data (8 Fraction bits) for record 104

input neurons was set to eight. It shows that for the selected number of inputs a precision of seven



fraction bits and eight neurons in the hidden layer already achieve 100% accuracy. For a higher number of inputs the required amount of neurons in the hidden layer and bit width decreases, while the neural network still achieves the same classification accuracy. It can be concluded,



**Figure 4.8:** Accuracy for 8 input neurons for record 104 MIT-BIH

that for every number of hidden layer neurons there are certain thresholds for minimum precision and amount of inputs. Above the thresholds, increasing the number of input neurons improves the accuracy more than increasing the precision of the data type. For the selected record the smallest neural network with 100% accuracy consists of eight input neurons, eight hidden layer neurons with 14 bit fixed point data width. The optimal configuration for classification of the entire database has to be determined separately

### Required Resources

With respect to accuracy, the amount of required resources have a higher dependence on optimal implementation during High Level Synthesis. Table 4.2 shows the synthesis results for four different configured neural networks implemented with three different data types. In comparison to floating point implementation, it is obvious that the proposed optimizations lead to a significant reduction of required resources and latency. For ANN with each one input, hidden and output-layer, concerning the different parameters it can be noted, that for bit widths of 12 and 16, Vivado HLS assigns one DSPs for each input and hidden neuron. For 24 bit width two DSPs are required for every input and hidden-layer neuron. When increasing the amount of hidden layers from four to six, Vivado HLS changes the routing and usage of DSPs, therefore the amount of used Flip-Flops and LUTs is reduced. Drawback is that the maximum frequency of the module

decreases, as the synthesis reports in Vivado HLS have shown. In general the required resources increase with the number of input and hidden layer neurons.

**Table 4.2:** Use of resources for different configurations

Floating Point					
Input/Hidden	DSPs	FFs	LUTs	Latency	Accuracy
8/6	42	9295	15163	1208	99.81%
12-bit Fixed Point					
Input/Hidden	DSPs	FFs	LUTs	Latency	Accuracy
8/4	12	1729	3945	62	98.56%
8/6	14	1551	1958	85	99.14%
10/4	14	1977	4399	71	99.28%
10/6	16	1613	1963	97	99.64%
16-bit Fixed Point					
Input/Hidden	DSPs	FFs	LUTs	Latency	Accuracy
8/4	12	1801	3653	60	98.87%
8/6	14	1561	1703	83	99.37%
10/4	14	2017	4045	68	99.19%
10/6	16	1646	1708	95	99.77%
24-bit Fixed Point					
Input/Hidden	DSPs	FFs	LUTs	Latency	Accuracy
8/4	24	2056	3910	63	99.05%
8/6	28	1772	1895	87	99.59%
10/4	28	2280	4366	71	99.28%
10/6	32	1926	1932	99	<b>100%</b>

Summed up it can be said, that the amount of input and hidden layer neurons alongside with precision of the selected data type influence ECG classification accuracy of neural networks. While increasing the data width does not significantly increase the amount of required resources, a higher number of input and hidden layer neurons requires more LUTs and FIFOs for FPGA implementation. With respect to previously simulated accuracies it can be concluded that only after increasing data precision, the number of input and hidden layer neurons should be increased.

#### 4.4.5 Activation Functions

For artificial neural networks, several activation functions are available. Choosing the best fitting activation function, secures the best result for the problem. In classification problems the sigmoid activation function for hidden layer neurons and the softmax function [DJ01] for output neurons are commonly used. These functions also meet all the requirements for classifying heartbeats.

#### Hidden Layer

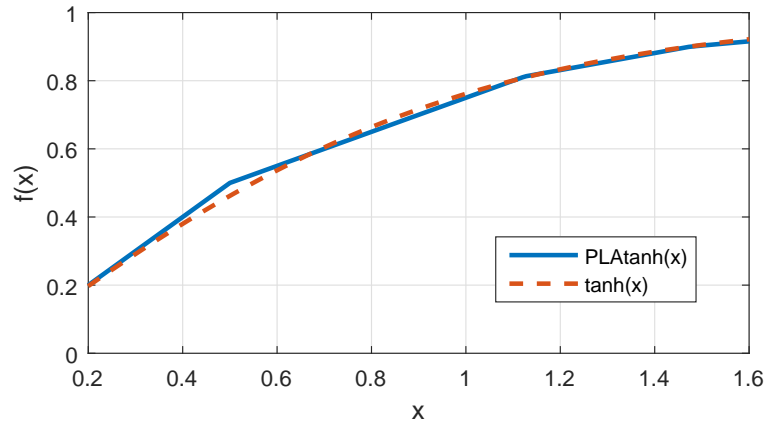
In neural networks, the hidden layer activation function is one of the most frequently used functions, and therefore a reduction of computation time was desired for hardware implementation. As shown in [ACHG97], Piecewise Linear Approximation (PLA) performs superior to exchanging the desired tansig activation function with logsig or ramp characteristics. Considering that the

