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

CAS-2018-42

M.Sc. Thesis

Design Space Exploration of a Neuromorphic ECG Classification System using a Spiking Self-Organizing Map

Johan Mes B.Sc.

Abstract

The Self-Organizing Map (SOM) is an unsupervised neural network topology that incorporates competitive learning for the classification of data. In this thesis we investigate the design space of a system incorporating such a topology based on Spiking Neural Networks (SNNs), and apply it to classifying electrocardiogram (ECG) beats.

We present novel insights into the characterization of the SOM and its encapsulating system by exploring configuration parameters such as learning rate, neuron models, potentiation and depression ratios, and synaptic conductivity parameters by performing high-level architectural simulations of the system whose SNN is developed with the aim of being implemented using power efficient neuromorphic hardware.

Due to the amount of manual work needed to monitor and analyze ECG signals when diagnosing cardiovascular problems, and because it is the leading cause of death in the world, an automated, realtime, and low power detection & classification system is essential.

Unsupervised and in realtime, this system performs beat detection with an average True Positive Rate (TPR) of 99.10% and a Positive Predictive Value (PPV) of 99.58% and classification of 500 detected beats with a Multidimensional Scaling Error (E_{MDS}) of 0.0169 and a beat recognition percentage of 100%.

TU Delft

Design Space Exploration of a Neuromorphic ECG Classification System using a Spiking Self-Organizing Map

Thesis

submitted in partial fulfillment of the
requirements for the degree of

Master of Science

in

Microelectronics

by

Johan Mes B.Sc.
born in Delft, The Netherlands

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

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 **"Design Space Exploration of a Neuromorphic ECG Classification System using a Spiking Self-Organizing Map"** by **Johan Mes B.Sc.** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: 2018-09-28

Chairman: _____
prof.dr.ir. A.J. van der Veen

Advisor: _____
dr.ir. T.G.R.M. van Leuken

Committee Members: _____
dr.ir. Z. Al-Ars

_____
dr.ir. S.S. Kumar

_____
dr. A. Zjajo

Abstract

The Self-Organizing Map (SOM) is an unsupervised neural network topology that incorporates competitive learning for the classification of data. In this thesis we investigate the design space of a system incorporating such a topology based on Spiking Neural Networks (SNNs), and apply it to classifying electrocardiogram (ECG) beats.

We present novel insights into the characterization of the SOM and its encapsulating system by exploring configuration parameters such as learning rate, neuron models, potentiation and depression ratios, and synaptic conductivity parameters by performing high-level architectural simulations of the system whose SNN is developed with the aim of being implemented using power efficient neuromorphic hardware.

Due to the amount of manual work needed to monitor and analyze ECG signals when diagnosing cardiovascular problems, and because it is the leading cause of death in the world, an automated, realtime, and low power detection & classification system is essential.

Unsupervised and in realtime, this system performs beat detection with an average True Positive Rate (TPR) of 99.10% and a Positive Predictive Value (PPV) of 99.58% and classification of 500 detected beats with a Multidimensional Scaling Error (E_{MDS}) of 0.0169 and a beat recognition percentage of 100%.

Acknowledgments

First and foremost I would like to thank my advisor dr.ir. T.G.R.M. van Leuken for providing me with the opportunity for doing this project. It allowed me to become familiar and delve deeper into the research field called neuromorphic computing, exploring concepts that would otherwise be inaccesible, and may even see widespread use in the future.

Also without him, and many others, I would not have been able to publish my first paper and present it at the SAMOS conference.

Second, I would like to thank my daily supervisors dr.ir. S.S. Kumar and dr. A. Zjajo for their indispensable support that finally allowed me to write the document you are reading right now.

Third, I would like to thank my M.Sc. student colleagues for lightening the mood, co-authorship of the paper, and other essential assistance in room HB17.090 through the months or even years: most importantly Eralp Kolagasioglu for his work put into the product that this document covers, and Xuefei You for co-authorship of my first paper, company during the SAMOS conference trip, and Chinese culture lessons.

Finally, I would like to thank the technical support staff of Circuits and Systems, namely Antoon Frehe and Wim Tiwon, for providing me with their low latency and high reliability support. I am also thankful for them supplying the high quality hardware and software tools required to complete this project.

Johan Mes B.Sc.
Delft, The Netherlands
2018-09-28

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

Neuromorphic computing introduced by Carver Mead presents a novel information processing method based on biological processes [13]. When implemented on simple low power specialized hardware it promises a high energy efficiency approach to many computational problems that today can be solved with conventional digital computer architectures like Harvard and Von Neumann implemented using standard CMOS devices.

