Circuits and Systems
Mekelweg 4,
2628 CD Delft
The Netherlands
http://ens.ewi.tudelft.nl/

CAS-2018-02

M.Sc. Thesis

Energy Efficient Feature Extraction for Single-Lead ECG Classification Based On Spiking Neural Networks

Ali Eralp Kolagasioglu B.Sc.

Abstract

Cardiovascular diseases are the leading cause of death in the developed world. Preventing these deaths, require long term monitoring and manual inspection of ECG signals, which is a very time consuming process. Consequently, a wearable system that can automatically categorize beats is essential.
Neuromorphic machines have been introduced relatively recently in the science community. The aim of these machines is to emulate the brain. Their low power design makes them an optimal choice for a low power wearable ECG classifier.
As features are crucial in any machine learning system, this thesis aims at proposing an energy efficient feature extraction algorithm for ECG arrhythmia classification using neuromorphic machines. The feature extraction algorithm proposed in this thesis consists of the merger of a low power feature detection and a feature selection algorithm. Also, different network configurations have been investigated to achieve classification using an LSM architecture. The resulting system can accurately cluster seven beat types, has an overall classification rate of 95.5%, and consumes an estimate of 803.62 nW.

Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

Energy Efficient Feature Extraction for Single-Lead ECG Classification Based On Spiking Neural Networks

Thesis

submitted in partial fulfillment of the
requirements for the degree of

Master of Science

in

Computer Engineering

by

Ali Eralp Kolagasioglu B.Sc.
born in Ankara, Turkey

This work was performed in:

Circuits and Systems Group
Department of Microelectronics & Computer Engineering
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

Delft University of Technology
Department of
Microelectronics & Computer Engineering

The undersigned hereby certify that they have read and recommend to the Faculty of Electrical Engineering, Mathematics and Computer Science for acceptance a thesis entitled **"Energy Efficient Feature Extraction for Single-Lead ECG Classification Based On Spiking Neural Networks"** by **Ali Eralp Kolagasioglu B.Sc.** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: 23.02.2018

Chairman:	Prof. Dr. Ir. Alle-Jan van der Veen
Advisor:	Dr. Ir. Rene van Leuken
Committee Members:	Dr. Ir. Zaid Al-Ars
	Dr. Carlo Galuzzi
	Dr. Amir Zjajo

Abstract

Cardiovascular diseases are the leading cause of death in the developed world. Preventing these deaths, require long term monitoring and manual inspection of ECG signals, which is a very time consuming process. Consequently, a wearable system that can automatically categorize beats is essential.
Neuromorphic machines have been introduced relatively recently in the science community. The aim of these machines is to emulate the brain. Their low power design makes them an optimal choice for a low power wearable ECG classifier.
As features are crucial in any machine learning system, this thesis aims at proposing an energy efficient feature extraction algorithm for ECG arrhythmia classification using neuromorphic machines. The feature extraction algorithm proposed in this thesis consists of the merger of a low power feature detection and a feature selection algorithm. Also, different network configurations have been investigated to achieve classification using an LSM architecture. The resulting system can accurately cluster seven beat types, has an overall classification rate of 95.5%, and consumes an estimate of 803.62 nW.

Acknowledgments

I would like to thank my supervisors Dr. Ir. Rene van Leuken Dr. Amir Zjajo and Dr. Sumeet Kumar for all their assistance and support during the writing of this thesis. Without them, this would not have been possible.

I would like to thank my colleague Johan Mes for providing the simulation tool for the neuromorphic machine. I would like to thank cardiologist Dr. Ozcan Ozeke for his assistance in helping me understand ECG signals and certain arrhythmias.

I would like to express my gratitudes to my family. Especially to my parents and my cousins. My parents who have always supported me through this process and my life. Thank you for enduring all my mistakes and helping me learn through them, I have always hoped that I have made you proud. My dearest cousins, my three sisters, who have always been there for me through good and bad. Thank you for all the memories, all the laughs and all the support, I have always been proud of all of you. They say that friends are the family we choose for ourselves, and I have always appreciated my friends but when it comes to family I don't think I could have been more lucky or picked anyone better.

I would never forget my friends as well. I have never wanted or cared for circumstantial friends. I have been lucky enough to have many friends that share my opinion and have been with me throughout my life no matter where I am, no matter how many thousands of kilometers are between us. I would like to believe that I have made such friends throughout my thesis journey as well. Thank you very much for always supporting me and giving me a wonderful time.

Ali Eralp Kolagasioglu B.Sc.
Delft, The Netherlands
23.02.2018

Contents

List of Figures

List of Tables

Introduction 1

Cardiovascular diseases are the leading cause of death in the developed world, accounting for more than 30% of the deaths [9]. In order to prevent these deaths long term *electrocardiography (ECG)* monitoring is required. Following the monitoring, cardiologists need to examine the acquired ECG signals manually and find the sick beats. This is a very time consuming job and some of the diseases might even require immediate action when detected. Thus, a wearable device that can cluster the beats is essential.

Neuromorphic systems have been introduced in 1990 by Carver Mead [10]. Neuromorphic systems aim to achieve brain emulation by imitating biologically plausible models of neurons, synapses and learning algorithms. The appeal of neuromorphic systems comes from their low power promise and the feasibility of implementing such a system. Therefore, they are an excellent candidate for the classification task at hand.

The nervous system consists of billions of *neurons*, which communicate through electrical and chemical spikes. Together they form what is called a *Spiking Neural Network (SNN)*. The neurons are not directly connected to each other, their connection is achieved by *synapses*, which convert their electrical spikes into chemical spikes to transfer them to the other neurons that they are connected to. Perhaps the most important functionality of a synapse is the learning. Learning is realized in the synapses by a mechanism commonly-named *synaptic plasticity*. An anatomical model of a neuron and a synapse can be seen in Figure 1.1

The most basic learning algorithms that govern or explain the synaptic plasticity are based on Hebbian learning, which was introduced by Donald Hebb in 1949 [11]. It was then further developed in time and rules such as Anti-Hebbian learning was introduced by various people such as [12], [13], [14].

1.1 Problem Statement

The main problem at hand is choosing the correct features when classifying ECG data. The aim is to create a wearable device that can cluster different beat types. Consequently, due to the large number of potential diseases, each with its own characteristic features, and due to each person having a distinct characteristic ECG signal, complexity of the problem is significant. It is also worth mentioning that, the features that work best for classifying varies even from learning model to learning model [15].

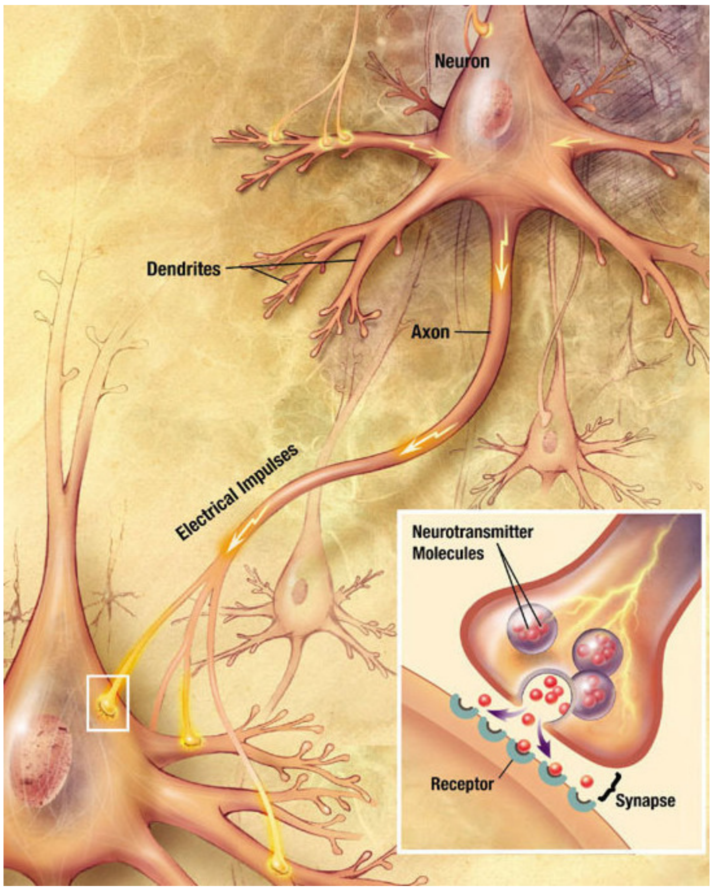

Figure 1.1: Anatomical Neuron and Synapse

Another problem noted is related to the character of ECG signals, namely the fact that they vary from person to person [16]. In addition, there is a very limited number of sick beats in the databases. These two problems emphasize the importance of using an unsupervised learning method.

1.2 Approach

Classification problem consists of many steps, the process flow to achieve classification can be seen in Figure 1.2.

First of all as mentioned in the problem statement using the right set of features is essential in classification. Thus, the main goal is to find a suitable feature extraction algorithm that is compatible with an SNN. Using as many features as possible is not always the optimal solution both in terms of power and accuracy. In order to achieve a good trade-off between power and accuracy, a feature selection algorithm is a must. The main goal in selecting and modifying algorithms for both feature detection and selection is to find algorithms that are very low power and still achieve acceptable results.

In order to achieve this goal various algorithms have been examined in terms of their

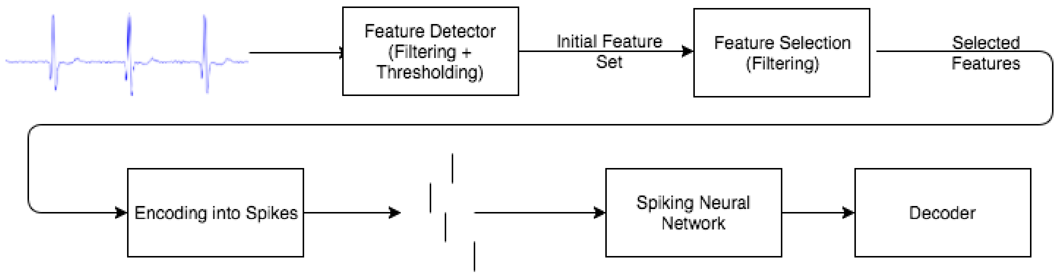

Figure 1.2: Flow Diagram of the Approach to the Classification Problem

power-efficiency, and they have been implemented and tested to see their performance. The next step is to classify the features. This consists of two steps: first the encoding of the features into spikes and then configuring the network for the classification.

1.3 Goals

The main goals of the thesis are:

- Proposing a low-power feature extraction algorithm to be used jointly with a neuromorphic machine.
- Designing a wearable low-power one-lead ECG signal global arrhythmia classifier.

1.4 Contributions

The main contributions of this thesis are:

- Providing a low power feature extraction algorithm (feature detection and selection) for ECG signals that would allow for the best accuracy-power trade-off for classification using neuromorphic circuits.
- Providing a better understanding of different configurations that are required by LSM neuromorphic circuit architecture, for it to be able to classify ECG signals given that, the features are encoded using population offset coding.

1.5 Thesis Outline

In Chapter 2 of the thesis a background on ECG signals and arrhythmias, feature extraction, spike encoding, and classification will be provided. Chapter 3 will go over the analysis of feature extraction in two parts, first feature detection and then feature selection. Chapter 4 gives the details on the configurations of the network, achieved classification results, analysis of the results and possible improvements to the classification system. Finally, the conclusion and future work is presented in Chapter 5. More detailed information on the biological background and the simulation model can be found in Appendix A.

Background 2

This chapter aims at giving a brief introduction to ECG signals and the classification process for ECG signals. The elaborations will focus on explaining how the system works, giving examples of the state of the art solutions and the restrictions set by the problem.

2.1 ECG Signals and Arrhythmia

Electrocardiography (ECG) measures electrical activity on the heart using the potential difference between different leads placed on the body. A typical healthy ECG heartbeat can be seen in Figure 2.1. An ECG beat consists of four typical complexes: P, QRS, T, and U. The U wave cannot be distinguished most of the time, thus it is generally not illustrated. The P wave indicates the contraction of the atrial rooms of the heart, the QRS complex shows the contraction of the ventricular rooms and finally the T wave is the relaxation of the ventricular rooms. The shape and size of these waves vary based on the lead that the signal is being observed from. As a result different features can be seen from different leads, which makes diagnosing certain diseases easier from certain leads. However, most of the arrhythmias can be distinguished from the modified lead two (ML-II lead), which can be seen in Figure 2.2. Moreover the aim of this thesis is to design a system that can potentially be implemented as a wearable device. This factor limits us to use only a three lead ECG instead of a hospital level 12 lead ECG. Using three leads means that we can still have multiple channels, however due to the lack of data from such a recording only the ML-II lead is used in this thesis.

We can also see some of the typical features in Figure 2.1 that are examined on an ECG signal to detect abnormalities. Arrhythmias as their name suggest are usually abnormalities in the rhythm of the heart (i.e. heart rate). An example of an arrhythmia can be seen in Figure 2.3. Making use of features other than the heart rate, help differentiate between a regular beat and a sick beat, and also assist in differentiating between different beat types. Existing beat types in the MIT-BIH Arrhythmia database [17] can be seen in Table 2.1.

All the features have a pre-determined normal range in an ECG signal. However, there are differences between each person's normals, so these pre-determined ranges are not always useful. Instead machine learning is a promising tool to overcome the problems caused by differences that emerge in different people. On top of this, another problem, which we encountered with ECG signals is that there are not enough labeled beats for training as it is a very time consuming job for

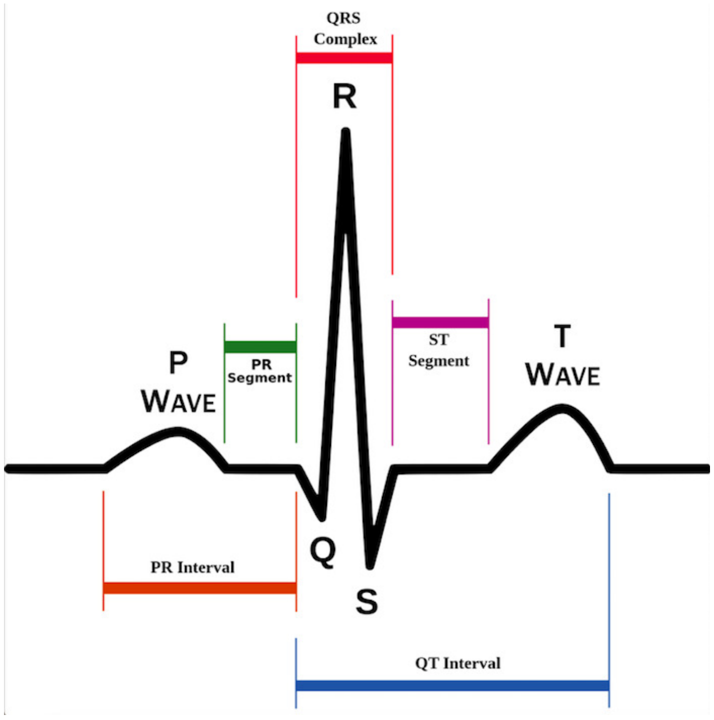

Figure 2.1: ECG Signal and Typical Features

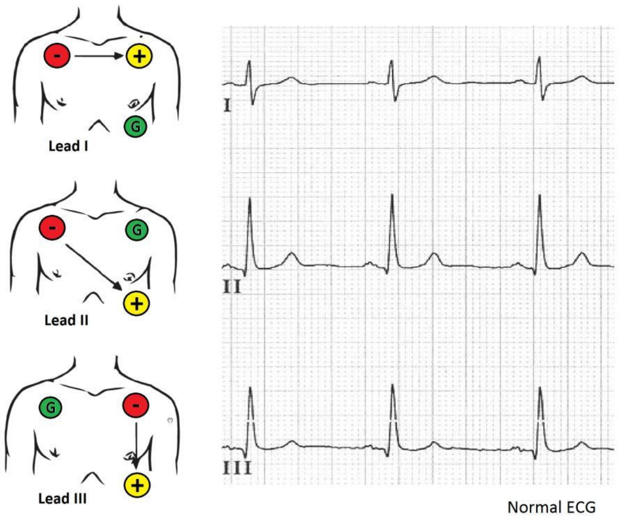

Figure 2.2: Placement of 3-lead ECG. Looking at the potential difference from the right shoulder to the left hip gives the ML-II signal.

cardiologists. This problem combined with the above one causes significant difficulties to classification tools. In order to solve these issues in unison we need a system that utilizes unsupervised machine learning. This has been strongly emphasized in [18].