final implementation was planned with fixed point arithmetic, piecewise linear approximation was performed with gradients that reduced the required multiplications to simple shift operations [Hik03]. While previously mentioned works ([Hik03] and [ACHG97]) used a maximum of seven ranges, to reduce the error of the approximated function symmetrical function shown in equation 4.1 is proposed. This leads to a better approximation of the function and a better overall performance of the neural network compared to other proposed implementations. The piecewise linearly approximated hyperbolic tangent function (PLAtanh) is defined in 4.1, with the borders given in 4.2

$$PLAtanh(x) = \begin{cases} 1 & x \geq a \\ x/4096 + 0.9986377 & a \geq x > b \\ x/32 + 0.905 & b \geq x > c \\ x/8 + 0.715625 & c \geq x > d \\ x/4 + 0.53125 & d \geq x > e \\ x/2 + 0.25 & e \geq x > f \\ x & f \geq x > g \\ x/2 - 0.25 & g \geq x > h \\ x/4 - 0.53125 & h \geq x > i \\ x/8 - 0.715625 & i \geq x > j \\ x/32 - 0.905 & j \geq x > k \\ x/4096 - 0.9986377 & k \geq x > l \\ -1 & l \geq x \end{cases} \quad (4.1)$$

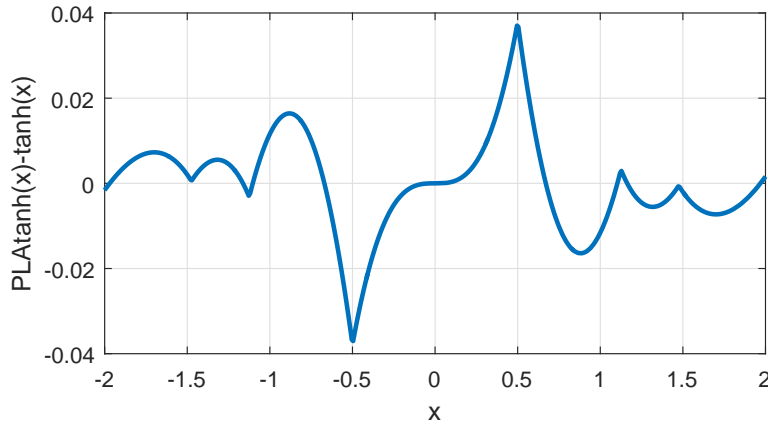
$$\begin{aligned} a &= 5.5799959, b = 3.02, c = 2.02, d = 1.475, e = 1.125, \\ f &= 0.5, g = -0.5, h = -1.125, i = -1.475, \\ j &= -2.02, k = -3.02, l = -5.5799959 \end{aligned} \quad (4.2)$$

Figure 4.9 illustrates the piecewise linearly approximated function in comparison with the hyperbolic tangent function.



**Figure 4.9:** Piecewise linear approximation of tanh

The error of the newly defined function is illustrated in figure 4.10. Due to the selected restrictions on the gradient of the linear functions, the biggest errors occur at  $x = 0.5$  and  $x = 0.5$  with approximately 0.03788 and -0.03788, respectively.