While neuromorphic computing can also be performed on processors based on these conventional computer architectures, as will be done for this thesis, the goal is to limit ourselves to emulating the specialized hardware so that the essential parts of the system architecture can later be conveniently translated to using the mentioned specialized hardware.

Computing that is inspired by biological phenomena is not new. An older method called the Artifical Neural Network (ANN), often combined with backpropagation, has proven to be an effective information processing method especially in fields where the computational task to be performed is not clear cut like in the field of recognizing and classifying patterns in data formats like images and voice. ANNs mimic biological processing methods like neural networks but on a more abstract level.

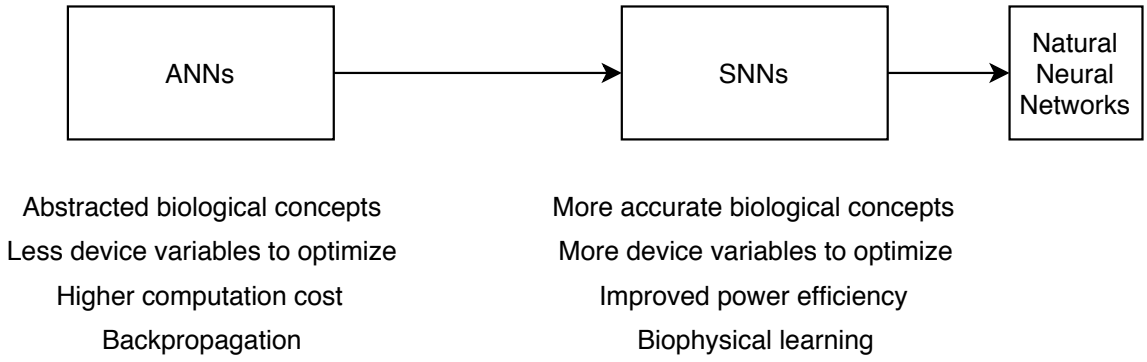

Figure 1.1: ANNs and SNNs can be seen as abstraction steps from actual biological learning

The Spiking Neural Network (SNN) is a more recent concept from the neuromorphic computing field. This network type mimics biophysical properties more accurately and uses computation blocks that can be conveniently implemented on custom mixed signal hardware, as shown in the diagram in Figure 1.1. This implementability is the key to the promise of achieving drastically improved computational efficiency.

Like ANNs, the SNN is an excellent candidate for processing unlabeled badly defined data where the exact computation to be performed is not immediately obvious. For

both types this processing incorporates learning: if configured correctly they can adapt their computational behaviour to input data to generate an arbitrary preferred function from input to output. The main difference between learning algorithms of ANNs like backpropagation [14] is that learning is based on biological models implementing Hebbian learning like Spike Timing Dependent Plasticity (STDP).

1.1 Problem Statement

Currently in literature various VLSI implementations and biological models of individual concepts of neuromorphic computing exist but there is a limited amount of content available that investigates how to properly configure a full classification or categorization device incorporating an SNN for a specific use case. There is also limited information available about characterization: SNN behaviour changes as various essential configuration parameters are modified.

To fill this gap in the state of the art a Design Space Exploration (DSE) will be performed on an SNN device that is tasked to categorize electrocardiogram (ECG) beats using features like the ones presented in Figure 1.2. This categorization will be performed by a spiking Self-Organizing Map (SOM) topology. Configuration parameters like neuron models, network learning rate, network size, input dataset, and input dataset processing will be investigated. When implementing the SNN special care is taken to keep neuromorphic hardware implementability in mind which is one of the main reasons for investigating this SNN approach.

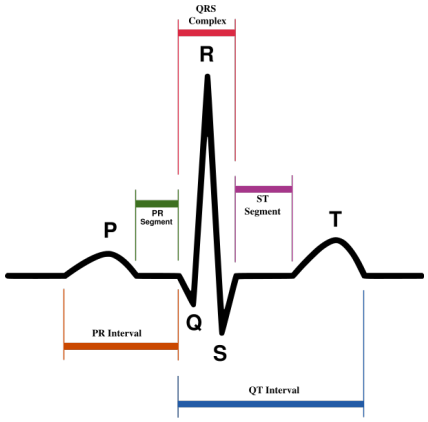

Figure 1.2: ECG beat features used for categorization [1] of beats

This categorization task is a good example of a computational task that is ill-defined: it is not immediately clear what the exact approach would be beforehand. Therefore tools like ANNs and SNNs that can approximate an almost universal variety of functions are good choices to tackle these tasks. When comparing these kinds of tools the SNN promises superior power efficiency, making it an excellent choice for this problem.