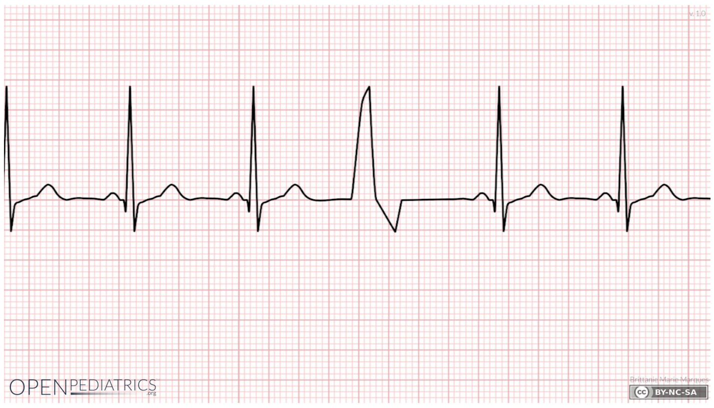

Figure 2.3: Premature Ventricular Contraction Beat Among Regular Beats

Table 2.1: Arrhythmia Types and Their Respective Annotations in MIT-BIH Database

Annotation	Meaning
N	Normal Beat
L	Left Bundle Branch Block Beat
R	Right Bundle Branch Block Beat
A	Atrial Premature Beat
a	Aberrated Atrial Premature Beat
J	Nodal (junctional) Premature Beat
S	Supraventricular Premature Beat
V	Premature Ventricular Contraction
F	Fusion of Ventricular and Normal Beat
!	Ventricular Flutter Wave
e	Atrial Escape Beat
j	Nodal (junctional) Escape Beat
E	Ventricular Escape Beat
/	Paced Beat
f	Fusion of Paced and Normal Beat
Q	Unclassifiable Beat
|	Isolated QRS-like Artifact

2.2 Feature Detection

Feature detection in ECG signals mainly focuses on the signal from the ML-II lead. The main property of this signal is (as can be seen from Figure 2.1) that the R waves are quite large and dominant. Thus, the algorithms mainly focus on first detecting the QRS complex and then finding the P and the T waves accordingly. All the algorithms have the same steps behind them. They subject the ECG signal to various filters (which is called preprocessing) both to suppress noise and to accentuate the QRS complex. The resulting signal then, goes through an adaptive thresholding phase to

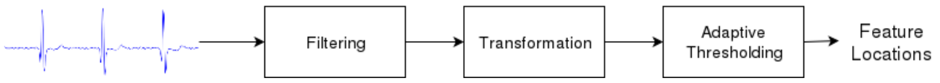

Figure 2.4: Flow Diagram for Feature Detection

find the QRS complexes. Finally if the algorithm supports it, the P and T waves are searched in small windows around the QRS complex. This section aims at exploring some of the possible algorithms for feature detection and comparing them with respect to the goal of this thesis.

2.2.1 Methods and Algorithms

In this section, four distinct feature detection algorithms for ECG will be investigated. ECG feature detection is divided in to two sections most of the time: QRS detection, and P and T wave detection. The following sections will go through some of the possible algorithms developed to detect these features, and will go into further detail in the implemented algorithms.

2.2.1.1 QRS Detection[1]

Most papers focus on the QRS detection of ECG signals, since they are much easier to locate due to the high amplitude R spike in the ML-II lead of an ECG as can be seen from Figure 2.1. Most algorithms depend on preprocessing the signal to obtain a cleaner signal with less noise, transforming the signal in a way that would emphasize the QRS complex, and finally using an adaptive thresholding method to find the QRS complex locations. A flow diagram for this process can be seen in Figure 2.4.

An exception to this method can be [19], which uses the energy of the signal to locate the QRS complex (making use of the high amplitude R peak). The algorithms to be explored in this subsection include [1], [8], and [20]. In this thesis, all these algorithms have been implemented and their accuracies and complexities compared. Since the source codes for these algorithms were not available, except for [8] and even for that one the provided code had to be altered, several additions have been made to all of them, which will be explained throughout this section. As a result of the added complexity, [21] has not been implemented. It was not worth for the small gain in accuracy in the case of a hardware implementation, .

In [8] a mathematical expression is used that calculates the length of a line within a given window. This transformation is called curve length transform. Since the QRS complex has a spike, the window that contains the spike has a much higher value than the others. The shape of the signal after transformation can be seen in Figure 2.5. After this transformation, an adaptive thresholding algorithm looks for the QRS complexes and the first threshold is found with the help of a training phase that goes

[1]The implementation work in this subsection has been done in collaboration with my colleague Johan Mes.

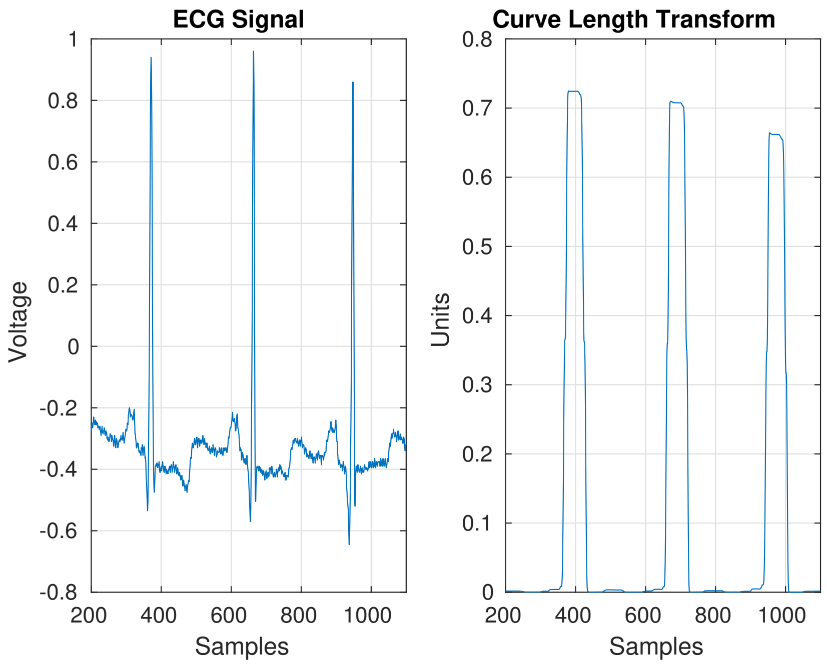

Figure 2.5: ECG Signal and Its Curve Length Transform

through the first 8 seconds of the data to find a reasonable threshold.

The oldest of the evaluated algorithms is [1]. For preprocessing, it subjects the signal to a low pass and a high pass filter cascaded to achieve a bandpass filter. The filter is necessary as it suppresses the muscle noise, 60 Hz interference from the power line, and baseline wander present in ECG signals. The desired pass band for the filter is 5-15 Hz for this algorithm. As it is a very old algorithm and the sampling frequency they used (250 Hz) is different than the one the current database has (360 Hz) a new filter has been designed in MATLAB to fit the requirements. This has considerably (about 50% less error) improved the results compared to using the old filter. Following this, the signal is passed through a derivative filter, this helps accentuate the QRS complex since it has a high slope spike in it. Also the output of this filter is used in the literature to find the onset and end points as well, such as in [22]. Then, the signal is squared to remove the negative parts and finally passed through moving-window integration. An example of the resulting signal after each stage can be seen in Figure 2.6.

The size of the window needs to be set as the largest possible QRS. If the window is too small there might be multiple peaks, and if it is too wide it might merge with the T wave. It has been found empirically in [1] to be 150 ms wide. Following this transform, an adaptive thresholding takes place to find the QRS locations, also a search-back principle is employed when a QRS has not been found in 1.66 times the local average heart rate, the threshold is halved and it goes through the area again. Since a way to find the onset and end points has not been described in detail in [1] the method that has been used in [8] has been adapted to this one. Due to the similarity of the resulting shapes after the transformation it works just as well as it does with [8].

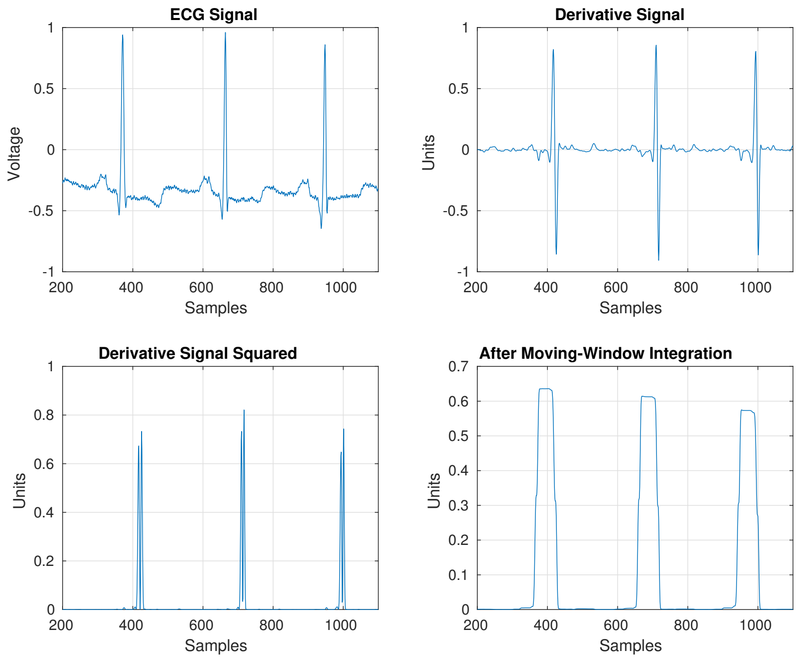

Figure 2.6: ECG Signal and Outputs of Each Stage in [1] Algorithm

Wavelet transforms are used in [20]. Wavelet transforms are a mathematical transform to represent any signal as a combination of multiple wavelets. Different transform levels of an actual ECG can be seen in Figure 2.7. The peaks that can be seen in Figure 2.7 are called the modulus maxima points of a signal. Distinct shapes are obtained in the wavelet domain, for each different shape in the time domain. The algorithm uses an adaptive thresholding algorithm to find the possible QRS complexes in the wavelet domain. It then eliminates the unlikely candidates and marks the QRS complexes. The algorithm starts by looking at the modulus maxima points on the fourth level and goes down from there. One of the reasons for this is that larger scales have fewer modulus maxima points and that helps with the calculation time and the other reason is that higher scales greatly suppress the high frequency noise components and prevents them from creating modulus maxima points. After the modulus maxima points are found using again an adaptive thresholding process, the ones that actually represent QRS complexes have been selected by elimination of isolated and redundant modulus maxima points. Then the R peak is located at the zero-crossing point of a positive maximum-negative minimum modulus maxima pair. The onsets and ends are looked for in level two of the transform, which is a modification taken from [21]. The onsets are the points where the first modulus maximum point of the QRS complex meets the baseline, and the end points are where the last modulus maximum point meets the baseline.

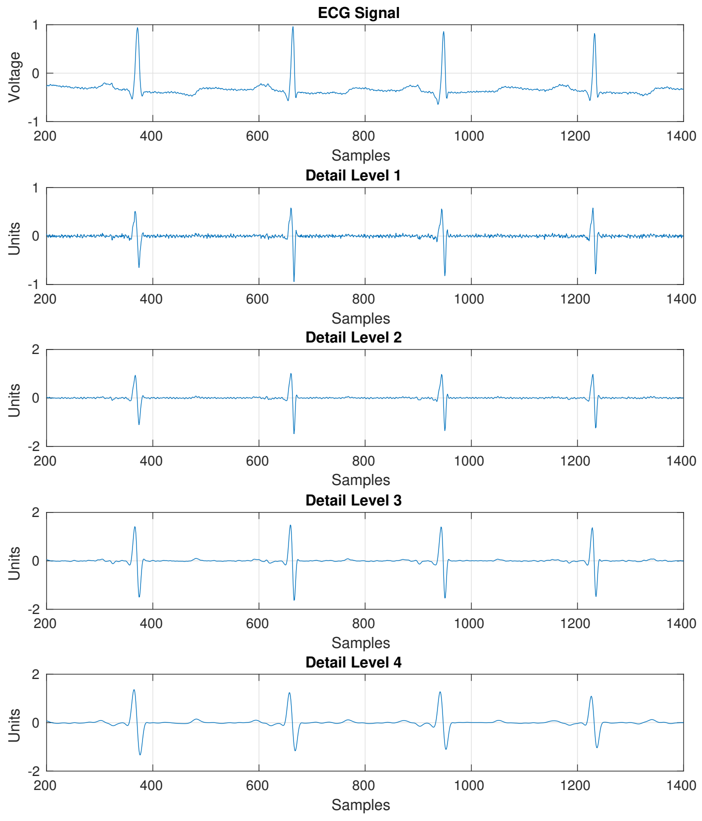

Figure 2.7: All detail levels of the wavelet transform of an ECG Signal

2.2.1.2 P and T Wave Detection

P and T wave detection is a more arduous task compared to QRS complex detection mostly due to their low amplitude. On top of that, it is also more difficult to find the onset and end points due to their much smaller slope in comparison with the QRS complex. There is currently no database available that properly annotates these waves. Only the QT database [23] annotates 30 P and T waves for each patient and the accuracies of the algorithms have been given based on that.

Due to the lack of a proper database to test the implementations, the values presented in the papers have been used for evaluation. Based on [24], [20] performs better than [22] and the method proposed in [24] is slightly better than even [20]. In [22], [1] is used for the QRS detection and then the output of the derivative filter from [1] again for finding P and T waves. On the other hand, [20] as discussed above uses the wavelet transform for detection, however assuming the QRS complexes have already been found the detection for P and T waves is not as complex as the QRS detection. Finally, [24] uses what is called a multiscale morphological derivative for detection of wave boundaries.

Table 2.2: Average Accuracy over All Patients in MIT-BIH on Finding Beats for the Implementation of All Three Algorithms

Evaluation	[8]	[1]	[20]
Average Beat Count	2289	2267	2266
True Positives	2261	2247	2215
False Positives	28	20	51
False Negatives	23	37	69
True Positive Rate	98.98%	98.48%	96.92%
Positive Predictive Value	98.73%	99.13%	97.58%

2.2.2 Comparisons

The main problem encountered is the evaluation of these methods. The most commonly used database to evaluate the performance of these algorithms is MIT-BIH Arrhythmia Database [17]. However, that database only has annotations around the R peak, which limits our evaluation to finding beats. The only database, as far as we are aware of that contains onset and end points, as well for the complexes, is the QT database [23]. As previously mentioned, even that database only contains manual annotations for only around 30 beats per patient. Thus, the algorithms were evaluated on the MIT-BIH database only on their capabilities at finding beats, and the results can be seen in Table 2.2. The achieved results are slightly worse than the ones reported in the papers due to the absence of the actual source code for the given algorithms.

Following this, a complexity comparison between the algorithms has been made to evaluate their power consumption. The results can be seen in Table 2.3. It can be seen that [1] has the most complex transform, whereas [20] has the most complex detection algorithm. On the other hand, [8] is relatively simple transform and a simple detection algorithm. Considering the accuracies obtained from the implementations as well, [8] proves to be the most suitable algorithm for the purpose of this thesis. It is worth mentioning that the accuracy percentages achieved by [20] and [21] are superior to [8] in the papers, however the added complexity prevents them from being a viable candidate for a wearable system.

2.2.3 Conclusion

Spiking neural network that will be used for classification causes no restrictions to the feature detection algorithms. Considering all the data in this section the adaptation of [8] achieves the best results in terms of QRS complex detection. Moreover, it is also the least complex one of the algorithms. For P and T wave detection [20] will be used as it is not as complex when only applied to the P and T wave detection and it achieves good results. The combination of [8] and [20] has actually been proposed in [3] as well, and has been implemented in [25]. This implementation uses 65 nm technology, has an area of 0.03416 mm^2, and consumes a mere 642 nW.

Table 2.3: Complexity Comparison of QRS Detection Algorithms (L is the length of the signal). “Transform” shows the number of operations required for each algorithm, to transform the signal to their respective domain. “Learning” indicates the required number of steps for [8] to learn a threshold. Finally, “Detection” shows the number of operations required in each iteration (sample) during the detection phase. “If Not Beat” is used to indicate the number of operations required if the sample being examined is detected as non-beat sample and conversely for “If Beat”. “If Beat” requires more operations since it requires finding the onset and end points.

	[8]	[1]	[20]
	Transform	Transform	Transform
Add	61L	67L	16L
Mul/Div	7L+1	71L	24L
Square	L	L	-
Square root	L	-	-
For Each Sample	Learning	Learning	Learning
Add	4	-	-
Min	2	-	-
Compare	5	-	-
Mul/Div	2	-	-
For Each Sample	Detection	Detection	Detection
	If not Beat	If not Beat	If not Beat
Add	1	-	4
Min	2	-	1
Compare	1	3	7
Abs	-	-	2
RMS over	-	-	720
	If Beat	If Beat	If Beat
Add	622	615	799
Min	-	7	9
Compare	629	631	1009
Mul/Div	28	18	5
Abs	-	-	319
Sort	-	1	3
RMS over	-	-	54000

2.3 Feature Selection

Having the onset, end and peak points of P, QRS and T complexes, we can obtain tens of signal features. However, not all of these features would be necessary to detect the beat types we are interested in. Since the aim is to create a global classifier, we are aiming to detect all the beats in Table 2.1. Furthermore, most diseases have

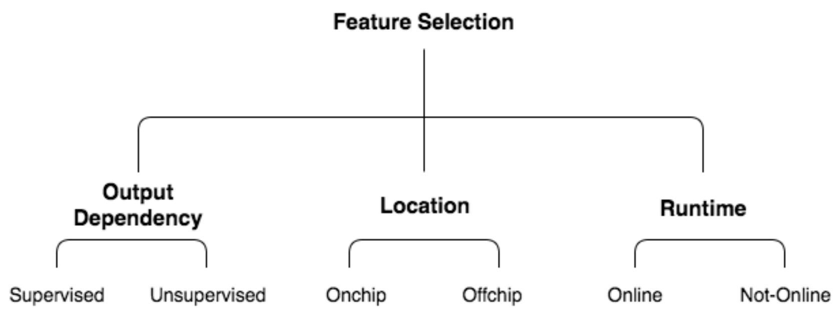

Figure 2.8: Choices to Be Made for Feature Selection

very few indicators and using too many features will make them look more and more similar to healthy beats. The fact that they have only one feature that is different than healthy beats and ten features that are the same as healthy beats might lead to misclassification. In addition, using fewer features lowers training time and network size, as well as allowing lower power consumption. The literature covering this topic is limited for ECG signals, except for [18], existing classification methods in Section 2.4 focus on fewer disease groups which allows them to use a much lower and select feature count. This section will go over various choices that need to be made to decide on a feature selection method and will explore methods that fit these decisions.