**Figure 4.10:** The biggest errors of PLAtanh in comparison to tanh occur at  $x = 0.5$  and  $x = -0.5$

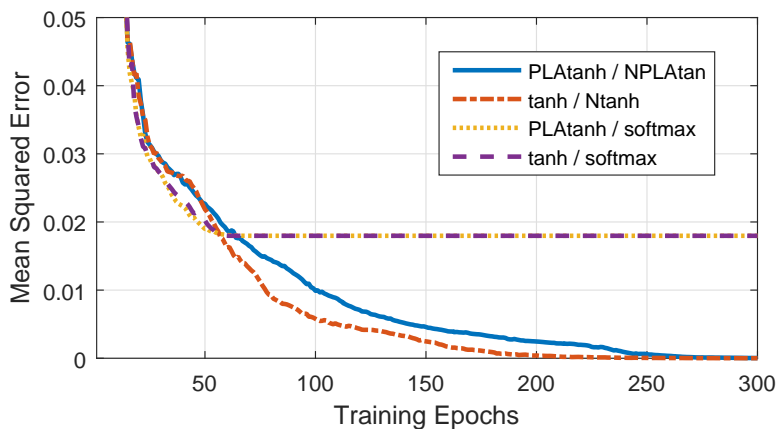
### Output Layer

To normalize output layer results, the hyperbolic tangent function is used with

$$Ntanh(x) = \frac{tanh(x) + 1}{2} \tag{4.3}$$

An implementation in this form not only allows to also make use of the already implemented simplified hyperbolic tangent function, but also reduces complexity.

To evaluate the performance of this activation function in comparison to the exact hyperbolic tangent function, several neural networks were trained once with exact functions and once with piecewise linear approximations, in both hidden and output layers. As a typical example Figure 4.11 shows the mean square error for one specific record in the database for different activation functions. The error was logged after every training iteration with RPROP algorithm. The figure shows that using the piecewise linear approximated activation function for hidden layer neurons, does not worsen the results in comparison to exact implementation. Replacing the softmax func-



**Figure 4.11:** Mean square error for training with different activation functions

tion with Ntanh, given in 4.3, as activation function for output layer neurons, leads to faster convergence during the training phase and to better fitting due to the small number of output

layer neurons. This difference occurs, because in backpropagation the influence of all output signals on the results of the softmax function increases the complexity of the algorithm and was therefore simplified in the application.

### Reduction of Required Resources

To evaluate the advantages of the newly defined function, both, the exact function and the approximation were implemented with Vivado HLS. For PLAtanh, equation 4.1 is implemented as series of if statements, each defining the linear function for one range (listing 4.8).

```

fix32 PLAtanh(fix32 x)
{
  if (x > 5.579995904)          //Range 13
    return 1;
  else if (x < -5.579995904)    //Range 1
    return -1;
  else if (x > 3.02)            //Range 12
    return x/4096+0.9986376963125;
  else if (x < -3.02)          //Range 2
    return x/4096-0.9986376963125;
  .
  .
  else                          //Range 7
    return x
}

```

**Listing 4.8:** Example function with axi-lite input and bram output

For floating point implementation, tanh can be rewritten as shown in equation 4.4. This is necessary as Vivado HLS math library does not offer a synthesizable tanh function.

$$\tanh x = \frac{\sinh x}{\cosh x} = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{e^{2x} - 1}{e^{2x} + 1} = 1 - \frac{2}{e^{2x} + 1} \quad (4.4)$$

As previously mentioned, the selection of gradients also influences size and latency of the resulting module. Choosing gradients  $grad = 2^n$  with  $n \in \mathbb{Z}$  reduces latency to a just two clock cycles, as the multiplication is reduced to a simple shift operation. Table 4.3 shows the required resources for the proposed implementation in comparison to floating point tanh implementation.

**Table 4.3:** Reduction of required resources of tanh, due to fixed point implementation

	DSPs	Flip-Flops	LUTs	Clock Cycles
tanh (Floating Point)	12	1553	2767	39
PLAtanh (Fixed Point)	0	183	705	2
<b>Reduction</b>	100%	88.2%	74.5%	94.8%

Piecewise linear approximation and fixed point implementation reduces the amount of necessary DSPs to zero. Additionally, the amount of required Flip-Flops and LUTs is reduced by 88.2%

and 74.5%, respectively. Alongside with the reduction of resources, the delay of the activation function is reduced from 39 to 2 clock cycles. This reduction is an important factor for efficient implementation of the artificial neural network, as the activation function is necessarily part of every neuron. Consequently, for the experiments PLAtanh and Ntanh were used as activation functions for the hidden layers and the output layer, respectively.