1.2 Goals

In this thesis the main goals that are presented are to:

- Develop the software architecture of a SNN based classifier, and all other computation blocks required to allow end users to utilize the SNN concept.
- Develop an SNN that can be realized as a physical hardware block based on the specialized neuromorphic hardware mentioned earlier.
- Develop that device such that all steps from raw ECG data feeding into the network to SOM output mapping analysis are fully automated and can be performed faster and preferably more accurate than manually or competing automation techniques.
- Present results and discussion of the behaviour changes of this device as various essential configuration parameters are modified in the form of a DSE.
- Combining all of that, discuss and conclude if an SNN based neuromorphic device can be applied to comparable tasks and if it is able to provide the promised power savings compared to current techniques.

1.3 Contributions

The main contributions of this thesis are:

- A full system architecture for a device that can categorize ECG signals using unsupervised SNNs based the SOM topology.
- An easy to use outer MATLAB configuration interface of this device, making it possible to perform fast exploration and optimization, that can be used to generate any kind of SNN layout with STDP and apply it to almost any kind of dataset.
- A synthesizable inner SNN block implemented in C that is able to simulate arbitrary SNN layouts orders of magnitude faster than biological time. It bypasses inefficiencies in MATLAB where performance matters and is hidden from the end user.
- A detailed presentation of beat recognition and mapping error changes as a result of changing various essential configuration options.
- An optimal configuration option set that makes the device capable of recognizing and positioning 500 consecutive real-time ECG beats from the MIT-BIH dataset, using an unsupervised Self-Organizing Map with STDP. The final mapping error defined by the E_{MDS} metric is equal to 0.0169.

1.4 Outline

In Chapter 2 the biological background of the components inside the SNN is discussed. In Chapter 3 the architecture of the device incorporating the SNN is presented. In Chapter 4 the design space exploration of the configurable components in the device is given. In Chapter 5 the conclusion and future work are presented.

2 Biological Background

In this chapter a thorough introduction of the biological background of both neuromorphic computing and the application use case is given. The background of neuromorphic computing covers the neural mechanisms in the brain that are modeled in a simplified manner allowing them to be used in artificial information processing.

In Section 2.1 the neuron will be discussed. This can be considered as the information processing and generating unit or mathematical graph vertex V of neuromorphic networks. In Section 2.2 the synapse will be discussed. This complements the neuron and functions as the information transfer unit and can be considered as the mathematical graph edge E of neuromorphic networks. Combining these to into a graph g(V,E) yields a Spiking Neural Network (SNN). Network level properties are discussed in Section 2.3. It is followed by an introduction to the electrocardiogram (ECG) in Section 2.4.

2.1 The Spiking Neuron

The neuron is the primary building block of the central (CNS) and peripheral (PNS) nervous system responsible for information processing using chemical and electrical signals. Although they are also found in other forms in the peripheral nervous system we will focus on the neurons found in the central nervous system only.

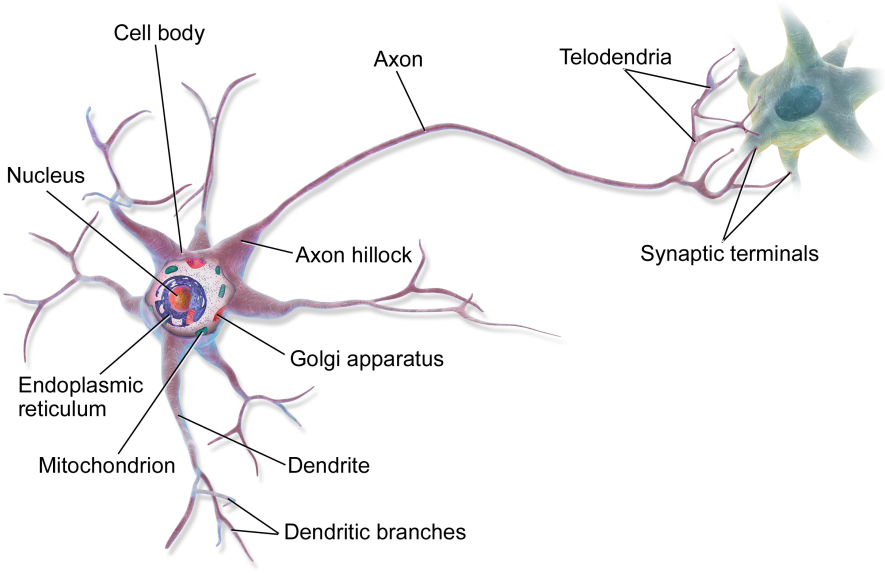

Figure 2.1: Anatomy of a biological multipolar neuron [2]