2.3.1 Feature Selection Choices

Feature selection itself is a vast field of research since it is a very important part of machine learning. However, there are certain choices that need to be made to find a fitting feature selection algorithm for our system. The different options we have can be seen in Figure 2.8.

The first option we have is to whether use a supervised or an unsupervised feature selection algorithm. Just as in machine learning for supervised feature selection the input data to the system have labels. Supervised feature selection algorithms are embedded and wrapper methods as those are the ones that use the classifier as part of the feature selection process. Most supervised algorithms use a greedy approach to find the feature set that gives the best classification results with the given classifier. As most other greedy approaches this approach gives exceptional results yet is an expensive method. It gives the smallest possible feature set that achieves the best results. However, as mentioned having labeled heartbeats is not a realistic case and the feature set that works for one patient might not work for another one. If we look into our specific case, we have said that we want to achieve unsupervised classification ergo we need an unsupervised feature selection algorithm.

The next choice to make would be whether to have an on-chip or an off-chip feature selection. The advantage of having an off-chip feature selection algorithm is the available computing power at our disposal. As mentioned before ECG signals are rather unique throughout the population. Just as each individual people will not have the same set of diseases, diseases may and will appear differently in different patients. This problem requires the feature selection method to be adaptive to each situation. Thus, an off-chip algorithm would require the data to be sent to a computer for feature detection during operation. However, data transmission is a very power consuming task. The other option in this case is to use an on-chip feature selection algorithm and this thesis will focus on that solution.

The final choice is the run time of the feature selection algorithm. An online feature selection has many benefits over a non-online one, especially for the case of an ECG, since what is normal for an ECG signal can change in time. The only downside of course is the increased computational requirement. It has been shown that an online feature selection algorithm can be used efficiently for ECG classification in [18]. However the nature of our classification system does not allow online feature selection. Classification in spiking neural networks is based on learning certain patterns for each distinct input data. However, an online feature selection algorithm will cause the input data set to change. This will completely change the results of the classifier and it would need to be retrained for the new feature set so that it can learn the new patterns. Thus, the use of spiking neural networks restrict us to non-online feature selection.

2.3.2 Methods and Algorithms

There are three main methods for feature selection: Filter, Embedded and Wrapper methods [26]. Filter methods rely on the intrinsic statistical properties of the given feature set. This can be looking at the correlations within the data using a correlation matrix or creating a new orthogonal basis for the feature set using Principal Component Analysis (PCA) or using Independent Component Analysis (ICA) to again create a reduced feature set. Embedded and wrapper methods on the other hand use the output of the learning machine (classifier) and try to optimize the output of this machine with various feature sets. As a result embedded and wrapper methods usually have better results, yet they are computationally more expensive. The main difference between these two methods is that embedded methods have an added metric or penalty in the learning process.

In the end it is decided to use a non-online, on-chip, unsupervised method for feature selection. Having this in mind, we can still select between all types of feature selection algorithms, namely filter, embedded and wrapper methods. When dealing with unsupervised feature selection embedded and wrapper methods still use the classifier output to update their feature set. Due to the absence of labeled data, they do this by using a pre-determined metric that shows them how good the output clusters are. However, taking into account that we are experimenting with an unexplored, abstruse classification method, filtering methods are a more reasonable choice. Fitting these criteria, two different methods have been explored: Principal

Component Analysis (PCA) and Correlation Matrix.

2.3.3 Conclusion

Due to the restriction of the problem at hand and restrictions caused by the neuromorphic machine, it is decided to use a non-online, on-chip, unsupervised filtering method for feature selection.

2.4 Encoding and Classification

The final step in the classification is encoding and the classifier. This section investigates various encoding methods for spiking neural networks and shows the state of the art classifiers for ECG classification. For the classification simulations the simulator in [27] has been used. A short introduction to neuromorphic machines and the simulator can also be found in Appendix A.

2.4.1 Encoding Algorithms

Two main methods of encoding are rate based coding and time based coding. As their names suggest the rate based coding creates a stream of spikes with different rates based on the input amplitude. Whereas time based coding employs the added flexibility of a spiking neural network and encodes the data based on the spike time. Since learning in spiking neural networks is also based on spike times this method can be practiced. Two algorithms implemented in [27] that employ these schemes are respectively BSA [28] and population offset coding [2]. Based on [27] population offset coding can be used efficiently in spiking neural networks. Therefore, this method has been used.

Population offset coding, as described in [2], employs an encoding based on receptive field arrays. This allows a continuously valued input variable to be encoded using a population of neurons. The neurons in the population display overlapping sensitivity profiles. In this work Gaussian activation functions have been used. In this encoding scheme, highly stimulated neurons fire first, whereas the less stimulated ones fire later. Their stimulation profile is as mentioned before determined by a Gaussian activation function and this function determines each neuron's receptive fields. Each input stream is normalized and then based on the number of input neurons assigned for each input, receptive fields are set. An example of an encoding can be seen in Figure 2.9. In this thesis, a five neuron resolution for each input has been used.

2.4.2 Classification Methods

This section discusses the various ECG classification methods in the literature. Most of these classifiers focus on a small number of beat classes as non-global classifier usually score better than global ones since they can be optimized for a smaller number

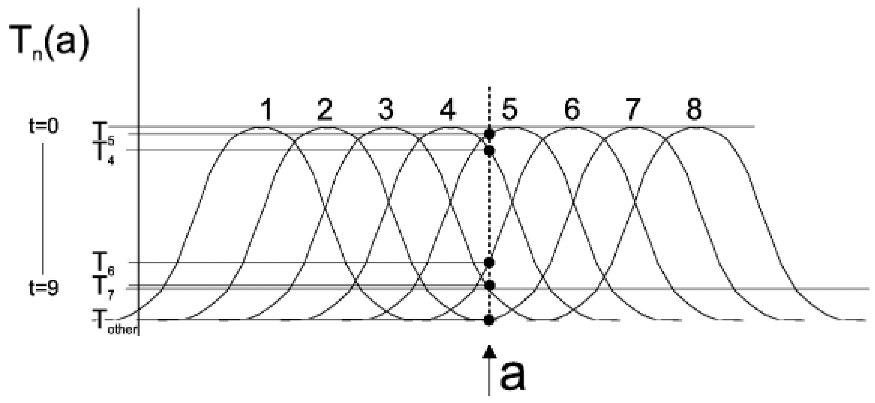

Figure 2.9: Population offset encoding [2]. The value **a** is encoded such that, first neuron five fires, then four, six, three, and seven in that order.

Table 2.4: State of the Art Comparison

	Accuracy	Learning	Features	Disease	Power	Tech.	Area	Supply
[30]	98.9%	Supervised	18	2	NA	NA	NA	NA
[31]	86%	Supervised	10	Prediction	2.78μW	65nm	0.112mm^2	1V
[32]	~99%	Unsupervised	3+Beat	2	NA	NA	NA	NA
[33]	NA	Supervised	7+Beat	5 Classes	NA	NA	NA	NA
[34]	98.5%	Unsupervised	12	16	NA	NA	NA	NA
[35]	97.25%	NA	Beat	3	5.967μW	0.18μm	2.465mm^2	1.2V
[36]	87%	Supervised	3	4	112.7μW	65nm	NA	NA
[37]	˜99%	Unsupervised	9	2	NA	NA	NA	NA
[38]	97%	Unsupervised	5	3	NA	NA	NA	NA
[39]	97%	Supervised	6	4	NA	NA	NA	NA
[29]	98.41%	Supervised	NA	4	NA	NA	NA	NA

of items [29]. A comparison of various methods can be seen in Table 2.4.

First of all it can be seen from Table 2.4 that only one of the classifiers actually is a global classifier, since only [34] is used to classify 16 types of diseases. This is important since this thesis also aims for global classification. All the other classifiers are focused on a small number of beats types. As a result, non of them have an embedded feature selection process, which is novel to this work. On the other hand, [34] uses what is called a Hermite function to transform the ECG signal and feeds these coefficients to an SOM. The shortcomings of this method are mainly the difficulty of implementing Hermite functions, and the need to use data from two separate leads.

Some other most common shortcomings in this list are the algorithms being supervised, and the fact that most of these are not designed for wearable systems and they count on the brute processing power of a computer using samples from the beats and complicated classifiers.

2.4.3 Conclusion

In order to make use of the timing in the network, population offset coding has been decided to be used for encoding. Though unconventional, spiking neural networks are very promising for classification tasks in the future, thus further research should be done in this area. In addition their low power promise makes them an excellent choice as a classifier for our problem.

2.5 Conclusion

This chapter introduces ECG signals and arrhythmias, feature detection for ECG signals, the restrictions imposed on feature selection by the ECG problem and the spiking neural network. Finally, it offers a brief introduction to the encoding scheme used in the simulator and compares existing ECG classifiers in the literature.

The most optimal solution for feature detection is chosen to be a combination of [8] and [20], which was also proposed in [3]. Feature selection choices have limited the solution to be a low power on-chip filtering method. Finally, population offset encoding has been introduced. The following chapter will provide in depth analysis for the implementation of feature detection and selection algorithms. This will be followed by the network configuration and results of classification.

Feature Extraction Implementation 3

This chapter elaborates the work done on feature extraction. In the course of research and experimentation for feature detection and selection various designs have been explored. This chapter will give a deep analysis of the feature detection algorithm and the implementation of the feature selection algorithm. The main factors leading to the decisions have been low power consumption and accuracy. A block diagram showing the overview of feature extraction can be seen in Figure 3.1.

3.1 Feature Detection

In this section, the feature detection algorithm is divided in two parts: QRS detection, and P and T wave detection. The following sections will go through in detail the algorithm's implementations.

3.1.1 QRS Detection

As mentioned in Chapter 2, four algorithms have been implemented and tested to reach a decision. The optimal choice for QRS detection has been decided to be a slightly modified version of [8]. This subsection will go through the details of the algorithm and provide example results for the feature detection scheme.

Preprocessing: As with most other algorithms, the first step is preprocessing. For this purpose only a low-pass filter is used, this was due to the complexity of designing a high-pass filter with a cutoff frequency as low as 5 Hz. The low-pass filter was designed using the filter design tool of MATLAB. Unlike [8] and [1] an FIR filter has been used. Greatest advantage of an FIR filter over an IIR is the linear phase response. The filters used in [8] and [1] also have linear phase responses, however due to its simplicity it has

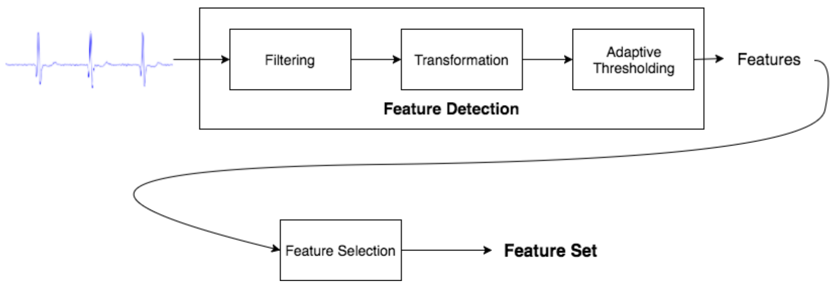

Figure 3.1: Feature Extraction Block Diagram

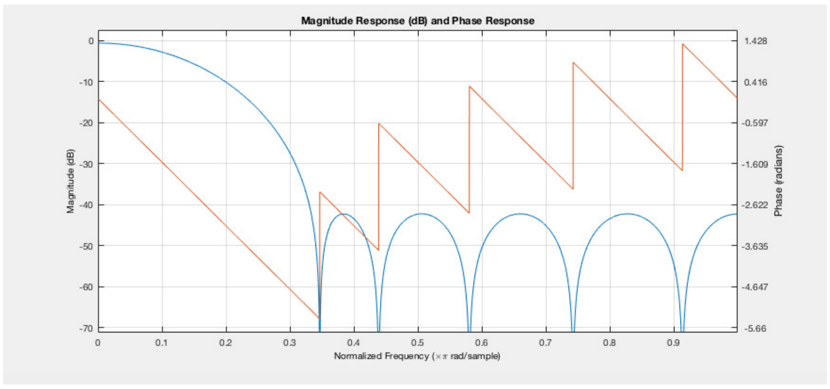

Figure 3.2: Magnitude and Phase Response of The Low-Pass Filter

been decided to use an FIR filter when starting a new. The filter used has a cutoff frequency at 10 Hz and the stop band frequency starts at 60 Hz (in order to suppress the noise from the power line) and has a stop band attenuation of 40 dB. The magnitude and phase response of the filter can be seen in Figure 3.2. Unlike the filters in [8] and [1] it has been designed for a sampling frequency of 360 Hz since that is the frequency at which the database is sampled. As a consequence, with respect to the original filter used in [8] it results in around 50% less errors. In addition, the resulting filter has 11 coefficients, which is less than the filter used in [8] (13) as well, so it is more efficient.

Signal Transformation: The following step is the transformation of the signal. A mathematical expression that calculates the length of a line within a given window is used in [8]. This transformation is called curve length transform. Since the QRS complex has a spike, the window that contains the spike has a much higher value than the others. Curve length transform for discrete signals is described as:

$$L(w,i) = \sum_{k=i-w}^{i} \sqrt{\Delta t^2 + \Delta y_k^2} \tag{3.1}$$

where w is the window length, i is the index, Δt shows the sampling period, and $\Delta y_k = y_k - y_{k-1}$. As mentioned in Chapter 2, the shape of the signal after transformation can be seen in Figure 2.5. The window size is suggested to be as wide as the widest QRS complex. If the window size selected is smaller than a QRS complex, that QRS complex will create multiple peaks after the transformation. If the converse is done, a very large window will result in the subsequent T wave to be involved in the window and the transformation will reach its maximum point later than it is supposed to. As a result, the end point of the QRS could be misdetermined. In [8] a window as wide as 125 ms is recommended, however, [1] recommends a 150 ms one. Our experiments have shown that a 150 ms window gives better results overall for both algorithms. However, 125 ms does give better results for some patients, since [8] has not been tested in the whole dataset their window size choice might be more appropriate for their select data. The relationship between the signal after the transformation and the actual QRS complex can be seen in Figure 3.3.

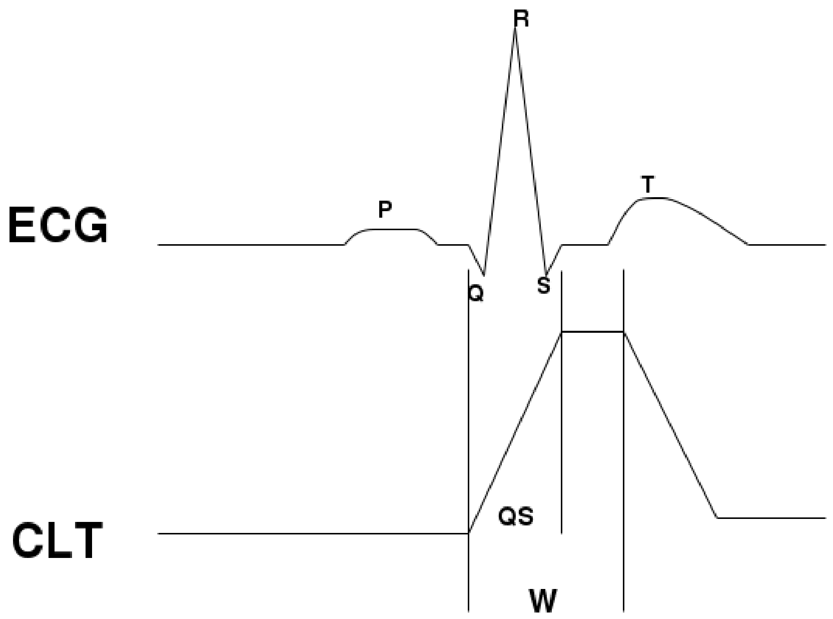

Figure 3.3: Onset and End Points Representation on CLT

QRS Detection: After this transformation an adaptive thresholding algorithm looks for the QRS complexes, and the first threshold is found with the help of a training phase that goes through the first 8 seconds of the data to find a reasonable threshold. After it finds a complex, it looks for the points at which the transform meets the baseline and the point at which it reaches its maximum point. The pseudocode for the training can be seen in Algorithm 1 and the QRS detection algorithm can be seen in Algorithm 2.