## 4.5 Results

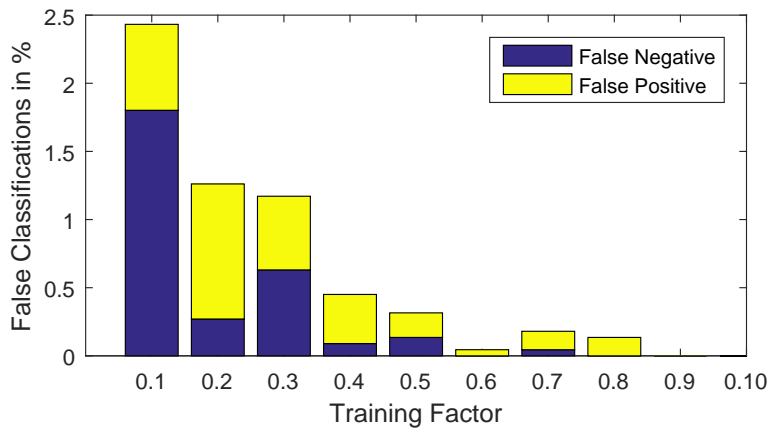
Based on the previous analysis, an artificial network with twelve input layer neurons, eight hidden layer neurons and two outputs was implemented. The employed activation functions are PLAtanh and Ntanh for the hidden layer and the output layer, respectively. The neural network is trained with resilient backpropagation. The following sections discuss the performance of the implemented classification system.

### 4.5.1 Classification Accuracy

The amount of employed neurons and the precision of the selected datatype influence the classification accuracy of the neural network. For the implementation a neural network with twelve input neurons and eight hidden layer neurons were selected. The data width was set to 32 bit with 16 fraction bits. These parameters were selected based on the prior trade-off analysis.

#### Training Factor

In supervised learning the neural network is only trained on one part of the dataset, while the remaining data is used for testing. The factor that determines the percentage of data used for training is called training factor. While a small training factor might result in overfitting, a training factor close to 100% does not represent a good metric since the neural network is already trained on a large part of the data. Figure 4.12 shows the resulting classification errors in % as a function of the training factor. For the further results a training factor of 70.0% was selected.



**Figure 4.12:** Classification errors in % for record 104 of MIT-BIH arrhythmia database

It can be noticed, that increasing the training factor decreases the amount of classification errors, as the neural network is trained on a larger training data set, and therefore a higher level of generalisation is achieved. Drawback is, that a higher training factor equals a higher amount of manually performed ECG classification.

### Classifying MIT-BIH Arrhythmia Database

Controlled by the graphical user interface, classification for the MIT-BIH arrhythmia database was performed. For each record the features vectors derived from the annotation files and the with Pan Tompkins algorithm detected beats were classified. For the Pan Tompkins based classification false positives were prior to the classification process eliminated to reduce the dependency on the beat detection process. Table 4.4 shows accuracy, specificity, sensitivity and positive predictivity for both datasets.

**Table 4.4:** Performance parameters for classification based on annotations and Pan Tompkins algorithm

	<b>Annotations</b>	<b>Pan Tompkins</b>
Accuracy	99.4817%	99.5185%
Specificity	99.7558%	99.7521%
Sensitivity	98.8285%	98.9237%
Positive Predictivity	99.3834%	99.3656%

Classification of the Pan Tompkins based dataset is performed with the same accuracy as for the feature vectors derived from the annotation files. Table 4.5 includes the incorrectly detected and non-detected beats into the statistics. Thereby every false detection and non-detection is rated as a classification error.

**Table 4.5:** Resulting accuracies after including false detections

	<b>Annotations</b>	<b>Pan Tompkins</b>
Beats to Classify	96210	94920
False Classifications	489	457
False Detections	0	1911
Accuracy	<b>99.4817%</b>	<b>97.5545%</b>

When making use of the included annotation files, the presented algorithm performs with 99.5% classification accuracy on the MIT-BIH arrhythmia database. Employing Pan Tompkins algorithm for beat detection the systems classification accuracy drops to 97.5% when including false detections. The drop of classification accuracy due to Pan Tompkins algorithm is induced by the not detected R peaks. Other existing systems classify with accuracies from around 85% to 99.9%.

#### 4.5.2 Required Resources and Power Consumption

To reduce power consumption of the classification system, an artificial neural network was implemented on FPGA. The numbers of required resources for the final implementation are listed in table 4.6. The two components using most of FFs and LUTs are the artificial neural network and the AXI-Periphery for the processing system.