A neuron is comprised of three main functional blocks shown in Figure 2.1: the dendrites, the soma (also called cell body), and the axon. The dendrites are the inputs: branch like structures that accept and transport electrochemical input from other neurons to the processing area of this data called the soma. The result of this processing is generated in the axon hillock and sent via the axon to the output the neuron: the synaptic terminals.

Even in the just the CNS neurons exist in many different shapes and sizes but we will focus on a specific subset of these called the multipolar neuron. This means that the neuron we are examining can accept input from many different neurons using a branched out dendritic structure but only has one output channel and generator. This output can still be propagated to many different neurons. The multipolar neuron is the most common type of neuron in the CNS.

The soma *membrane potential* can be considered as an approximation of the state of a neuron. Like a capacitor, this potential exists across the bilipid outer layer of all biological cells. Also like a capacitor this state variable can be modified by changing the concentration of charge carrying ions including K^+ and Na^+ ions on both sides of the membrane. In the resting state the measured voltage across this layer of a biological cell is actively maintained between -70 mV and -50 mV [15] where the interior is more negative than the exterior. Ion transportation and concentration gradient maintenance between the inside and outside is facilitated by ion channels and pumps that can be

membrane potential dependent themselves creating a feedback loop.

The *action potential* or simply *spike* is the primary way of communication between neurons. This is a sudden change in the membrane potential of the neuron. This spike is generated at the axon hillock and the result of a rapid change in ion concentrations that define the membrane potential. These spikes are sent through the axon via synaptic connections described in Section 2.2 to dendrites of the next neuron. The next neuron receives this information carrier in its dendritic branches and sends it to its soma where the spike is accumulated and can result in the generation of another spike.

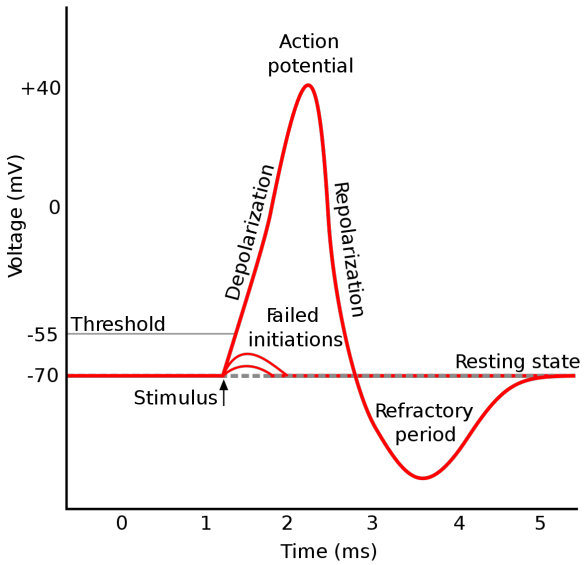

Figure 2.2: Various membrane potential phenomena shown over time [3]

In the labeled action potential in Figure 2.2 we can observe the resting membrane potential of a neuron at -70 mV being changed by two possible stimulations: first the arrival of input action potentials of *insufficient* magnitude only generating a slight raise in membrane potential that, again, similar to a capacitor leaks away to resting state over time (failed initiations). Second, if *sufficient* spikes arrive within a short time frame of each other before the additional ion concentration gradient can leak away the neuron generates its own spike that it transports to all of its neighbours. This example assumes that the subject neuron is of the potentiaton type which is not always the case. For more information on this subject refer to Section 2.2.

2.1.1 Neuron Models

The electrophysiological phenomena described above are extremely complex to model accurately but we are not looking for such a detailed model of single neurons in the brain. What we are looking for is a more simple model that only needs to be accurate enough so that it can be used for spike-based learning but not more. Overly accurate models are an unneccesary burden in total system complexity.

The following three neural models have been investigated: the complex but accurate

Hodgkin-Huxley model [4], the simpler Izhikevich model [16] and the Integrate & Fire model [17]. The Integrate & Fire model has been chosen because it is a commonly used as-simple-as-possible model that can generate unique spike events. The Izhikevich model is investigated because it is presented as computationally similar to Integrate & Fire but with more detailed functionality. For a more thorough list of neural models available, see [18].