The values in the algorithm that are used to adjust the thresholds are based on the amplitude of the curve length transformed signal. Since a different filter has been used those values are different than [8]. For finding the onset of the QRS complex [8] only looks at five samples being less than *Lonset*, whereas the conducted experiments have shown that using 25 samples gives a more robust result. The *eyeClosing* parameter has been set for power saving concerns. After each QRS 250 ms of the signal is skipped since there cannot be another QRS in that duration. Finally, if the algorithm cannot find a QRS complex for 2.5 seconds it reduces the threshold before continuing. A look-back procedure has been implemented in this stage to find the missing QRS complexes. However, it has been concluded that the added complexity was too much for the gain from this modification.

A good and a bad example of onset and end detections can be seen in Figure 3.4 and Figure 3.5. All the other evaluated algorithms also have this problem, but we are focusing on [8]. In all the observed cases the misplacement of the end point occurred on these wide signals. Three methods have been implemented to overcome this issue,

Algorithm 1 Training for Initial Threshold Calculation

```
1: **Parameters** :
2: $sfreq \leftarrow$ sampling frequency
3: $learnTime \leftarrow 8 * sfreq$
4: $adjustThreshold \leftarrow 10$
5: $thresholdMaxDiv \leftarrow 3$
6: $eyeClosing \leftarrow \text{round}(sfreq * 0.25)$
7: $QRSheightThresh \leftarrow 0.002$
8: $minThresh \leftarrow 0.01$
9: $notFoundStep \leftarrow 0.01$
10: $timer \leftarrow 0$
11: $Texpected \leftarrow sfreq * 2.5$
12: $flatRange \leftarrow 25$
13: $QRSoffset \leftarrow 5$
14: **InitialValues** :
15: $thresh \leftarrow 0$
16: $i \leftarrow 1$
17: $Tlower \leftarrow$ Average of $transformSignal$
18: $Tupper \leftarrow$ $Tlower$*$thresholdMaxDiv$
19: **procedure** Initial Training
20:     **while** $i < learnTime$ **do**
21:         $thresh \leftarrow 2 * Tlower$
22:         **if** $transformSignal(i) > thresh$ **then**
23:             $Lmax \leftarrow$ Maximum value of $transformSignal$ from $i$ to $i + eyeClosing/2$
24:             $Lmin \leftarrow$ Minimum value of $transformSignal$ from $i - eyeClosing/2$ to $i$
25:             **if** $Lmax > Lmin + QRSheightThresh$ **then**
26:                 $Tupper \leftarrow Tupper + (Lmax - Tupper)/adjustThrehsold$
27:                 $thresh \leftarrow Tupper/thresholdMaxDiv$
28:         $i \leftarrow i + 1$
29:     $thresh \leftarrow Tlower$
```

however, they all failed. The first method was to increase the window size for the summation and also to change the *eyeClosing* value, which has made the results better for specific patients but made it worse for others. The next approach was based on a stable baseline. If the end point was detected at a much lower signal level than the onset, the end point will be moved forward until they were around the same level. This gave perfect results for patients with little or no baseline wander but, caused trouble in patients with a lot of baseline wander. The final method was based on the assumption that when we see this morphology the QRS width will be set as the maximum assumed QRS width, thus the end point would be moved forward only when the QRS width was equal to the maximum. However, in most cases that was not true so the problem could not be solved.

Algorithm 2 QRS Detection Algorithm

```
1:  procedure QRS DETECTION
2:      while $i <$ Signal Length do
3:          if $transformSignal(i) > thresh$ then
4:              $Lmax \leftarrow$ Maximum value of $transformSignal$ from $i$ to $i + eyeClosing/2$
5:              $Lmin \leftarrow$ Minimum value of $transformSignal$ from $i - eyeClosing/2$ to $i$
6:              $timer \leftarrow 0$
7:              if $Lmax > Lmin + QRSheightThresh$ then
8:                  $Lonset \leftarrow Lmax/100$
9:                  loop with $j$ from $i - eyeClosing/2$ to $i$
10:                     if $transformSignal(j$ to $j + flatRange) < Lonset$ then
11:                         $QRSonsetPoint \leftarrow j - QRSoffset$
12:                 loop with $j$ from $i$ to $i + eyeClosing/2$
13:                     if $transformSignal(j) > Lmax - (Lmax - Lmin)/25$ then
14:                         $QRSend \leftarrow j + QRSoffset$
15:                 $i \leftarrow i + eyeClosing$
16:                 $Tupper \leftarrow Tupper + (Lmax - Tupper)/adjustThrehsold$
17:                 $thresh \leftarrow Tupper/thresholdMaxDiv$
18:             $timer \leftarrow timer + 1$
19:             if $timer > Texpected$ and $Tupper > minThresh$ then
20:                 $Tupper \leftarrow Tupper - notFoundStep$
21:                 $thresh \leftarrow Tupper/thresholdMaxDiv$
22:         $i \leftarrow i + 1$
```

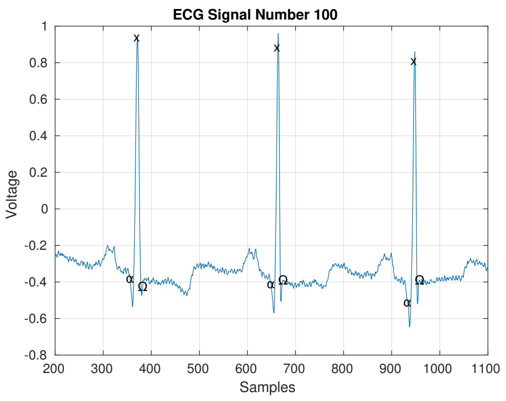

Figure 3.4: Good Example of a QRS Onset (α) and End (Ω) Detection

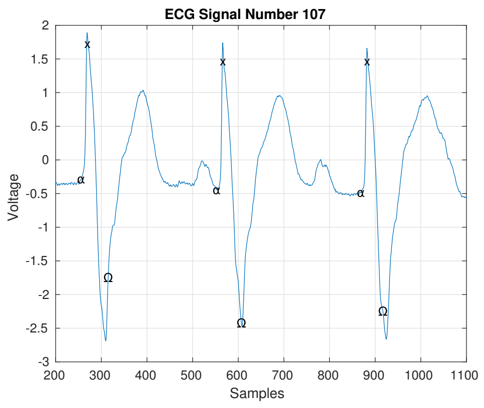

Figure 3.5: Bad Example of a QRS Onset (α) and End (Ω) Detection

3.1.2 P and T Wave Detection

As mentioned in Chapter 2, it is decided to use the wavelet transform method based on [20], for P and T wave detection. Having found the QRS peaks, the P and T waves are only searched where they are expected to be, respectively in small windows before and after the QRS. Thus, the wavelet transform is only applied to those small windows

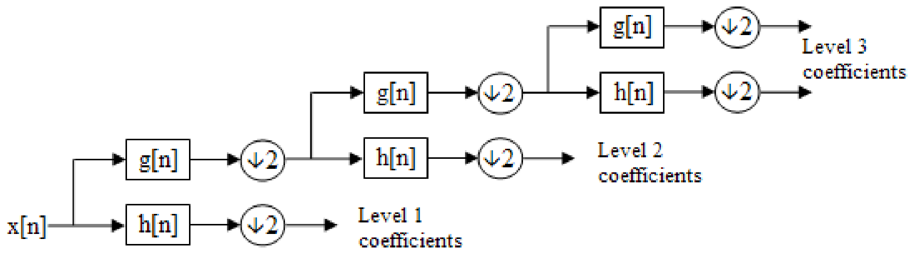

Figure 3.6: Wavelet Transform as Cascaded Filters. Each level consists of a low pass and a high pass filter. The output of the high pass filter a level is that detail levels response, and the output of the low pass filter is transfered to the next stage.

around the QRS complex instead of the whole signal. Window widths have been tried to be determined using online medical resources for an average window size for both P and T waves. However, it has been found that, again due to the fickleness of an ECG signals these values do not hold most of the time. Thus they have been determined empirically. For the P wave the window size is $(heartrate)/4$, where $heartrate$ is in samples and the maximum window size is 410 ms (150 samples). For the T wave on the other hand the window size is $(heartrate)/1.9$, where $heartrate$ is in samples and the maximum window size is 555 ms (200 samples). Thus, wavelet transform is applied only on these small areas. The wavelet transform can be mathematically defined as:

$$f(x) * \Psi_s(x) = \frac{1}{s}\int_{-\infty}^{\infty} f(t)\Psi(\frac{x-t}{s})dt \tag{3.2}$$

where s is a scaling factor so $\Psi_s(x) = \frac{1}{s}\Psi(\frac{x}{s})$, which is the dilation of a basic wavelet $\psi(x)$. When s is selected as 2^j, where $j \in Z$ (integral set), the wavelet transform is called dyadic. In which case, the fast wavelet transform (Mallat algorithm) [40] can be used, which can be represented as a cascaded filter bank as shown in Figure 3.6.

The complexity of the algorithm is much reduced when QRS complexes are already located and only the P and the T waves are searched. The algorithm searches for the peaks that can be seen in Figure 2.7; these peaks are called the modulus maxima points of a signal. While for QRS detection the algorithm goes through all the detail levels of the signal, for P and T wave detection only the fourth level is searched. For higher accuracy, if it cannot be found on the fourth level the next one can also be searched. However, this has not been implemented as a method of power conservation. One of the reasons to start looking from a higher level and going down is that, larger scales have fewer modulus maxima points and that helps with the calculation time and the other reason is that higher scales greatly suppress the high frequency noise components and prevents them from creating modulus maxima points. After the modulus maxima points are found using again an adaptive thresholding process, the ones that actually represent P and T waves have been selected by elimination of isolated and redundant modulus maxima points. According to [20] the algorithm also supports detecting the morphology of these waves. It does this based on the number and order of negative and positive modulus maxima points that represent the wave. This may have worked in the small manually annotated database, however it has been found that it causes many false positives and mistaken onset and end points for the waves. Thus, for the sake of higher accuracy for the majority of the signals it has been assumed that these waves have a positively directed bump shape (the normal shape). This limits the choice of the modulus maxima points to the maximum positive modulus maxima and the negative modulus maxima that comes after it. In case the distance between those points is too much, more than the maximum width of a P wave and a T wave (55 samples, 153 ms) it is assumed that there is no wave. In case of other morphologies, the detection algorithm would either fail to detect the wave at all, or detect a wider or a narrower wave complex, which should be enough for the classification of different types of beats. After finding the onsets and end points the peak is found at the zero-crossing point of a positive maximum-negative minimum modulus maxima pair. The pseudocode for this algorithm can be seen in Algorithm 3. The same algorithm can be used for each wave.

Algorithm 3 Locating P or T waves using discrete wavelet transform, PMM is positive modulus maximum point NMM is negative modulus maximum point

```
1: **Parameters** :
2: $maxPwindow \leftarrow 417ms$
3: $Pwindow \leftarrow (QRSonset - min(HeartRate/4, maxPwindow)$ to $QRSonset)$
4: $maxPwave \leftarrow 153ms$
5: $PonsetThresh \leftarrow 0.5$
6: $PendThresh \leftarrow 0.9$
7: **Variables** :
8: $dwtSignal \leftarrow$ Discrete Wavelet Transform of Signal
9: $PsearchWindow \leftarrow dwtSignal(Pwindow)$
10: procedure MODIFY PSEARCHWINDOW
11:     $wt \leftarrow PsearchWindow$
12:     $PsearchWindow = wt - (max(wt) - (max(wt) - min(wt))/2)$
13: $mm \leftarrow$ Modulus maxima points in $PsearchWindow$
14: procedure REMOVE NON-POSSIBLE ZERO-CROSSINGS
15:     if $((mm(i+1) - mm(i) < maxPwave)\&\&(mm(i+2) - mm(i+1) < maxPwave))\&\&sign(mm(i+1)) == +$ then mm(i) = Deleted
16:     if $mm(i+1) - mm(i) > maxPwave$ then mm(i) = Deleted
17: procedure DETECT ONSET, END AND PEAK
18:     $peak \leftarrow$ zero-crossing between the modulus maxima
19:     $start \leftarrow$ point before PMM $< PonsetThresh * PMM$
20:     $end \leftarrow$ point after NMM $< PendThresh * NMM$
```

Algorithm 3 consists of three procedures. One of the most important modifications made to [20], while trying to optimize it was to modify the wavelet transform such that the maximum positive maximum modulus peak would have the same amplitude as the minimum negative maximum modulus peak. Meaning that if the maximum point of the wavelet transform had a higher amplitude than the minimum point, the signal was moved lower so that they would have the same amplitude and vice versa.

This assured that both peaks that represent a signal are above the noise level for the same noise multiplier. Otherwise, due to the shape of the signal, sometimes one of the peaks are multiple times larger than the other important peak, which pushes us to set different threshold for the negative and the positive part. In some cases the wavelet transform is all-positive or all-negative which, prevents finding any useful modulus maxima pair at all. Therefore, this modification has considerably improved the results. However, no numbers can be provided since there is no way to actually test it due to the lack of a database. Based on empirical testing on various patients showed up to 80% improvement in detection rates.

The next procedure in the algorithm removes the modulus maxima points, which cannot be representing a P or T wave. The first if condition is used to eliminate a certain morphology trait, while at the same time allowing the wave to be detected still. The second if condition on the other hand removes a point if it is too far from another modulus maxima point so it cannot actually be representing a wave.
Finally, the third procedure is for finding the onset, end and peaks of the waves. For the T wave the onset threshold is 0.05 and the end threshold is 0.4. These values were taken from [20]. An illustration of the annotations of the algorithm can be seen in Figure 3.7.

The main problem that could not be solved was caused by the errs in the QRS end detection as mentioned in Subsection 3.1.1. An example of this can be seen in Figure 3.8. It can be seen the early detection of the QRS end causes a large peak in the wavelet transform that prevents finding the actually important peaks. In the first beat in Figure 3.8 this completely prevented the algorithm from finding a peak while in the second one it found a wrong place. Various methods have been tried to overcome this problem, however the problem could not be solved in the end. These methods were mainly based on removing the beginning or the end of the signal in case they had significantly higher amplitude than the mean of the signal.

3.2 Feature Selection

This section will focus on the extracted features from the detected ECG points of interest, the features that help us differentiate between different diseases, and finally about the feature selection algorithms that helps reduce this feature set.

3.2.1 Feature Set

Having detected the important points on the signal, the next most important step is to figure out the most important features to differentiate arrhythmias. In order to do this, we have consulted cardiologists, and websites such as [41] and [42]. Numerous features from all the papers studied have also been taken into consideration while coming up with the feature set. The existing arrhythmias in the MIT-BIH database have been listed in Chapter 2. All the picked features are useful in differentiating between various diseases. The resulting complete feature set is as follows:

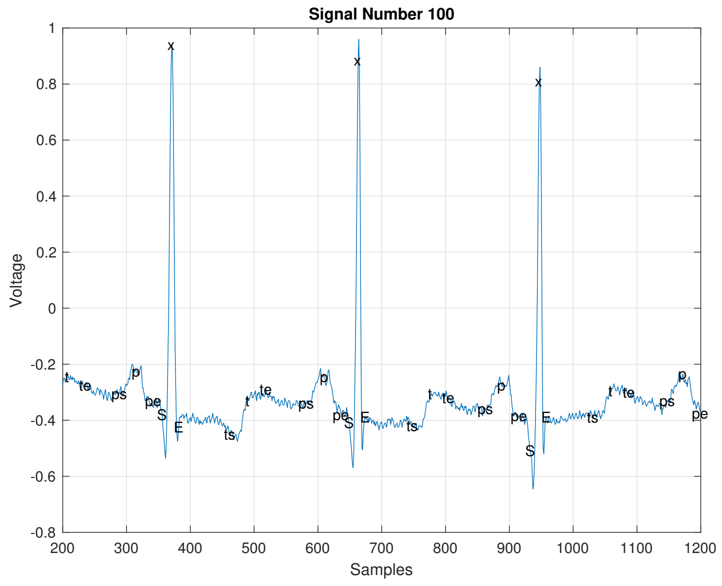

Figure 3.7: Annotations on the ECG Signal

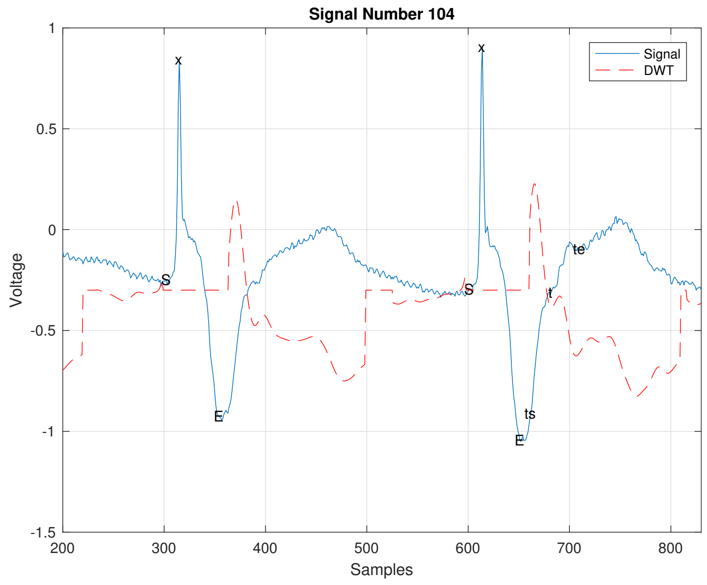

Figure 3.8: Annotations Mistake Using [3]. The dashed line shows the wavelet transform for the windows.