**Table 4.6:** Required resources of the block design, ANN and AXI-periphery are the two main components

	DSPs	Flip-Flops	LUTs
Block Design	80	5783	3898
Artificial Neural Network	80	3198	1671
Processing System AXI-Periphery	0	1604	1267

The resulting, by Vivado estimated, maximal power consumption lies at 1.851W with 0.162W static load. About 91.2% of the maximal dynamic load of the system is used by the processing system. Maximum power consumption of the processing system and the neural network are 1.529W and 0.124W respectively.

As the largest part of power is consumed by the processing system additional power reduction could be achieved by including the training algorithm into the programmable logic, while also reducing the amount of required periphery components. As a final step the implementation of preprocessing on FPGA could minimize power consumption.

### 4.5.3 Time Measurements

For time measurements, the classification process can be divided into three steps: data transmission, training, testing. The durations of all three phases depend on the chosen parameters. Table 4.7 shows the measured time for classification with the implemented 12-8-2 32 bit neural network and training for 70.0% of the data set for 100 epochs.

**Table 4.7:** Duration data transmission, training and testing-phase

Phase	Measured Time
Data Transmission	7.2s
Training	2.7s
Testing	32ms
Total	9.9s

During training phase, for every epoch three steps are executed:

- Weight and bias update is performed via AXI-lite.
- Data input, neural network execution and data output is executed for the entire test dataset.
- Resilient backpropagation algorithm computes the new weights and biases.

Duration of training mainly depend on the number of necessary epochs to reach a low mean squared error. The number of epochs varies from 100 to 4000.



Table 4.8 shows the time measurements for one epoch of training. The actual classification process takes about 19ms while update and computation of weights and biases are performed in several microseconds.

**Table 4.8:** Time measurements in one epoch of training

Step	Measured Time
Weight and Bias Update	27 $\mu$ s
Data Input ANN Execution Data Output	18.425ms
Weights and Biases Computation	47 $\mu$ s
Total	18.507ms

In testing for every feature vector the input data is written into the BRAM, the neural network function is executed and lastly the output data is read from BRAM. Table 4.9 shows the measured times of the three steps for one feature vector. The duration of testing depends on the size of the dataset.

**Table 4.9:** Time measurements of the steps for the classification of one feature vector

Step	Measured Time
Input	5.904 $\mu$ s
Execution	2.331 $\mu$ s
Output	0.984 $\mu$ s
Total	9.214 $\mu$ s

It can be noticed that the data input takes longer than the actual execution of the artificial neural network.

## 5 Conclusion

A FPGA-accelerated neural network-based ECG anomaly detection system was presented. In the classification process at the first stage heart beats are detected with Pan Tompkins algorithm. To correctly detect anomalies additional features are extracted creating a feature vector for each heartbeat. Dimension of the feature vector is reduced with principal component analysis before performing the classification with a multi-layer perceptron.

For beat detection Pan Tompkins algorithm proved more accurate than wavelet transform based beat detection. The advantage of wavelet transform lies in the extraction of additional features. The best feature set consisted of 180 samples around the R peak, and two parameters describing the interval length to the preceding and succeeding heartbeat.

In the second part, to reduce power consumption of the system, FPGA implementation of the multi-layer perceptron was proposed. Therefore the effects of fixed point implementation and approximation of activation functions on the required resources and accuracy of the neural network were studied. In the proposed architecture preprocessing is performed on a PC platform while training and testing of the neural network, implemented in the programmable logic, is performed by an ARM processor.

The trade-off analysis showed that fixed point implementation effectively reduces the amount of required resources and latency of the neural network without severely influencing classification accuracy. Higher data type precision and additional inputs increase classification accuracy of the neural network, but 100% accurate classification cannot be reached by these measures.

For implementation of activation functions, piecewise liner approximation was successfully performed for the hyperbolic tangent function. While reducing the amount of required resources, implementation of the approximated function does not worsen the classification accuracy of the system.

As the focus of the work was successful implementation of the artificial neural network on FPGA, the preprocessing needs further improvement to increase the accuracy of the entire system. For further reduction of power consumption, beat detection and feature reduction need to be implemented on the ARM processor. Implementation of neural network training and preprocessing in the programmable logic would further decrease the required resources.