2.1.1.1 Hodgkin-Huxley Model

The Hodgkin-Huxley is a more complex electrical modeling of current flow through a membrane of a biological neuron. This model provides a more reference of modeling of the biological neuron and is meant to show what simpler models discussed further down are approximating.

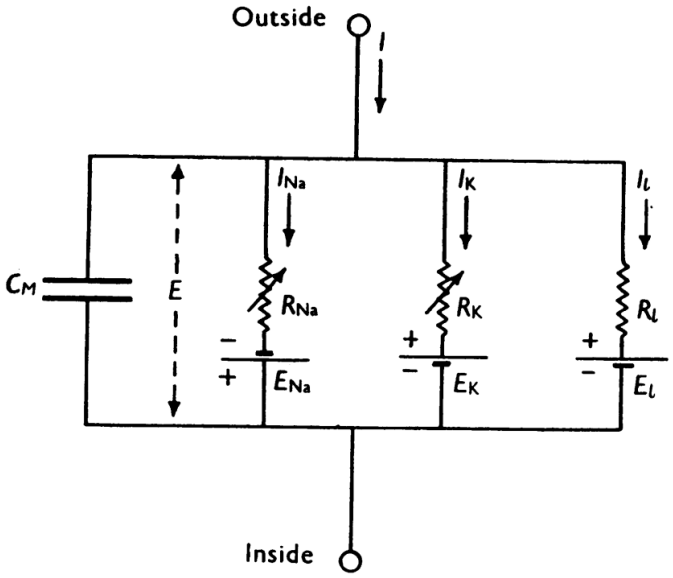

Figure 2.3: Equivalent circuit of the Hodgkin-Huxley membrane model [4]

As can be seen in Figure 2.3 the Hodgkin-Huxley (HH) model describes the membrane as a capacitance that stores the membrane potential in parallel with various nonlinear ion channels for sodium and potassium (Na and K) and a leakage current (L) that are able to change the membrane potential. The membrane potential E is defined as the voltage between the top and bottom node.

The full HH model consists of four differential equations and is strictly nonlinear. Unlike the Integrate & Fire model described in Section 2.1.1.2 it is able to produce a vast array of behaviours generally coming from the equations for the ion channels being dependent on the membrane voltage in intricate ways.

2.1.1.2 Integrate & Fire Model

The Integrate & Fire model is a more formal model that describes a neuron as a much simpler electronic circuit. This electronic circuit is shown in Figure 2.4. It models the

neuron membrane as a simple constant value parallel capacitor C_{mem} and resistor R_{leak} (an RC circuit). The membrane potential is defined as the voltage across the capacitor plates and therefore also across the resistor. If leaking is included the model is called the Leaky Integrate & Fire Model, that includes leakage of capacitor charge through the parallel resistance. If the voltage across the capacitor plates is high enough, a spike is generated and the capacitor is reset.

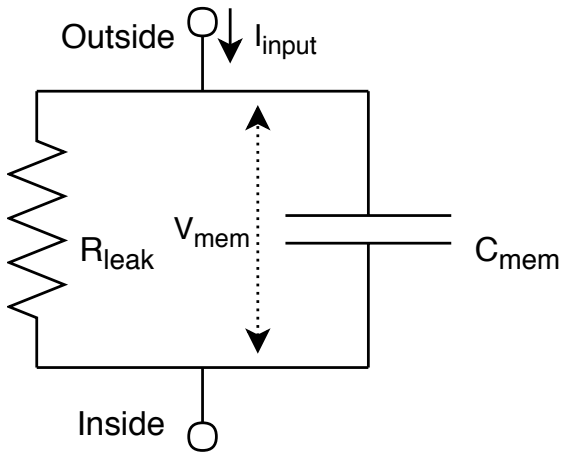

Figure 2.4: Equivalent circuit of the Leaky Integrate & Fire Model membrane model. Complexity can be compared with Figure 2.3. For both circuits threshold detection logic is omitted.

We can observe its electrical behaviour if we write down the accumulating capacitor equation:

$$C_{mem}\frac{dV_{mem}}{dt} = I \tag{2.1}$$

$$C_{mem}\frac{dV_{mem}}{dt} = I_{input} - \frac{V_{mem}}{R_{leak}} \tag{2.2}$$

$$V_{mem} = V_{mem} + \frac{dt}{C_{mem}}\left(I_{input} - \frac{V_{mem}}{R_{leak}}\right) \tag{2.3}$$

$$\text{if } V_{mem} > V_{thresh}: \tag{2.4}$$

$$V_{mem} = V_{reset} \tag{2.5}$$

As can be seen in Equation 2.3 the membrane accumulates external charge and simulatenously leaks away through a parallel resistance.