- Heart Rate: The time between the R peaks of the current and the previous beat
- QRS Width: The time from the onset of the QRS complex to the end of the QRS complex
- R Voltage: The voltage level of the R peak

- QR Width: The time from the onset of the QRS complex to the R peak
- RS Width: The time from the R peak to the end of the QRS complex
- P Exists: Binary value showing whether the P wave exists
- PR Interval: The time from the onset of the P wave to the onset of the QRS complex
- PR Segment: The time between the end of the P wave to the onset of the QRS complex
- P Voltage: The voltage level of the P peak
- P Width: The time between the onset and the end of the P wave
- T Exists: Binary value showing whether the T wave exists
- PT Interval: The time between the onset of the P wave and the end of the T wave
- QT Interval: The time between the onset of the QRS complex and the end of the T wave
- ST Segment: The time between the end of the QRS complex and the onset of the T wave
- ST Interval: The time between the end of the QRS complex and the end of the T wave
- T Voltage: The voltage level of the T peak
- T Width: The time between the onset and the end of the T wave
- Previous Heart Rate: The time between the preceding beats R peaks
- PP: The time between the P peaks of the current and the previous beat
- Previous PP: The time between the preceding beats P peaks
- QRS Lowest Voltage: The lowest voltage level of the QRS complex

One more feature that has been found to be important for some diseases, towards the end of this work was the subsequent RR distance. It is recommended that this be considered for future implementations, however, it has not been tested in this work. After deciding on the feature set, a small program to test the differences between the means of these values for different diseases has been implemented. The goal was to examine whether there were or not, some useful features in distinguishing between two diseases. The program showed that while some features are more important than others, all the features were useful in different cases.

In case there was a problem with the detection of P or T waves all the values that are related to either one of them were set to a certain value. The initial method was to set them to zero. However, this has caused many problems with classification due to the encoding method. As mentioned in Chapter 2, in population offset encoding, before the values are encoded into spikes they are normalized. However, when we set some values to zero, the small important changes become unnoticeable. For example, for some patients the heart rate is around 400 samples, and a sick beat is around 350 but, when if this is scaled based on 0-400 rather than 350-400 the difference between 350 and 400 becomes negligible. Another problem with the beats are some outliers. These are the cases when the signal is very noisy or sometimes there is a connection problem and a beat has not been detected for a while, we again receive exceptionally unlikely values for the features. This again causes problems in the scaling of the data. In order to overcome these problems a two-stage algorithm was developed. First the outliers were determined and set to an average value, and in the next step all the zeros were set to the min value within the observation set of those features. They were set to the min value instead of an average value because, it was assumed that it is more likely for something like that to happen in an unhealthy beat. The pseudocode for this algorithm can be see in Algorithm 4.

Algorithm 4 Handling the Outliers in the Observations

```
1: Parameters :
2: $observations \leftarrow$ all observations of a feature
3: $thresh \leftarrow 0.8$
4: procedure FIXING THE OUTLIERS
5:     $outliers \leftarrow observations > (1 + thresh) * mean(observations)$ && $observations < (1 - thresh) * mean(0observations)$
6:     $outliers \leftarrow mean(observations)$
7: procedure FIXING THE MISDETECTIONS
8:     $misdetecitons \leftarrow$ All the zeros in the detection
9:     $misdetections \leftarrow min(observations)$
```

3.2.2 Algorithms

Based on the analysis in Chapter 2 the most reasonable choice for feature selection is an unsupervised, on-chip, non-online filtering method. Based on these limitations two algorithms have been implemented: Principal Component Analysis (PCA) and Correlation Matrix.

PCA: PCA is a statistical analysis tool. It is an orthogonal transformation that is used to convert a correlated set of observations to a set of linearly uncorrelated set of values. Computing PCA is done through first calculating the covariance matrix of the observations matrix, then the eigenvectors and eigenvalues of this matrix are calculated. The eigenvectors form the orthogonal basis for our data. The cumulative energy content of each eigenvalue is calculated, and using this value a set of eigenvectors are selected as the new orthogonal basis to be used, such that more than 90% of the correlation in our data is explained. This algorithm can be seen in Algorithm 5.

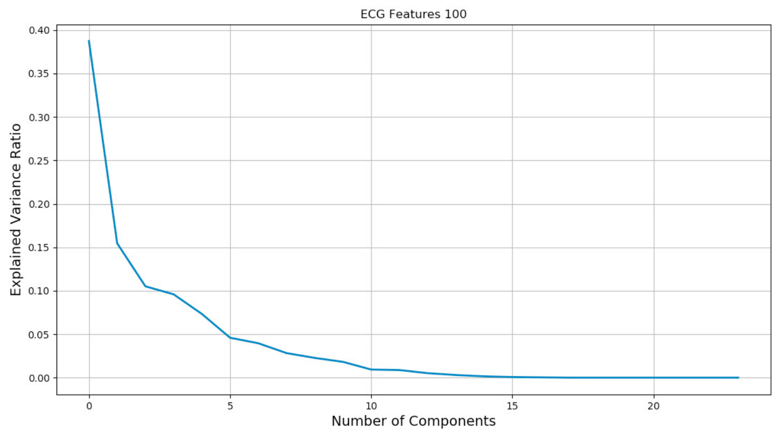

Figure 3.9: Explained Variance vs. Number of PCA Components Used for Patient 100

Algorithm 5 Computing PCA

```
 1: Parameters :
 2: features ← matrix of all observations of all features
 3: procedure PCA Calculation
 4:     $u \leftarrow$ mean value of each feature through all observations
 5:     $B \leftarrow$ deviations of each observation from the mean
 6:     $C$ (Covarience Matrix) $= \frac{1}{n-1} B^* \otimes B$
 7:     $V \leftarrow$ Eigenvectors of $C$
 8:     $D \leftarrow$ Eigenvalues of $C$
 9:     Sort $V$ and $D$ such that $D$ is decreasing
10:     $g_j = \sum_{k=1}^{j} D_k$ for $j = 1, ..., p$ Cumulative Energy Content
11:     Select $L$ components such that $\frac{g_L}{g_p} > 0.9$
```

An example of a graph about explanation of correlation values with each component can be seen in Figure 3.9. In Figure 3.9 we can see that choosing approximately 8 principal components provides sufficient accuracy. This can be interpreted from the graph as after 8 components, each component only adds approximately 1% to the explanation of the variance and 90% of the variance is already explained. This plot also varies from patient to patient, however 8 components are a good approximation for each patient. However, the transform still needs to be implemented for each patient. In addition using the same number of features (or components in this case) for each patient is beneficial since it means we can use the same network.

Correlation Matrix: Correlation matrix of a feature set can be calculated in many different ways. However, an easy way is to calculate the standardized random values (feature observations) and calculate the covariance matrix of that. Pseudo code for this algorithm can be seen in Algorithm 6. Correlation matrix shows the correlation between different features in the observations. As common sense suggests, we do not

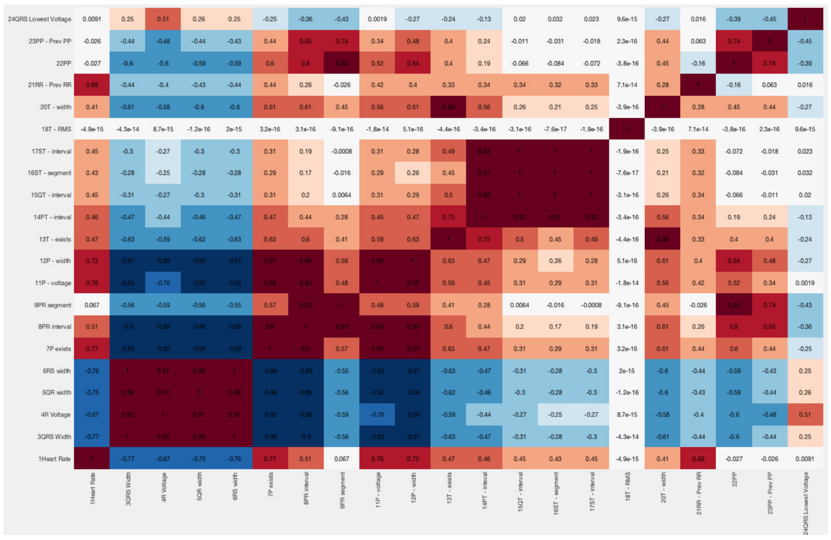

Figure 3.10: Correlation Matrix of Patient 100

require highly correlated data or we can use them for higher robustness in some cases. In looking at a correlation matrix, one also needs to check what are called probability values (p-values). A p-value shows how likely it is to get that correlation in a randomly generated set. Thus, if the p-value is higher than a certain number (say 0.05) the correlation value can be disregarded. Fortunately for our observations that was never the case. If the correlation value is above 80% we have disregarded one of the features that are correlated to one another. Between those two features, if one of the features are correlated to more than one feature, we have kept that feature that is correlated to more features, since that meant that we could eliminate at least one more feature if we kept that one. In other words, the feature that is correlated to most features is kept. The final decision made for eliminating a feature was the trustworthiness of a feature. For example, since it was possible to test the QRS detection, the QRS related futures are trustworthier than P or T wave related features. Thus, if a P or T wave related feature was correlated to a QRS related feature, it made more sense to keep the QRS related feature. An example of a correlation matrix can be seen in Figure 3.10.

Algorithm 6 Computing the Correlation Matrix

```
1: Parameters :
2: $features \leftarrow$ matrix of all observations of all features
3: procedure CORRELATION MATRIX CALCULATION
4:     $features \leftarrow$ Standardize Features
5:     $u \leftarrow$ mean value of each feature through all observations
6:     $B \leftarrow$ deviations of each observation from the mean
7:     $C$ (Covarience Matrix) $= \frac{1}{n-1} B^* \otimes B$
8:     Remove one of the features that are correlated more than 80% or 0.8
```

Figure 3.10 shows the correlation matrix as a heat map, the features that are highly correlated have a red hue where as the ones that have no correlation are white and the ones that are negatively correlated are blue.

Due to the added complexity of PCA and the superior results achieved by the correlation matrix feature set in the classification, correlation matrix has been selected to be used for feature selection.

3.3 Conclusion

This chapter has explained the details of feature detection and selection algorithms used on the ECG signals for this thesis. It has been found that the feature detection algorithms are far from perfect, while due to the restrictions of our problem we are forced to make the hardest and less effective choices for feature selection. In the end, a combination of curve length transform and wavelet transform have been implemented for feature detection and correlation matrix have been used for feature selection. The following chapters will go in to the details of network configurations and results, and finally, conclude this thesis.

Network Configuration and Results 4

This chapter explains the approach taken in configuring the networks and then introduces silhouette coefficients, which are used to evaluate the results. Finally, it gives the results of the classification with explanations of the problems faced and offers possible solutions to overcome these problems.

4.1 Approach

A brief introduction to neuromorphic circuits and the simulation tool used in this thesis can be found in Appendix A. In principle, a neuromorphic circuit emulates the functions of a human brain. It consists of neurons and synapses, and communication between neurons is established based on spikes. The learning process is managed based on spike arrival times to neurons. The main logic behind it is that if two neurons fire consecutively, there should be a correlation between them and the weight of the connection between them will be changed based on the timing of both the spikes. When a certain input pattern is shown to the network repeatedly, the network learns a path and associates it with the respective input pattern. After certain paths have been created in the network, it responds to each distinct input pattern in a unique way.

Among the most common architectures for neuromorphic networks Feed-forward [43], Self Organizing Map (SOM) [44], Liquid State Machine (LSM) [45] have been considered for use. Feed-forward networks consist of three parts: Input layer, hidden layers, and output layer. There can be any number of hidden layers in a feed-forward network. As its name suggests, the data flow in a feed-forward network is only forward. The data (or spikes in our case) from the input layer is fed to the first hidden layer, the one from the first hidden layer to the second hidden layer and so on until the output layer. An SOM network on the other hand consists of two layers: Input layer, and SOM pool. The SOM pool is also used to determine the output. In an SOM network the first layer is fully connected to the pool, and the pool is fully connected within itself. The connections in the pool are arranged in a special manner, every neuron is connected with decreasing weights to its neighbors up to a predetermined range (spatially). Neurons further away from this distance are inhibited. As a result, based on the initial weights of the inputs a different part of the SOM pool fires and the rest are inhibited. Finally, the LSM architecture mainly uses two layers of neurons. The first layer consists of the input neurons, and the second layer is a pool of neurons that are randomly connected among each other. As a result of this connection pattern, each distinct input pattern results in a different order of firing at the LSM pool. An illustration of an LSM network can be seen in Figure 4.1. Consequently, the inputs

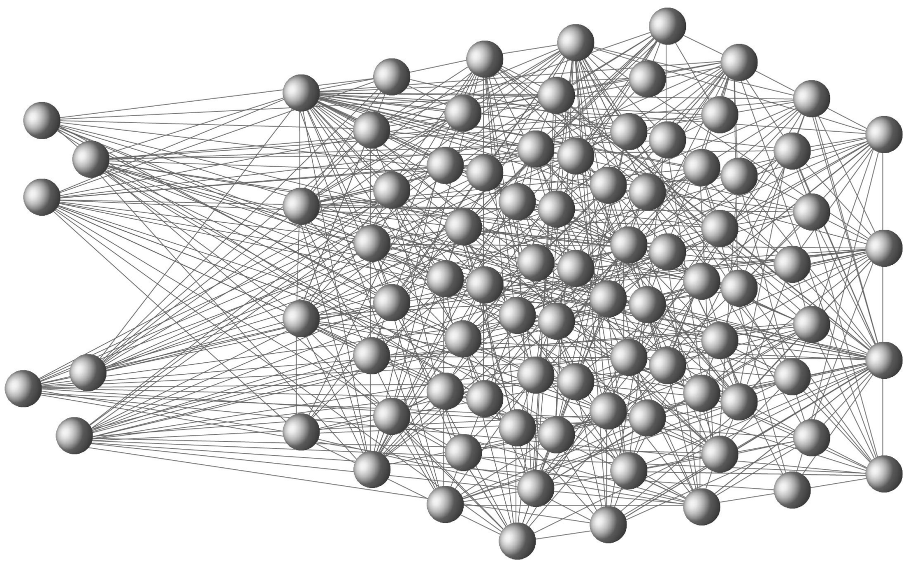

Figure 4.1: Liquid State Machine Architecture

are coded in the time to spike values of the pool neurons. All these architectures can be obtained by changing the configuration parameters of the simulation tool ([27]).

SOM structure is inherently adapted for clustering. However, the aim of this thesis is to create a global clustering scheme, and such a scheme would require a relatively large SOM pool (assuming 9 neurons per cluster, and none overlapping clusters, would result in at least 153 neurons in the pool to cluster all 17 beat types in the database). Such a network causes memory problems in the simulation and also is not necessary as the LSM offers a much more eloquent solution. Since the input is coded in the firing time of the neurons in the pool more clusters can be obtained using a smaller network. Another advantage of LSM networks is their recurrent connections (connections creating loops and allowing a neuron to fire more than once). This architecture type allows the processing of temporal data. One of the goals of this thesis was to compare the results achieved with different architectures. However, as mentioned before the required SOM architecture causes memory problems in the simulator, and a generalized configuration that works on all patients could not be found for the Feed-forward network to complete the tests. As a result this work focuses on the LSM architecture.

As mentioned before, the end goal in a neuromorphic structure is to have a unique path for distinct input patterns, and a unique firing pattern for LSM architecture. Achieving this goal requires delicate adjustments in the parameters. Otherwise, due to the nature of this architecture all the neurons might spike at the same time or maybe non of them will. The first step in the most appropriate and generalizable approach has been found to be making the connections between the first and the second layer more sparse, which would provide the opportunity for different input patterns to trigger different parts of the pool first and increase the number of clusters.