## Literature

- [ABI<sup>+</sup>03] ACHARYA, U R. ; BHAT, P S. ; IYENGAR, S S. ; RAO, Ashok ; DUA, Sumeet: Classification of heart rate data using artificial neural network and fuzzy equivalence relation. In: *Pattern Recognition* 36 (2003), Nr. 1, S. 61–68
- [ACHG97] AMIN, Hesham ; CURTIS, K M. ; HAYES-GILL, Barrie R.: Piecewise linear approximation applied to nonlinear function of a neural network. In: *IEE Proceedings-Circuits, Devices and Systems* 144 (1997), Nr. 6, S. 313–317
- [BG97] BOGER, Zvi ; GUTERMAN, Hugo: Knowledge extraction from artificial neural network models. In: *Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on* Bd. 4 IEEE, 1997, S. 3030–3035
- [BL97] BERRY, Michael J. ; LINOFF, Gordon: *Data mining techniques: for marketing, sales, and customer support*. John Wiley & Sons, Inc., 1997
- [Blu92] BLUM, Adam: *Neural Networks in C++: An Object-oriented Framework for Building Connectionist Systems*. New York, NY, USA : John Wiley & Sons, Inc., 1992. – ISBN 0–471–53847–7
- [CEES14] CROCKETT, Louise H. ; ELLIOT, Ross A. ; ENDERWITZ, Martin A. ; STEWART, Robert W.: *The Zynq Book: Embedded Processing with the Arm Cortex-A9 on the Xilinx Zynq-7000 All Programmable Soc*. Strathclyde Academic Media, 2014
- [CÖ07] CEYLAN, Rahime ; ÖZBAY, Yüksel: Comparison of FCM, PCA and WT techniques for classification ECG arrhythmias using artificial neural network. In: *Expert Systems with Applications* 33 (2007), Nr. 2, S. 286–295
- [DCDR04] DE CHAZAL, Philip ; DWYER, Maria O. ; REILLY, Richard B.: Automatic classification of heartbeats using ECG morphology and heartbeat interval features. In: *Biomedical Engineering, IEEE Transactions on* 51 (2004), Nr. 7, S. 1196–1206
- [DJ01] DUCH, Wlodzislaw ; JANKOWSKI, Norbert: Transfer functions: hidden possibilities for better neural networks. In: *ESANN Citeseer*, 2001, S. 81–94
- [DR13] DUBEY, Vichitra ; RICHARIYA, Vineet: A Neural Network Approach for ECG Classification. In: *International Journal of Emerging Technology & Advanced Engineering* 3 (2013)
- [Dun01] DUNKELS, Adam: Design and Implementation of the lwIP TCP/IP Stack. In: *Swedish Institute of Computer Science* 2 (2001), S. 77
- [ESRCF10] ESPIRITU-SANTO-RINCON, Antonio ; CARBAJAL-FERNANDEZ, Cuauhtemoc: ECG feature extraction via waveform segmentation. In: *Electrical Engineering Computing Science and Automatic Control (CCE), 2010 7th International Conference on IEEE*, 2010, S. 250–255