This model only consists of one differential equation (Equation 2.3) and one reset conditional (Equation 2.4) that detects and handles spiking events and is therefore very computationally efficient but does not take into account any physical or chemical properties of the neuron.

When Integrate & Fire is mentioned it is from now on assumed that the leak component is present.

2.1.1.3 Izhikevich Model

The model by Izhikevich is presented as a model that combines the biological plausibility of the HH model and the computational efficieny similar to the Integrate & Fire model. It consists of two simple differential equations compared to four more computationally expensive ones in HH. This is complemented with one reset conditional that detects and handles spiking events similar to the Integrate & Fire model.

In mathematical form the model is as follows:

$$\frac{dv}{dt} = 0.04v^2 + 5v + 140 - u + I \tag{2.6}$$

$$\frac{du}{dt} = a(bv - u) \tag{2.7}$$

$$\text{if } v > v_{thresh} : \tag{2.8}$$

$$v = c \tag{2.9}$$

$$u = u + d \tag{2.10}$$

This model is a mathematical fitting of biological neural membrane potential behaviour over time aimed at reducing simulation costs. The model equations and variables should be viewed as abstract mathematical objects that are not based on physical units or quantities. The variable descriptions can include physical units even though they would not make sense in the equation context.

The variable v is to be interpreted as the membrane voltage and accumulates a current I. The variable u describes the activity of membrane ion channels related to recovery from spike events. When v reaches a certain threshold it is reset to a configurable zero point c and u increases by d that is to be interpreted as a measure of how much a neuron has to recover from a spike event. The constant a describes the time scale of recovery from spikes. The constant b can be used to tune the neuron to exhibit different kinds of oscillatory behaviour.

2.1.1.4 Computational Comparison

A overview that summarizes the models from a computational viewpoint is shown below in Table 2.1.

Table 2.1: Computational complexity comparison of selected neural models

Property	Hodgkin-Huxley	Integrate & Fire	Izhikevich
Diff. equations	4	1	2
Aux. equations	6	1	1
State variables	4	1	2
Config. variables	7 + 6 eq.	4	5
Hardware cost	high	low	medium

This table shows that if one requires high model accuracy the HH model is preferred. However this comes at a larger hardware cost, increased computational complexity and configuration difficulty. For quantitative hardware cost figures please refer to various survey publications by Indiveri et al. [19][20] Other hardware implementations of comparable but not equivalent Izhikevich and Integrate & Fire models can be found in [21] and [22] respectively.

2.2 The Synapse

The synapse is the primary way in which a biological neural network transports information in the form of action potentials from spiking neuron A to spiking neuron B.

The biological synapse that we are modeling consists of three basic parts: the *presynaptic* terminal or axon terminal, the synaptic cleft and the *postsynaptic* terminal or dendritic receptor. In Figure 2.5 the presynaptic terminal is the upper part of the zoomed in picture, and the lower part is the postsynaptic terminal. Terminology for the gap in between varies, and includes the terms *synaptic gap*, *synaptic cleft* (this document), and simply synapse.

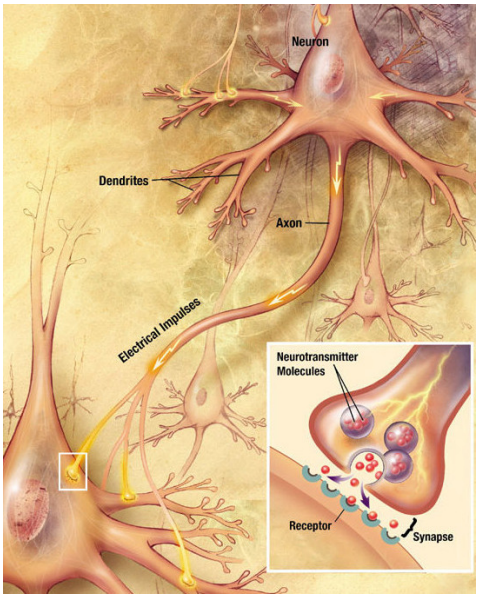

Figure 2.5: A chemical depiction of a neural synapse [5]

The critical component of the synapse is the synaptic cleft that physically separates both terminals. When a presynaptic neuron generates an action potential and it arrives at the axon terminal it causes a release of neurotransmitters from the presynaptic terminal into the cleft that subsequently drift towards the dendritic terminal and activate receptors that are a part of the receiving postsynaptic neuron. The subsequent activity in the postsynaptic neuron is called the *postsynaptic potential* (PSP).