The next step was to add a randomized delay matrix to the connections, which makes sure that the time difference to spike between neurons increases and sometimes even prevents some neurons from firing because they have discharged due to the added delay. This makes the classification task in the end easier as the distinction between the patterns has increased. The final adjustment to the configurations was to lower the weight distributions both from first to second layer and the ones within the pool. This alteration is also used to prevent the network from becoming an all-pass network. One observation in the network was that, initializing the pool as a fully connected pool resulted in the best results, and the pool became sparser after the training. It is also worth mentioning that a pool of 25 neurons was used due to memory constraints but a greater pool size might be beneficial in clustering. As mentioned before, LSM architecture tends to have a recurrent pattern of firing in the pool, which is interpreted as "naturally" handling time varying inputs. However, this behavior leaves very little control over the process and it becomes harder to interpret the results. Taking also into consideration that the duration between two beats is considerably larger than the working frequency of the network (approximately 1000 times), it is difficult to leverage this property. Therefore, a refractory period was selected for the neurons such that, during the testing, when all the connections are strong enough, each neuron would only fire once for each input pattern.

There are also certain things in the network that should be assured for correct operation. One of them is to ascertain that the membrane voltage of each neuron reaches its resting value before feeding each input pattern. Another important thing to make sure is that the depression in the network is strong enough such that a stable state can be reached that is not an all-pass network.

The output spike patterns have been clustered based on the spike time difference in the pool. This clustering is done by various methods, which will be explored in Section 4.3. As in other unsupervised cases such as [34], the clusters have been evaluated based on the dominant beat in a cluster. That is if a beat type is in a cluster that it is not the dominant beat type in, that means it is in a bad cluster.

4.2 Silhouette Coefficients

After conducting a few experiments, it has been found that a metric to evaluate the clusters was necessary. The metric should help in determining whether more clusters are required, whether the clustering methods are resulting in strong clusters, and finally, they should be indicating whether the problem is with the network or the futures are not good enough for clustering.

A common approach that provides information about all these methods is called a silhouette coefficient [46]. Silhouette coefficients can be defined by the following equation:

$$s(i) = \frac{b(i) - a(i)}{max(a(i), b(i))} \tag{4.1}$$

where $a(i)$ is the average distance of beat i to all the other beats in its cluster, and $b(i)$ is the minimum of the average distances of beat i to all the other beats in other clusters. Thus, the smaller the $a(i)$ value the better as it would show similarity of the beat to its own cluster and the larger the $b(i)$ value the better as it would show the dissimilarity to other clusters. It can be seen from (4.1) that:

$$-1 \leqslant s(i) \leqslant 1 \tag{4.2}$$

A value that is closer to 1 shows that a beat belongs to its cluster, whereas a value that is close to -1 shows that the beat fits some other cluster more, and a value that is close to 0 shows that the beat is fits in to at least two clusters well so it is right on the edge.

In this algorithm any means can be used to calculate the distance between the elements of a cluster. In this thesis Manhattan distance has been used to calculate this distance. For each experiment the silhouette coefficients have been calculated both based on the selected features and the spike times of the pool for each beat. This provides us with two separate silhouette values and it can show us whether the problem lies in the features or the output of the network. Another way to use the silhouette coefficient is to determine the number of necessary clusters. Having high silhouette values for the clusters would show that the clusters are very strong and there is no need to create further clusters.

4.3 Clustering Methods for the Output

In order to cluster distinct output patterns of the spiking neural network automatically, five different clustering methods have been used. The strongest silhouette coefficients have been achieved using sequential clustering, so all the clustering methods will be mentioned and more detail will be provided on the sequential clustering method. An additional method that has been used is automatic clustering, which creates the clusters based on the annotations of the beats. This shows how the network affects the silhouette coefficients for perfect clustering and also helps us see whether the beat types are distinct enough to be clustered.

The five approaches used for clustering are as follows: Mean difference clustering, Otsu clustering [47], K-means clustering [48], minimum difference clustering, and sequential clustering.

Mean difference method uses the mean of the sum of absolute value of spike timing differences between a beat and all the rest of the beats for clustering. It starts by calculating the sum of absolute spike timing differences for every output neuron between the first beat and the rest of the beats. Next, a multiple of the mean of these values is used to establish a cluster and the beats in that cluster are removed from the database and the process is iterated repeatedly until there are no beats left to cluster.

Otsu clustering method uses the famous Otsu's method for clustering. Otsu's method is mainly used for finding a threshold to create black and white images from gray scale images using a histogram-based method. The histogram for this method is also based on the sum of absolute spike timing differences for every output neuron between the first beat and the rest of the beats. The method was first done based on just the difference of every beat to the first beat. However, it has been concluded that this might not always give the best results; so two iterative versions have been developed. The first one just picked the first threshold value that Otsu's algorithm created and then removed those beats from the database and iterated again until the desired number of clusters were reached. The other method selected the cluster, created after each iteration that had the lowest variance and iterated again by removing that cluster from the database. However, contrary to my expectations the iterative methods resulted in weaker clusters.

K-means clustering algorithm uses the K-means++ algorithm that is implemented in MATLAB. K-means algorithm is based on finding cluster centers to minimize the intra-class variance. The spike times of the output neurons are directly inputted for this algorithm.

Minimum difference method uses the minimum of the sum of absolute value of spike timing differences between a beat and all the rest of the beats for clustering. It starts by calculating the sum of absolute spike timing differences for every output neuron between the first beat and the rest of the beats. Then a multiple of the minimum of these values is used to establish a cluster and the beats in that cluster are removed from the database and the process is iterated repeatedly until there are no beats left to cluster.

Some of these four methods gave good results for some conditions, but they were not working in every case. An alternative solution was to use a sequential clustering method, which gave better results than all the other algorithms based on the silhouette coefficients and also is a good candidate for implementation in a real world system since it is a sequential system and does not need all the beats to be detected to start classification. The pseudocode for this algorithm can be seen in Algorithm 7. Since we have all the data at hand and we have a limited amount of data the algorithm is implemented in that fashion with for loops, but it is easy to convert it to a sequential version. It is also worth mentioning that the iterative version of this algorithm gives really good results, however the non iterative version can also be used in most cases and this can be set by the parameters.

The sequential algorithm also uses the sum of absolute spike timing differences for every output neuron. In this algorithm this value is calculated between the new coming beat and the average of each cluster. Then these differences are compared to the set limit, which decreases by a percentage at each iteration. Finally, the beat is inserted in one of the existing clusters if it is deemed similar enough; otherwise a new cluster is

Algorithm 7 Sequential Clustering for Spike Times

```
 1: Parameters :
 2: seqDiff ← The initial difference between values to seperate clusters
 3: seqItPercentage ← The percentage by which the threshold is lowered every iteration
 4: iterationCount ← number of iterations
 5: Inputs :
 6: spikeTimes ← Spike times for a beat
 7: beatCount ← Number of beats
 8: procedure Clustering
 9:     Put the first beat in a cluster
10:     currentClusterCount ← 1
11:     for $i = 1$ to $iterationCount$ do
12:         for $j = 2$ to $beatCount$ do
13:             for $c = 1$ to $currentClusterCount$ do
14:                 $diffs = sum(abs(spikeTimes(j) - mean(spikeTimes(Clusters(c)))))$
15:             possible ← Clusters that have $diffs < (seqDiff * seqItPercentage^{i-1})$
16:             Remove beat $j$ from the clusters if it is already there
17:             if possible is empty then
18:                 Create new cluster for beat $j$
19:             else
20:                 Add to the cluster that has the least difference
```

created for the beat. Based on the silhouette coefficients this algorithm resulted in the strongest clusters.

4.4 Results

As mentioned in Section 4.1, LSM architecture has been used to obtain the classification results. As the feature set varies form patient to patient, the input size also varies from patient to patient. Ergo the network has to be slightly modified for each patient. The results are presented in three stages in this section, due to the nature of the problem. There are two steps in the algorithm that directly affect the results we achieve: feature detection and classification. This is because if the feature detection algorithm could not detect certain beats, the classification of those beats automatically fails when we look at the whole system.

In Table 4.1 we can see the detection rate, and the percentage of missed beats from the database due to detection. It can be seen that there are problems with detecting certain beat types, namely aberrated atrial premature beats, fusion of paced and normal beats, unclassifiable beats, isolated QRS-like artifacts, and lastly ventricular flutter waves. It is normal and good to have missed QRS like artifacts since they are not actually heart beats and it is understandable to have missed the unclassifiable beats as they are also disfigured. Also ventricular flutter wave doesnt have the characteristic QRS shape, which makes it hard to detect and all the regular detection algorithms ignore it due to that. However, since it is quite important there

Table 4.1: Detection Accuracy per Beat Type

Beat Type	Undetected Beat Percentage
Normal Beat	3.44
Left Bundle Branch Block Beat	0.19
Right Bundle Branch Block Beat	1.42
Atrial Premature Beat	0.24
Aberrated Atrial Premature Beat	54
Nodal (junctional) Premature Beat	0
Supraventricular Premature Beat	0
Premature Ventricular Contraction	3.87
Fusion of Ventricular and Normal Beat	2.49
Atrial Escape Beat	0
Nodal (junctional) Escape Beat	2.62
Ventricular Escape Beat	1.89
Paced Beat	0.08
Fusion of Paced and Normal Beat	15.38
Unclassifiable Beat	53.3
Isolated QRS-like Artifact	69.7
Ventricular Flutter Wave	46.61

are algorithms just to detect them and even with low detection rate they have been included in this study. Aberrated atrial premature beats and fusion of paced and normal beats on the other hand should have been detected, their low percentages can be attributed to low amplitude of their peaks in those cases.

In terms of classification, the results of the clustering of detected beats can be seen in Table 4.2. As mentioned before [34] is the only paper that aims for global classification such as this research. Thus, this work can truly be compared to only [34]. Just as [34], in order to evaluate the clustering, the clustering of each beat has been evaluated good or bad on whether they are the dominant beat type in a cluster. In overall classification of the beats [34] is 3% (95.5% this work and 98.5% [34]) more accurate compared to this work. However, this work is designed with the constraint of being able to be adapted to a wearable device, whereas the work in [34] uses an additional step to create an orthogonal basis for the ECG signal using Hermite functions, which is a very compute intensive method and no hardware implementations were found in literature. In addition, they are using data from two leads one of which would only be available in a hospital level 12-lead ECG machine. Another shortcoming of [34] is that they achieve these results with a fixed number of 25 clusters, whereas the results of these thesis were achieved using an average of 15 clusters and a maximum of 25.

Comparing the results from Table 4.2 to [34], we see that they also fail at detecting atrial escape beats, and supraventricular premature beats, and the results achieved for junctional premature beats are almost the same. For beat types such as normal

Table 4.2: Clustering accuracy with number of beats dominant and non-dominant in their clusters

Beat Type	Dominant	Non-Dominant	Accuracy
Normal Beat	71146	1072	98.52%
Left Bundle Branch Block Beat	7880	180	97.77%
Right Bundle Branch Block Beat	6809	347	93.8%
Atrial Premature Beat	1809	731	71.22%
Aberrated Atrial Premature Beat	15	54	21.74%
Nodal (junctional) Premature Beat	44	39	53.01%
Supraventricular Premature Beat	0	2	0%
Premature Ventricular Contraction	6110	738	89.22%
Fusion of Ventricular and Normal Beat	69	714	8.81%
Atrial Escape Beat	0	16	0%
Nodal (junctional) Escape Beat	16	207	7.17%
Ventricular Escape Beat	2	102	1.92%
Paced Beat	3580	37	98.98%
Fusion of Paced and Normal Beat	101	119	45.91%
Unclassifiable Beat	1	6	14.29%
Isolated QRS-like Artifact	3	37	7.5%
Ventricular Flutter Wave	216	36	85.71%

beats, left and right bundle branch blocks, ventricular premature beats and paced beats the results of [34] are slightly better. In case of atrial premature beats, [34] achieves almost perfect results. The shortcoming of our method results from problems with feature detection in this case. One of the most important features of an atrial premature beat is a small PR interval, however the same value is given to all the P related features of different beats, if the P wave is not detected. As a result, due to undetected or actually missing P waves of atrial premature beats, they are sometimes grouped with other beats that have P wave detection problems. When we compare the other diseases they only achieve good results in the classification of fusion of normal and paced beats, the rest of the classification accuracies are better than this work, yet also not good enough. Ventricular escape beats are also clustered better in [34], and in this work they have been clustered with sub par accuracy. However, that is due to them being put in the same cluster as ventricular flutter waves in this work. If ventricular flutter waves had been ignored as they were in [34], around 80% accuracy would have been achieved for their clustering as well, which is in fact better than [34]. The reason ventricular flutter waves were not ignored in this work, even though they were not well detected, was their importance and good result have been achieved in clustering them if not for detection. Ventricular flutter waves can lead to ventricular fibrillation, which leads to death without urgent treatment.

As the thesis focuses on the whole system the results of the whole system can be seen in Table 4.3. Of course the results are slightly worse due to missed beats in detection.

Table 4.3: Clustering Accuracy Over the Whole Database

Beat Type	Accuracy in Percentage
Normal Beat	95.13
Left Bundle Branch Block Beat	97.56
Right Bundle Branch Block Beat	93.8
Atrial Premature Beat	71.05
Aberrated Atrial Premature Beat	10
Nodal (junctional) Premature Beat	53.01
Supraventricular Premature Beat	0
Premature Ventricular Contraction	85.77
Fusion of Ventricular and Normal Beat	8.59
Atrial Escape Beat	0
Nodal (junctional) Escape Beat	6.99
Ventricular Escape Beat	1.87
Paced Beat	98.9
Fusion of Paced and Normal Beat	38.85
Unclassifiable Beat	6.67
Isolated QRS-like Artifact	2.27
Ventricular Flutter Wave	45.76

An estimate power value could be provided using the per spike energy values from the synapses in [49], and the neurons in [50]. Assuming the worst-case scenario that two beats need to be clustered each second, there is an average of 200 spikes fired by neurons every second and an average of 3000 synapses that pass the spikes. Even taking into account tSTDP this adds upto 111.62 nW. Adding the 692 nW from [25], which has a very similar feature detection as the one used in this thesis as mentioned before, we achieve 803.62 nW. This is a mere 13% of the best implemented classifier found in literature which matches the detected signals to its database for classification, and can only classify between 3 disease types [35]. In [35] 0.18 μm technology is used, has an area of 2.465 mm^2, and consumes 5.967 μW.

4.5 Analysis of Results

This section gives an analysis over the results giving specific examples over some patients. Based on the silhouette coefficients the problem was sometimes in the classification and sometimes in the detection phase. However, it has been found by manual inspection that feature selection was using all the required features for each case.

In some cases it has been observed that the final clustering of a beat had high (0.9) silhouette coefficients based on the time to spike values and low (0.1) or even negative silhouette coefficients based on the features. This shows that while the features of the beat we are looking at does not actually fit in its cluster, after passing through the network it has been fitted into that cluster. More research needs to be done to properly configure the network. Manual inspection has even shown that in some cases beats that clearly had many different features ended up in the same cluster.

Another observation was that the initial silhouette clusters, when all the same beats were assigned to their own cluster automatically, had relatively low values (0.4) based on their features. This is partly due to feature detection, as when the P and T waves are not detected properly, every beat gets the same values. This results in many beats in different clusters having the same value for certain features. It was expected that, when the silhouette coefficients based on the time to spike values were calculated, they would be higher than the values calculated for the features. The neuromorphic network was expected to properly separate different beat types further away from each other. However, this was rarely the case, in reality the silhouette coefficients remained quite close and sometimes even less.

While this all indicate that more research needs to be done on the configurations of a neuromorphic machine, it has also been found that the problems with feature detection also had a big impact. It has been mentioned that the failure of detecting P and T waves result in many features being the same among different beat types. However, even if that was not the case, there would still be problems in classification. For example the most relevant feature of an atrial premature contraction is the short PR interval, however if the P wave is not detected it can be mistaken for many other diseases. For example in patient 105, there are many premature ventricular contraction beats, however even though they are almost all in the same cluster, there are more normal beats in the cluster. This patient also has too many detections that are not actually beats, which shows that it has a very noisy signal. Looking at the cluster containing the premature ventricular contraction beats (which do not have P waves generally), all the normal beats in that cluster had a problem with P wave detection, most likely due to the noisy signal. Patient 114 also has the same problem where all the premature atrial contractions are put in the same cluster as premature ventricular contractions due to the undetected P waves. It is worth mentioning that a missing P wave is by no means the only difference between a premature ventricular and atrial contraction. However, when the P wave is missing, the only difference between them becomes the QRS width, which is not always enough for the network to differentiate between them.

One other problem faced during the testing was rhythm change in patient 201. The rhythm change resulted in regular heartbeats going from approximately 400 to 250 samples. As a result all the arrhythmias with faster heart rates resulted with a similar heart rate to regular heartbeats after the rhythm change and vice versa. Since heart rate is the most reliable feature obtained this has resulted in wrongly clustering approximately 80% of premature atrial contractions, 15% of premature ventricular contractions 90% of atrial escape beats and all the other beat types this patient has.

Another important observation is the fact that one of the best results was achieved from patient 119. Undoubtedly the main reason for this is the amount of each beat type in this patient. The patient has 444 premature ventricular contractions and 1543 regular beats, as a result of this the network sees both the possible combinations for enough times. This is not the case for most patients and most beat types, as most of the time there are only a couple examples of some beat types in a patient, which makes clustering quite difficult.