- [GAG<sup>+</sup>00] GOLDBERGER, Ary L. ; AMARAL, Luis A. ; GLASS, Leon ; HAUSDORFF, Jeffrey M. ; IVANOV, Plamen C. ; MARK, Roger G. ; MIETUS, Joseph E. ; MOODY, George B. ; PENG, Chung-Kang ; STANLEY, H E.: Physiobank, physiotoolkit, and physionet components of a new research resource for complex physiologic signals. In: *Circulation* 101 (2000), Nr. 23, S. e215–e220
- [GLG<sup>+</sup>83] GENDELMAN, Howard E. ; LINZER, M ; GABELMAN, M ; SMOLLER, S ; SCHEUER, J: Syncope in a general hospital patient population. Usefulness of the radionuclide brain scan, electroencephalogram, and 24-hour Holter monitor. In: *New York state journal of medicine* 83 (1983), Nr. 11-12, S. 1161–1165
- [HF07] HUSSAIN, Hafizah ; FATT, Lai L.: Efficient ECG signal classification using sparsely connected radial basis function neural network. In: *Proceeding of the 6th WSEAS International Conference on Circuits, Systems, Electronics, Control and Signal Processing* Citeseer, 2007, S. 412–416
- [Hik03] HIKAWA, Hiroomi: A digital hardware pulse-mode neuron with piecewise linear activation function. In: *IEEE Transactions on Neural Networks* 14 (2003), Nr. 5, S. 1028–1037
- [JC10] JEWAJINDA, Yutana ; CHONGSTITVATANA, Prabhas: FPGA-based online-learning using parallel genetic algorithm and neural network for ECG signal classification. In: *Electrical Engineering/Electronics Computer Telecommunications and Information Technology (ECTI-CON), 2010 International Conference on IEEE*, 2010, S. 1050–1054
- [JK07] JIANG, W. ; KONG, S. G.: Block-Based Neural Networks for Personalized ECG Signal Classification, 2007. – ISSN 1045–9227, S. 1750–1761
- [JKP05] JIANG, Wei ; KONG, Seong G. ; PETERSON, Gregory D.: ECG signal classification using block-based neural networks. In: *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on* Bd. 1 IEEE, 2005, S. 326–331
- [JNG10a] JADHAV, Shivajirao M. ; NALBALWAR, Sanjay L. ; GHATOL, Ashok A.: ECG arrhythmia classification using modular neural network model. In: *Biomedical Engineering and Sciences (IECBES), 2010 IEEE EMBS Conference on IEEE*, 2010, S. 62–66
- [JNG10b] JADHAV, Shivajirao M. ; NALBALWAR, Sanjay L. ; GHATOL, Ashok A.: Generalized feedforward neural network based cardiac arrhythmia classification from ecg signal data. In: *Advanced Information Management and Service (IMS), 2010 6th International Conference on IEEE*, 2010, S. 351–356
- [JNG10c] JADHAV, Shivajirao M. ; NALBALWAR, SL ; GHATOL, Ashok: Artificial neural network based cardiac arrhythmia classification using ECG signal data. In: *Electronics and Information Engineering (ICEIE), 2010 International Conference On* Bd. 1 IEEE, 2010, S. V1–228
- [JNG11] JADHAV, Shivajirao M. ; NALBALWAR, Sanjay L. ; GHATOL, Ashok A.: Artificial neural network based cardiac arrhythmia disease diagnosis. In: *Process Automation, Control and Computing (PACC), 2011 International Conference on IEEE*, 2011, S. 1–6
- [Kar12] KARSOLIYA, Saurabh: Approximating number of hidden layer neurons in multiple hidden layer BPNN architecture. In: *International Journal of Engineering Trends and Technology* 3 (2012), Nr. 6, S. 713–717
- [Kiş05] KIŞI, Özgür: Comparison of three back-propagation training algorithms for two case studies. In: *Indian journal of engineering & materials sciences* 12 (2005), Nr.