Two types of PSPs exist: the *excitatory postsynaptic potential* (EPSP) and the

inhibitory postsynaptic potential (IPSP). Depending on the physiology of the neuron it is able to release either excitatory or inhibitory neurotransmitters into the cleft. In response to those transmitters the postsynaptic membrane potential either depolarizes or hyperpolarizes (see Figure 2.2). Depolarization can be seen as activating and is required for action potentials to be generated while hyperpolarization can be seen as suppressing.

Similar to the biological neuron the biological synapse is extremely complex to model accurately. Again, we are not interested in the detailed electrophysical behaviour of a single synapse, but the behaviour of a population of synapses in a network coupled with neurons to form a network. To increase network scalability we would like to use a model that is as accurate as possible while being simple to compute.

2.2.1 Synaptic Properties

The synapse has various properties that are of interest in our exploration, namely: its *conductivity* or *weight*, propagation *delay*, and changeability of conductivity and delay or *synaptic plasticity*.

The conductivity is a measure of the response of a synapse to an incoming action potential. When comparing synapses to wires they are modeled as the $\frac{1}{R_{synapse}}$ of a *RC* wire delay model that spikes have to pass through. This model is depicted in Figure 2.6. It is determined by the neurotransmitter emission of the presynaptic terminal, the physical dimensions and drift characteristics of these neurotransmitters across the cleft and response properties to these transmitters in the postsynaptic terminal. The variation of conductivity due to learning is further discussed in Section 2.2.3.

The delay is a measure of the time it takes for a spike to traverse the synapse. Looking at the wire analogy again, think of the delay as a factor primarily determined by the to-ground capacitance $C_{synapse}$ of a *RC* wire model. In biology this property is a combination of the delay of the axonal part of a neuron (which is officialy outside of what we call a synapse in this document) and the delay of the synaptic cleft. For spike transport, the axon needs to be depolarized along its entire length which takes time depending on things like membrane capacitance and leakage factors, both modified by myelin sheath quality, axon length and more. For the synaptic cleft delay occurs due to earlier explained consecutive chemical processes of receptor binding, release of response, travel distances, and further such iterations.

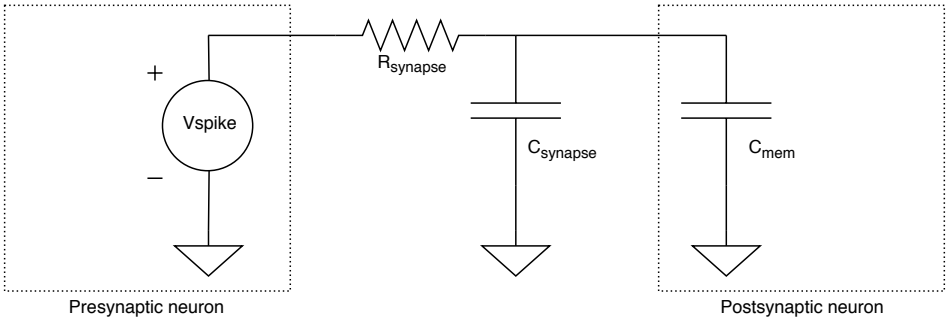

Figure 2.6: Synapse electrically modeled as an RC circuit

2.2.2 Synaptic Delay Plasticity

Although the existence and potential of synaptic *delay* plasticity in artificial learning is known to the author, due to time constraints of this project it has not been investigated further. All plasticity from here on is targeted at modifying *conductivity*.

2.2.3 Synaptic Plasticity

Synaptic plasticity is the ability of synapses to change their response strength to stimulus in the form of action potentials or *spikes* generated by the neurons they are connected to. Conventionally this research topic only covers $\frac{1}{R_{synapse}}$ conductivity changes and resulting modification of action potential amplitude. Much less research is available on the use in artificial learning of changes in synaptic delay caused by C and resulting changes in spike timing.

In neurophysiology this conductivity plasticity is described by two opposing concepts: Long Term Potentiation (LTP) and Long Term Depression (LTD). In LTP, there is a lasting (in the time frame of hours or longer) increase in conductivity of the synapse while in LTD there is a lasting decrease in the conductivity of the synapse.