4.6 Possible Improvements for Clustering

Based on the analysis of the results from the tests and their reasons, various methods can be proposed to improve the results for clustering. The two most important steps to be taken are to first get a better understanding of the required configurations of a neuromorphic circuit and the following step is to design a better feature detection algorithm for ECG feature detection. However, the main obstacle in doing this is to have a well annotated database to evaluate the results and learn from the mistakes.

One of the greatest difficulties with any neural network is to find an optimal initial configuration. Especially in an unsupervised network the initial connections are very important since the learning is also based on them. In this work random initialization has been used, and it has been observed that modifying the random seed resulted in much variations in the overall clustering. A promising approach to finding an optimal initialization for a neural network is a genetic algorithm based on [51], which will greatly improve the results.

Another option is to change the network to a multiple step version. Even though this would complicate the design considerable by adding extra pools and configuring them, its effectiveness has been tested on patient 116. Only the heart rate has been used for the first layer and only one of the sick beats was clustered in the normal beats cluster and only one of the normal beats was clustered in the sick beats cluster. When this method seems promising, also considering the fact that only a photoplethysmogram (PPG) sensor could be used to detect heart rate for the first stage of classification, not all the diseases could be differentiated with heart rate such as fused beats, junctional beats, paced beats and so on. This makes picking which feature to use in the first level a problem.

In order to overcome the problem with rhythm change the only method is to retrain the network or have continuous learning capability. However, the problem with that is, if we continue learning while in operation, our cluster centers will evolve over time and we will not be able to interpret the results anymore. Thus, a more straight forward option is to retrain the network in a less frequent basis, which is neither a good nor a sustainable approach.

A final method to improve the clustering would be to use a semi-supervised approach in training the network for certain diseases. This might help eliminate the inefficacy of the classifier when there is a very low beat type count, or when two different beat types are quite similar to each other.

4.7 Conclusion

This chapter has explained the logic behind the evaluated network configurations and given the classification results of the system with an estimated power consumption. Even though good results have been obtained in some beat types, there is still much more that can be improved and the ultra-low power design of neuromorphic circuits provides a very promising solution. The following chapter will conclude this thesis and offer some alternatives for future work.

Conclusion 5

5.1 Conclusion

In this thesis a novel ultra-low power ECG beat type classification for arrhythmia detection has been proposed and simulated. In this system, a slightly modified version of an already implemented feature detection has been used and it has been incorporated with an unsupervised feature selection algorithm. The clustering of the beats has been achieved using a neuromorphic machine.

Considering the shortcomings introduced by the system being a global classifier and the aim to design a low power wearable system, it has achieved remarkable results compared to the literature. The system as a whole is estimated to consume less than 1 μW and has an overall clustering accuracy of 95.5%. In addition to 6 other well clustered beat types it can also cluster the detected ventricular flutter waves with 85% accuracy, which are crucial since they can lead to ventricular fibrillation and death.

In summary, this thesis provides the preliminary work for a very promising design for wearable ECG classification in the future.

5.2 Future Work

As mentioned throughout the thesis, there are some clear steps that the clustering would benefit from. Since neuromorphic computing is a newly flourishing research area, work in the understanding of this system will help improve the results for this system.

Another important topic for future research would be to use different feature selection methods such as embedded feature selection, which uses the classifier as part of the feature selection process. As a result optimal features for a given classifier can be selected. However, this research would also benefit from a better understanding of neuromorphic machines.

Research on using multiple leads for classification will also help improve the results and will provide insight into sensor fusion in neuromorphic machines. However, both this, and developments in feature detection algorithms require a new database to be developed.

It has been mentioned that the existing feature detection algorithms have proved

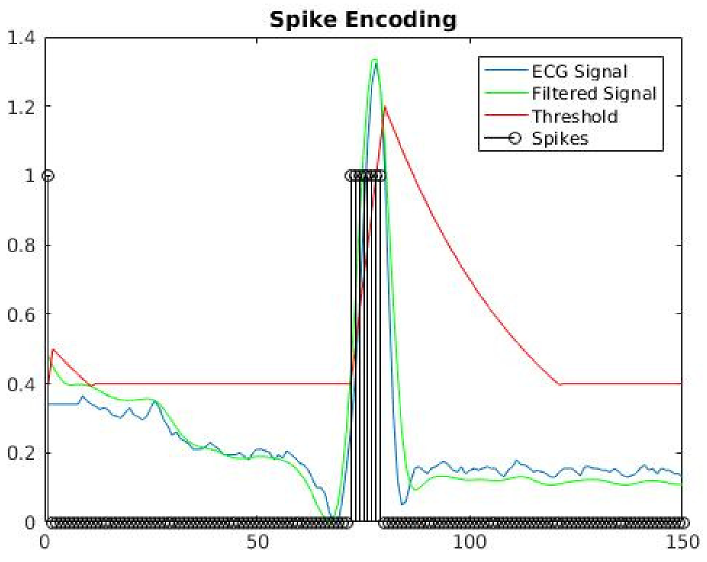

Figure 5.1: ECG Signal Spike Encoding

insufficient. Moreover, taking into consideration that the most power consuming part of the system is the feature detection, a neuromorphic design for feature detection is proposed. This proposal offers an interesting potential research topic, however it has not been researched further in this thesis.

This encoding scheme is inspired by [52], which uses spike encoding to detect speech edges. The ultra-low power design of spiking systems makes it a good candidate for speech edge detection in mobile systems. The element they have used is very much like an integrate and fire neuron, with the exception of increasing its threshold after each spike. This has been done to follow the noise and prevent misinterpreting noise with speech. In case there is a signal with a high slope, the threshold does not increase quickly enough and multiple spikes occur. This, is used to understand the presence of speech.

This logic could just as well be applied to an ECG signal. An example can be seen in Figure 5.1. For this implementation, it is decided that if there are 6 consecutive spikes at any point in time, there is a QRS complex there. When tried on the MIT BIH database this simple method achieves 94.77% true positive rate and 97.23% positive predictive value, when we remove the patients without the ML-II leads (since the other leads do not have the same characteristic QRS complex the algorithm is not optimized for that, therefore the data with other leads have been removed). These results are achieved without any other optimization or noise level adaptive thresholding. Thus it is possible to improve on them. However, this algorithm only gives us information about the presence of a beat and nothing about the onset and end points of the QRS complex at its current stage.

Of course the added benefit of this algorithm is that it encodes the ECG signal directly into spikes, which of course should be the end goal, to remove all the conventional circuits. When the classifier is more advanced with small modifications to this algorithm, a direct use of it without any other feature detection might be possible, since the frequency of the spikes also gives valuable data of the shape of the QRS complex as well. If the sampling is increased, such values could be picked for the capacitor (which determines the threshold in this algorithm) that, the threshold can also follow the downward slope of the QRS complex, without also spiking at every noise component.

Bibliography

[1] J. Pan and W. J. Tompkins, "A real-time qrs detection algorithm," *IEEE Transactions on Biomedical Engineering*, no. 3, pp. 230–236, 1985.

[2] S. M. Bohte, J. N. Kok, and H. La Poutre, "Error-backpropagation in temporally encoded networks of spiking neurons," *Neurocomputing*, vol. 48, no. 1, pp. 17–37, 2002.

[3] T. Tekeste, N. Bayasi, H. Saleh, A. Khandoker, B. Mohammad, M. Al-Qutayri, and M. Ismail, "Adaptive ecg interval extraction," in *Circuits and Systems (ISCAS), 2015 IEEE International Symposium on*, pp. 998–1001, IEEE, 2015.

[4] "OpenStax College how neurons communicate." https://cnx.org/contents/cs_Pb-GW@5/How-Neurons-Communicate. Accessed: 2017-09-30.

[5] J. Sjstrm and W. Gerstner, "Spike-timing dependent plasticity," *Scholarpedia*, vol. 5, no. 2, p. 1362, 2010. revision #184913.

[6] G.-q. Bi and M.-m. Poo, "Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type," *Journal of Neuroscience*, vol. 18, no. 24, pp. 10464–10472, 1998.

[7] J.-P. Pfister and W. Gerstner, "Triplets of spikes in a model of spike timing-dependent plasticity," *Journal of Neuroscience*, vol. 26, no. 38, pp. 9673–9682, 2006.

[8] W. Zong, G. Moody, and D. Jiang, "A robust open-source algorithm to detect onset and duration of qrs complexes," in *Computers in Cardiology, 2003*, pp. 737–740, IEEE, 2003.

[9] R. Lozano, M. Naghavi, K. Foreman, S. Lim, K. Shibuya, V. Aboyans, J. Abraham, T. Adair, R. Aggarwal, S. Y. Ahn, *et al.*, "Global and regional mortality from 235 causes of death for 20 age groups in 1990 and 2010: a systematic analysis for the global burden of disease study 2010," *The Lancet*, vol. 380, no. 9859, pp. 2095–2128, 2013.

[10] C. Mead, "Neuromorphic electronic systems," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1629–1636, 1990.

[11] D. O. Hebb, *The organization of behavior: A neuropsychological theory*. Psychology Press, 2005.

[12] V. R. Kompella, M. Luciw, and J. Schmidhuber, "Incremental slow feature analysis: Adaptive low-complexity slow feature updating from high-dimensional input streams," *Neural Computation*, vol. 24, no. 11, pp. 2994–3024, 2012.

[13] Z. K. Malik, A. Hussain, and J. Wu, "Novel biologically inspired approaches to extracting online information from temporal data," *Cognitive Computation*, vol. 6, no. 3, pp. 595–607, 2014.

[14] Y. Choe, "Anti-hebbian learning," in *Encyclopedia of Computational Neuroscience*, pp. 191–193, Springer, 2015.

[15] J. Brownlee, "An introduction to feature selection." https://machinelearningmastery.com/an-introduction-to-feature-selection/. Accessed: 2017-09-30.

[16] S. H. Jambukia, V. K. Dabhi, and H. B. Prajapati, "Classification of ecg signals using machine learning techniques: A survey," in *Computer Engineering and Applications (ICACEA), 2015 International Conference on Advances in*, pp. 714–721, IEEE, 2015.

[17] G. B. Moody and R. G. Mark, "The mit-bih arrhythmia database on cd-rom and software for use with it," in *Computers in Cardiology 1990, Proceedings.*, pp. 185–188, IEEE, 1990.

[18] J. Rodríguez-Sotelo, E. Delgado-Trejos, D. Peluffo-Ordóñez, D. Cuesta-Frau, and G. Castellanos-Domínguez, "Weighted-pca for unsupervised classification of cardiac arrhythmias," in *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, pp. 1906–1909, IEEE, 2010.

[19] S. Mukhopadhyay and G. Ray, "A new interpretation of nonlinear energy operator and its efficacy in spike detection," *IEEE Transactions on Biomedical Engineering*, vol. 45, no. 2, pp. 180–187, 1998.

[20] C. Li, C. Zheng, and C. Tai, "Detection of ecg characteristic points using wavelet transforms," *IEEE Transactions on Biomedical Engineering*, vol. 42, no. 1, pp. 21–28, 1995.

[21] J. P. Martínez, R. Almeida, S. Olmos, A. P. Rocha, and P. Laguna, "A wavelet-based ecg delineator: evaluation on standard databases," *IEEE Transactions on Biomedical Engineering*, vol. 51, no. 4, pp. 570–581, 2004.

[22] P. Laguna, R. Jané, and P. Caminal, "Automatic detection of wave boundaries in multilead ecg signals: Validation with the cse database," *Computers and Biomedical Research*, vol. 27, no. 1, pp. 45–60, 1994.

[23] G. A. Laguna P, Mark RG and M. GB, "A database for evaluation of algorithms for measurement of qt and other waveform intervals in the ecg," in *Computers in Cardiology 1997, Proceedings.*, pp. 673–676, IEEE, 1997.

[24] Y. Sun, K. L. Chan, and S. M. Krishnan, "Characteristic wave detection in ecg signal using morphological transform," *BMC Cardiovascular Disorders*, vol. 5, no. 1, p. 28, 2005.

[25] N. Bayasi, T. Tekeste, H. Saleh, B. Mohammad, and M. Ismail, "A 65-nm low power ecg feature extraction system," in *Circuits and Systems (ISCAS), 2015 IEEE International Symposium on*, pp. 746–749, IEEE, 2015.

[26] I. Guyon and A. Elisseeff, *An Introduction to Feature Extraction*, pp. 1–25. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006.

[27] J. Mes, "Design Space Exploration of the Unsupervised Spiking Neural Network Configuration problem Using a Custom Made Synthesizable SNN Simulation Framework," Master's thesis, TU Delft, the Netherlands, 2018, in preparation.

[28] B. Schrauwen and J. Van Campenhout, "Bsa, a fast and accurate spike train encoding scheme," in *Neural Networks, 2003. Proceedings of the International Joint Conference on*, vol. 4, pp. 2825–2830, IEEE, 2003.

[29] C. Wen, M.-F. Yeh, and K.-C. Chang, "Ecg beat classification using greyart network," *IET Signal Processing*, vol. 1, no. 1, pp. 19–28, 2007.

[30] D. Azariadi, V. Tsoutsouras, S. Xydis, and D. Soudris, "Ecg signal analysis and arrhythmia detection on iot wearable medical devices," in *Modern Circuits and Systems Technologies (MOCAST), 2016 5th International Conference on*, pp. 1–4, IEEE, 2016.

[31] N. Bayasi, T. Tekeste, H. Saleh, B. Mohammad, A. Khandoker, and M. Ismail, "Low-power ecg-based processor for predicting ventricular arrhythmia," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 5, pp. 1962–1974, 2016.

[32] D. Cuesta-Frau, M. O. Biagetti, R. A. Quinteiro, P. Mico-Tormos, and M. Aboy, "Unsupervised classification of ventricular extrasystoles using bounded clustering algorithms and morphology matching," *Medical & Biological Engineering & Computing*, vol. 45, no. 3, pp. 229–239, 2007.

[33] P. De Chazal, M. O'Dwyer, and R. B. Reilly, "Automatic classification of heartbeats using ecg morphology and heartbeat interval features," *IEEE Transactions on Biomedical Engineering*, vol. 51, no. 7, pp. 1196–1206, 2004.

[34] M. Lagerholm, C. Peterson, G. Braccini, L. Edenbrandt, and L. Sornmo, "Clustering ecg complexes using hermite functions and self-organizing maps," *IEEE Transactions on Biomedical Engineering*, vol. 47, no. 7, pp. 838–848, 2000.

[35] S.-Y. Lee, J.-H. Hong, C.-H. Hsieh, M.-C. Liang, S.-Y. C. Chien, and K.-H. Lin, "Low-power wireless ecg acquisition and classification system for body sensor networks," *IEEE Journal of Biomedical and Health Informatics*, vol. 19, no. 1, pp. 236–246, 2015.

[36] Q. Long, Y. Ren, J. Han, and X. Zeng, "Vlsi implementation for r-wave detection and heartbeat classification of ecg adaptive sampling signals," in *Solid-State and Integrated Circuit Technology (ICSICT), 2016 13th IEEE International Conference on*, pp. 1597–1599, IEEE, 2016.

[37] J. L. Rodríguez-Sotelo, D. Cuesta-Frau, and G. Castellanos-Dominguez, "Unsupervised classification of atrial heartbeats using a prematurity index and wave morphology features," *Medical & Biological Engineering & Computing*, vol. 47, no. 7, pp. 731–741, 2009.

[38] Y. Sun, K. Chan, S. Krishnan, and D. Dutt, "Unsupervised classification of ecg beats using a mlvq neural network," in *Engineering in Medicine and Biology Society, 2000. Proceedings of the 22nd Annual International Conference of the IEEE*, vol. 2, pp. 1387–1390, IEEE, 2000.

[39] A. Walinjkar and J. Woods, "Personalized wearable systems for real-time ecg classification and healthcare interoperability: Real-time ecg classification and fhir interoperability," in *Internet Technologies and Applications (ITA), 2017*, pp. 9–14, IEEE, 2017.

[40] S. Mallat, "Zero-crossings of a wavelet transform," *IEEE Transactions on Information Theory*, vol. 37, no. 4, pp. 1019–1033, 1991.

[41] E. Burns, "Life In The Fast Lane: ecg library." `https://lifeinthefastlane.com/ecg-library/`. Accessed: 2017-10-15.

[42] "ECGPEDIA introduction to arrhythmias." `http://en.ecgpedia.org/index.php?title=Introduction_to_Arrhythmias`. Accessed: 2017-10-15.

[43] G. Bebis and M. Georgiopoulos, "Feed-forward neural networks," *IEEE Potentials*, vol. 13, no. 4, pp. 27–31, 1994.

[44] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.

[45] G. M. Wojcik and W. A. Kaminski, "Liquid state machine built of hodgkin–huxley neurons and pattern recognition," *Neurocomputing*, vol. 58, pp. 245–251, 2004.

[46] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, 1987.

[47] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[48] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1027–1035, Society for Industrial and Applied Mathematics, 2007.