- 5, S. 434–442
- [MAO<sup>+</sup>04] MARTINEZ, J. P. ; ALMEIDA, R. ; OLMOS, S. ; ROCHA, A. P. ; LAGUNA, P.: A wavelet-based ECG delineator: evaluation on standard databases, 2004. – ISSN 0018–9294, S. 570–581
- [MK10] MOAVENIAN, Majid ; KHORRAMI, Hamid: A qualitative comparison of artificial neural networks and support vector machines in ECG arrhythmias classification. In: *Expert Systems with Applications* 37 (2010), Nr. 4, S. 3088–3093
- [MMAJ<sup>+</sup>13] MOHAMAD, FN ; MEGAT ALI, MSA ; JAHIDIN, AH ; SAAID, MF ; NOOR, MZH: Principal component analysis and arrhythmia recognition using Elman neural network. In: *Control and System Graduate Research Colloquium (ICSGRC), 2013 IEEE 4th IEEE*, 2013, S. 141–146
- [Mor78] MORÉ, Jorge J.: The Levenberg-Marquardt algorithm: implementation and theory. In: *Numerical analysis*. Springer, 1978, S. 105–116
- [ÖCK06] ÖZBAY, Yüksel ; CEYLAN, Rahime ; KARLIK, Bekir: A fuzzy clustering neural network architecture for classification of ECG arrhythmias. In: *Computers in Biology and Medicine* 36 (2006), Nr. 4, S. 376–388
- [ÖD15] ÖZDEMİR, AHMET T. ; DANIŞMAN, KENAN: A comparative study of two different FPGA-based arrhythmia classifier architectures. In: *Turkish Journal of Electrical Engineering & Computer Sciences* 23 (2015), Nr. Sup. 1, S. 2089–2016
- [PB10] PANTELOPOULOS, Alexandros ; BOURBAKIS, Nikolaos G.: A survey on wearable sensor-based systems for health monitoring and prognosis. In: *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on* 40 (2010), Nr. 1, S. 1–12
- [PM10] PAL, Saurabh ; MITRA, Madhuchhanda: Detection of ECG characteristic points using multiresolution wavelet analysis based selective coefficient method. In: *Measurement* 43 (2010), Nr. 2, S. 255–261
- [PS11] PAULIN, F ; SANTHAKUMARAN, A: Classification of breast cancer by comparing back propagation training algorithms. In: *International Journal on Computer Science and Engineering* 3 (2011), Nr. 1, S. 327–332
- [PT85] PAN, Jiapu ; TOMPKINS, Willis J.: A real-time QRS detection algorithm. In: *Biomedical Engineering, IEEE Transactions on* (1985), Nr. 3, S. 230–236
- [RB93] RIEDMILLER, Martin ; BRAUN, Heinrich: A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In: *Neural Networks, 1993., IEEE International Conference On IEEE*, 1993, S. 586–591
- [RTSD12] RAI, Hari M. ; TRIVEDI, Aditya ; SHUKLA, Satyavati ; DUBEY, Vikas: ECG arrhythmia classification using daubechies wavelet and radial basis function neural network. In: *Engineering (NUiCONE), 2012 Nirma University International Conference on IEEE*, 2012, S. 1–6
- [SAN<sup>+</sup>12] SHUKRI, MH A. ; ALI, MSAM ; NOOR, MZH ; JAHIDIN, AH ; SAAID, MF ; ZOLKAPLI, M: Investigation on Elman neural network for detection of cardiomyopathy. In: *Control and System Graduate Research Colloquium (ICSGRC), 2012 IEEE IEEE*, 2012, S. 328–332
- [SC12] SUN, Yuwen ; CHENG, Allen C.: Machine learning on-a-chip: A high-performance low-power reusable neuron architecture for artificial neural networks in ECG classifications. In: *Computers in biology and medicine* 42 (2012), Nr. 7, S. 751–757
- [Sed14] SEDGHAMIZ, Hooman. *Pan Tompinks, Matlab implementation*. March 2014
- [STB97] SAHAMBI, JS ; TANDON, SN ; BHATT, RKP: Using wavelet transforms for ECG characterization. An on-line digital signal processing system. In: *Engineering in*

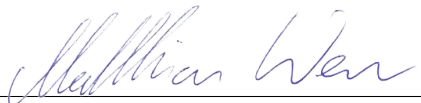
- Medicine and Biology Magazine, IEEE* 16 (1997), Nr. 1, S. 77–83
- [Tha10] THALER, Malcolm S.: *The only EKG book you'll ever need*. Lippincott Williams & Wilkins, 2010
- [VV13] VERMA, Shalini ; VASHISTHA, Richa: Efficient RR-interval time series formulation for heart rate detection. In: *Multimedia, Signal Processing and Communication Technologies (IMPACT), 2013 International Conference on IEEE*, 2013, S. 84–87
- [WBC13] WINTERSTEIN, Felix ; BAYLISS, Samuel ; CONSTANTINIDES, George A.: High-level synthesis of dynamic data structures: A case study using Vivado HLS. In: *Field-Programmable Technology (FPT), 2013 International Conference on IEEE*, 2013, S. 362–365
- [ZFX11] ZHENGZHONG, Gao ; FANXUE, Kong ; XU, Zhang: Accurate and rapid QRS detection for intelligent ECG monitor. In: *Measuring Technology and Mechatronics Automation (ICMTMA), 2011 Third International Conference on Bd. 1 IEEE*, 2011, S. 298–301
- [Zha00] ZHANG, Guoqiang P.: Neural networks for classification: a survey. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 30 (2000), Nr. 4, S. 451–462
- [ZKR10] ZADEH, Ataollah E. ; KHAZAEI, Ali ; RANAEE, Vahid: Classification of the electrocardiogram signals using supervised classifiers and efficient features. In: *computer methods and programs in biomedicine* 99 (2010), Nr. 2, S. 179–194

## Erklärung

*Hiermit erkläre ich, dass die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt wurde. Die aus anderen Quellen oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet.*

*Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder in ähnlicher Form in anderen Prüfungsverfahren vorgelegt.*

Wien, am 6.1.2017



---

Matthias Wess