[49] X. You, "Full-Custom Multi-Compartment Synaptic Circuits in Neuromorphic Structures," Master's thesis, TU Delft, the Netherlands, 2017.

[50] E. Stienstra, "A 32 x 32 Spiking Neural Network System On Chip," Master's thesis, TU Delft, the Netherlands, 2017.

[51] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary computation*, vol. 10, no. 2, pp. 99–127, 2002.

[52] D. Du and K. Odame, "A bio-inspired ultra-low-power spike encoding circuit for speech edge detection," in *Biomedical Circuits and Systems Conference (BioCAS), 2011 IEEE*, pp. 289–292, IEEE, 2011.

[53] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *The Journal of Physiology*, vol. 117, no. 4, pp. 500–544, 1952.

[54] E. M. Izhikevich, "Simple model of spiking neurons," *IEEE Transactions on Neural Networks*, vol. 14, no. 6, pp. 1569–1572, 2003.

[55] L. Lapique, "Recherches quantitatives sur l'excitation electrique des nerfs traitee comme une polarization.," *Journal of Physiology and Patholol gy*, vol. 9, pp. 620–635, 1907.

[56] L. I. Zhang, H. W. Tao, C. E. Holt, W. A. Harris, and M.-m. Poo, "A critical window for cooperation and competition among developing retinotectal synapses," *Nature*, vol. 395, no. 6697, pp. 37–44, 1998.

A

This chapter aims at giving a brief introduction to neuromorphic circuits, their biological background and the simulator model used in this thesis.

A.1 Neuromorphic Machine

As mentioned before neuromorphic machines have been introduced by [10]. They aim to emulate the brain behavior and thus, consist of so-called neurons, synapses and learn based on synaptic plasticity algorithms. This section will explore the biological phenomenon and some models used in literature to emulate this system. Then the simulation model of a neuromorphic model that has been created by Johan Mes and used in this thesis will be discussed [27].

A.1.1 Neuron

Neurons are the basic functional units of the brain. They are in charge of signal processing and transmission, which is done by both electrical and chemical signals.

Neurons consist of three main parts, namely *soma, dendrites* and *axon.* These parts can be seen in Figure 1.1. Soma is the body of the neuron and holds the nucleus. The small branches coming out of the soma are called the dendrites, they are used to receive the signals coming from other neighboring neurons. Long branch coming out of the neuron is called the axon. When the inputs coming from the dendrites are large enough the neuron is stimulated and an electrical spike is created, which is carried through the axon to other neurons.

The activation of a neuron is based on the inputs it receives from the dendrites. There can be two types of currents coming from the dendrites: excitatory currents and inhibitory currents. These currents can respectively induce potentiation and excitation of the membrane voltage. When the membrane voltage is higher than a certain threshold the neuron fires. Whereas in some cases one stimulus is enough to fire a neuron, in most cases multiple neurons act together to fire a neuron. The created pulse, when the membrane voltage passes the threshold voltage is irrespective of the difference between the threshold voltage and the stimulating signals. The created pulse once the threshold is surpassed always has the same height and it is called a spike. After the firing the neuron enters a resting phase called *refractory period* which has even lower potential than its resting state. These phenomenon can be better explained in Figure A.1 and Figure A.2.

There have been many mathematical models explored in literature and they will be

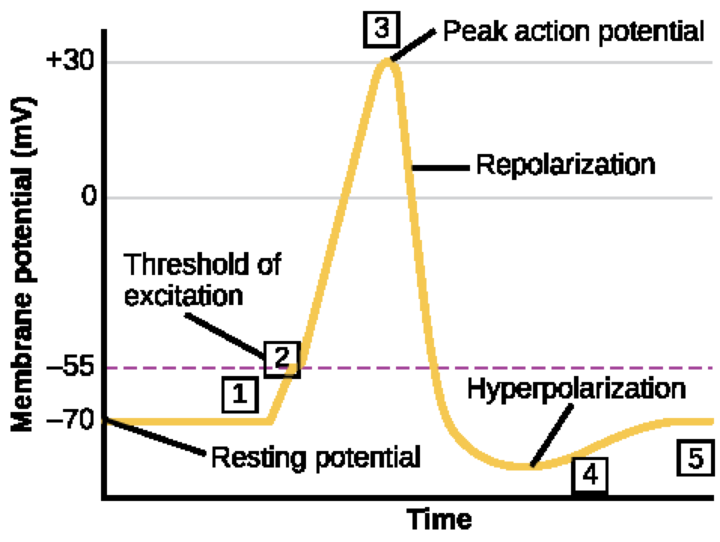

Figure A.1: Elaboration of Action Potentials [4]. Point 1 shows where the neuron cell is in its resting state and starts to get excited. In point 2 the membrane potential crosses the threshold and the cell spikes (depolarizes). It reaches its peak value on point 3 and starts to repolarize. Point 4 shows the refractory period, the hyperpolarized state of the cell and finally on 5 the cell returns to its original resting state.

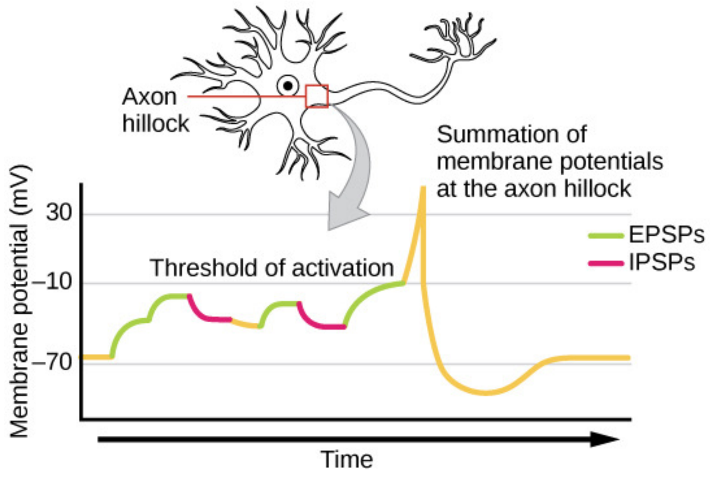

Figure A.2: Signal Summation in Neurons [4]. The jumps in the figure represent excitatory inputs to the cell where the membrane potential increases and conversely the drops represent inhibitory inputs to the cell, which decrease the membrane potential. The neuron accumulates these signals, charges and discharges, until enough stimuli has been acquired to pass the threshold voltage and it spikes.

discussed in section A.1.3.1.

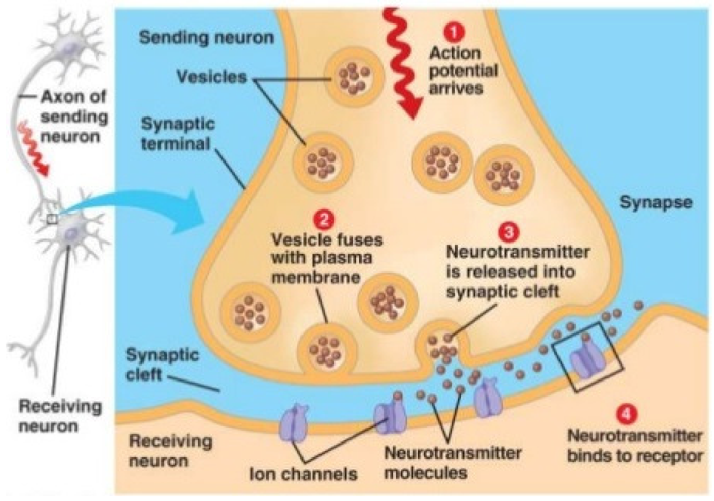
Sending neuron
Vesicles
Axon of sending neuron
Synaptic terminal
1 Action potential arrives
Synapse
2 Vesicle fuses with plasma membrane
3 Neurotransmitter is released into synaptic cleft
Synaptic cleft
Receiving neuron
Receiving neuron
Ion channels
Neurotransmitter molecules
4 Neurotransmitter binds to receptor

Figure A.3: Synapse structure

A.1.2 Synapse

Synapse is an abstract concept that defines the gap between the end of a neuron's axon and the end of another neuron's dendrite, which can also be seen in Figure 1.1. Its biological application is to prevent a short circuit between the neurons. If the neurons were directly connected one created pulse would go through the whole network and depolarize everything. Thus, the synapse comes into play in order to prevent this.

The electrical signal of a spike is converted to a chemical signal when being transfered to another neuron. This transfer is achieved by using the elements called *neurotransmitters*. When action potential reaches an axon terminal, the neurotransmitters that are stored in small bags called *synaptic vesicles* are released into the *synaptic cleft*. Neurotransmitters diffuse towards the dendrites and bind to their corresponding receptors on dendrite terminals. This process allows the ions created by the action potential to flow in between and create action potentials in the postsynaptic neurons. This process can be better observed in Figure A.3.

The learning process in the brain is attributed to the synapse. In each connection between neurons (each synapse), there are "synaptic weights". This concept is what learning is based on. The weights of each synapse vary in time and this adaptation actualizes learning. This weight adaptation is called *synaptic plasticity*. The most basic logic behind the weight adaptation has been introduced by Donald Hebb [11]. Even before we knew anything about plasticity Hebb postulated that if a neuron contributes to another neurons firing the connection between them will get stronger over time. This idea has been further improved over time and mathematical models have been developed, which will be discussed in section A.1.3.2.

A.1.3 Models

This section will discuss the various mathematical models created to realize and understand neurons and synaptic plasticity.

A.1.3.1 Neuron Models

The most detailed neuron model has been created by Hodgkin and Huxley in 1952 [53]. The model is extremely detailed and gives a perfect representation of the spike response of a neuron. However, the details come with a price and the model is too complicated to simulate large networks.

Two other methods have been widely utilized that are more simplified and can be used in simulating large networks. These two models are known as the Izhikevich [54] and the leaky integrate and fire (which was first proposed in [55] and since then small changes have been made) neuron models. Izhikevich model is purely based on mathematics; it is a simplified version of the Hodgkin-Huxley model and can create a realistic spiking behavior. The leaky integrate and fire neuron on the other hand is a simple RC circuit. Therefore, the membrane voltage is defined by the linear differential equation of this circuit and there is a threshold for spiking.

A.1.3.2 Synaptic Plasticity Models

As mentioned before learning in the brain is achieved by synaptic plasticity. Furthermore, it is essential for us to know the details behind the learning since, it helps us make sense of the system and configure it accordingly. Hebb was basically correct in the working principle of the system, when two neurons fire consecutively the strength of the connection between them changes based on the timing of those spikes. The spike that comes from the neuron before the synapse is named pre-spike, on the other hand the one that occurs on the neuron after the synapse is called a post-spike. Thus, the naming is a location based naming rather than a temporal one. If the pre-spike comes to the synapse before the post spike the connection weight is increased, but if the adverse happens the weight is decreased. Continues repetition of one of these events causes long term effects which are called Long Term Potentiation (LTP) and Long Term Depression (LTD) respectfully.

Further studies have mathematically modeled this behavior on neurons [6], [56]. It is named Pair-based Spike Timing Dependent Plasticity (PSTDP). In Figure A.4 the phenomenon can be seen. Two constants affect this behavior namely τ the time coefficient that determines the time window and A the amplitude coefficient that determines the change in weight. The expression for PSTDP can be seen in the following equations:

$$\Delta w^+ = A^+ * e^{-\Delta t/\tau_+} \qquad \Delta t > 0 \tag{A.1}$$

$$\Delta w^- = -A^- * e^{\Delta t/\tau_-} \qquad \Delta t < 0 \tag{A.2}$$

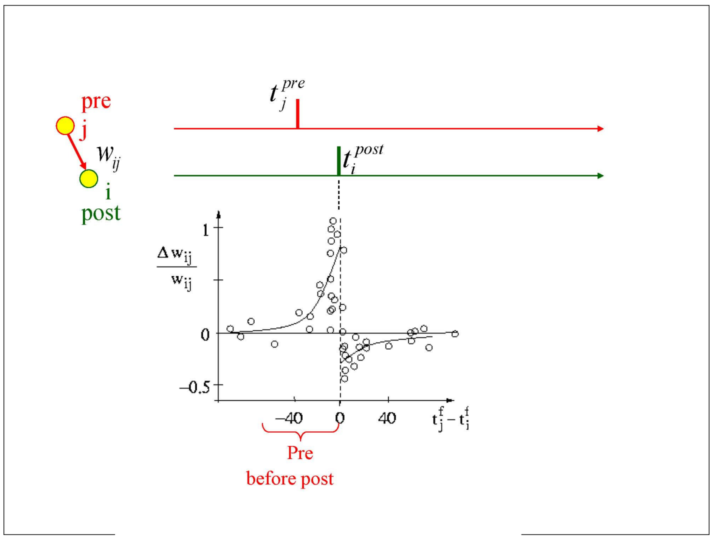

Figure A.4: PSTDP Learning Window Schematic [5]. The circles on the graph represent the experimental data of the percentage change of excitatory post synaptic current at 20-30 min after repetitive stimulation of pre and post-spikes at 1 Hz [6]. An exponential curve fitting on this data is also shown.

where Δt is the time difference between the pre and post spikes. τ_+ and τ_- are as mentioned time constants for potentiation and depression respectively, whereas A^+ and A^- represent the maximum amplitudes for the weights in both ends. These parameters determine the area under the weight update curve. For stable learning the area under the depression curve should be greater than the one under the potentiation curve. In the antagonistic case the network becomes an all pass network as a result of all the connections being highly potentiated.

Studies have further shown that when the stimulation frequency was increased PSTDP couldn't predict the weight change correctly. When we use a higher frequency the second pre-spike arrives too soon and is in the depression window or vice versa, however experiments have shown the contrary. As a result a new model had to be developed for such a case. This method is called Triplet-based STDP (TSTDP). The following equations are used to express the rule:

$$\Delta w^+ = e^{-\Delta t_1/\tau_+} * (A_+^2 + A_+^3 * e^{-\Delta t_2/\tau_y}) \tag{A.3}$$

$$\Delta w^- = -e^{\Delta t_1/\tau_-} * (A_-^2 + A_-^3 * e^{-\Delta t_3/\tau_x}) \tag{A.4}$$

where τ_+,τ_-,τ_x and τ_y are again the time constant coefficients. However, in this case whether there will be potentiation or depression is slightly more complicated, Figure A.5 clarifies this situation.

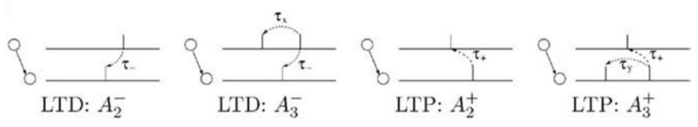

Figure A.5: TSTDP time constant coefficient interpretation. A_-^2 and A_+^2 are the pair STDP cases as can be seen. In the case of triplets though, for depression the presence of a pre-spike gives an additional contribution just as for potentiation the presence of an additional post-spike gives a contribution. For the third spike to be included the τ_x shouldn't be much larger than τ_- or for potentiation τ_y shouldn't be much larger than τ_+ [7].

A.1.4 Simulation Model

A simulation model has been created by Johan Mes [27] to emulate a spiking neural network. This section will briefly go over some of the decisions made over the creation of this simulation tool, and available configuration options for the created network.

The simulator uses integrate and fire model for the neurons. The leading factors that lead to this choice were the nonlinear behavior of the integrate and fire neuron and its mathematical simplicity. The Hodgkin-Huxley model is unnecessarily complicated for such a task even though it can model a biological neuron very well. The Izhikevich model on the other hand can model a neuron better than an integrate and fire model in terms of spike shape yet it fails to exhibit a nonlinear response (i.e. its spiking frequency increases linearly based on the input strength). Thus, it causes problems for nonlinear classification.

TSTDP is being used as the learning rule in the simulator. Mainly due to its use in high frequency spiking. TSTDP functioning in high frequency allows the overall simulation time to drop. In a network that is slow enough PSTDP should be enough, which might be considered in the hardware implementation, since one of the drawbacks of TSTDP is its complexity. However, the simulation duration is quite important not just because a computer simulation takes much longer than a hardware implementation would, but also as the simulation duration gets longer we need to store more points in the simulation which causes memory problems. It is worth mentioning that since TSTDP is just an extension of PSTDP, by changing input configurations it is possible to simulate PSTDP as well.

The most important configurations for the simulator can be grouped into three categories: Network Structure, Input/Output, and Basics. The network structure is completely modifiable in the simulator. It is possible to change the neuron count, sparsity of connections between the neurons, weight limits and distributions in synapses, delay limits and distributions of synapses. Finally it is also possible to customize the network architecture (Feed-forward [43], Self Organizing Map (SOM) [44], Liquid State Machine (LSM) [45] and more custom networks), and add different types of feedbacks to the network. In the Input/Output section other than the input, what we can configure are the sampling frequency from the input, input encoding style, input training time, number of epochs for input training. Finally, we can also change the basic settings of the whole network, such as the capacitances and resistances of neurons to have a control over discharge time, and we can change the parameters of the learning, as mentioned before. This helps us use triplet or pair based STDP, it is also possible to change the areas under the curves of depression and potentiation.