In 1949, Donald Hebb presented a theory that describes how biological systems implement plasticity known as Hebb's rule [23] that can be summarized as follows: *'a metabolic change in a synapse takes place if repeated firing of a presynaptic neuron induces firing of a postsynaptic neuron causing the efficiency of the synapse to increase'*. An even shorter summary would be *'cells that causally fire together wire together'*.

Various mathematical processes have been proposed and hardware-implemented to describe this biological process in a simple mathematical way [11]. These processes can be divided into a group of discrete spike based algorithms and more abstract spiking activity based algorithms. This thesis will limit itself to purely spike timing based algorithms that are also implemented in VLSI. Other spike timing or spike activity based algorithms like BCM [24] and SDSP [25][26] exist, even in VLSI, but are only shown as a point of comparison in Table 2.2.

2.2.3.1 Spike Timing Dependent Plasticity

The Spike Timing Dependent Plasticity (STDP) algorithm [6] attempts to describe synaptic plasticity observed in dissociated rat hippocampal neurons using two simple spike timing dependent rules:

$$\Delta w^{+} = f_{+}(w) \cdot A_{+} \cdot exp(\frac{-\Delta t}{\tau_{+}}) \qquad \Delta t > 0 \tag{2.11}$$

$$\Delta w^{-} = -f_{-}(w) \cdot A_{-} \cdot exp(\frac{-\Delta t}{\tau_{-}}) \qquad \Delta t < 0 \tag{2.12}$$

$$\Delta t = t_{last,post} - t_{last,pre} \tag{2.13}$$

These equations are the result of a fitting of the change in synaptic conductivity Δw observed as a function of the Δt between two neurons connected by a synapse. This Δt is the time difference between the last occurrence of a spike in the postsynaptic neuron and presynaptic neuron and is positive when the postsynaptic neuron spikes later. When combined the first two equations form the *STDP window*. The original experimental results that led to these equations are shown in Figure 2.7 below.

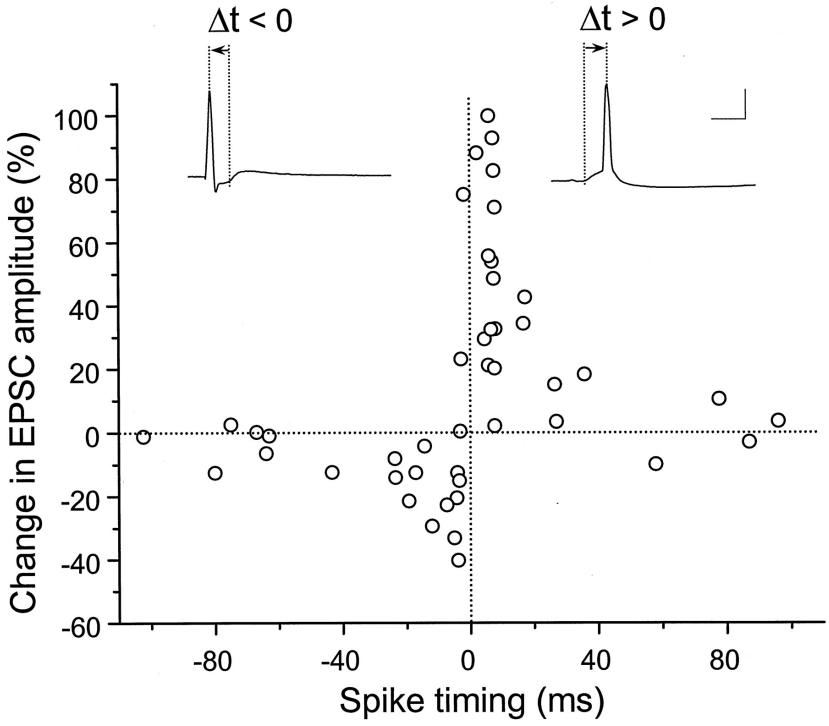

Figure 2.7: STDP time window measurements of possible potentiation or depression [6]

By closer inspection it is possible to observe the two exponentially decaying fitting functions in this data. The area of effect where synaptic conductivity changes significantly depending on the spike time difference Δt stretches to roughly 100 ms both ways around each spike. No significant synaptic conductivity change happens for Δt higher or lower than that.

These two equations accurately describe synaptic conductivity change only in an isolated environment of one biological synapse connecting two neurons where for each spike event in any of the two neurons all neurons including the spiker itself only spike up to one time in the significant portion of the *STDP window* around that event.

This algorithm contains four parameters to configure: the positive learning magnitude A_+ (100% normalized in Figure 2.7), negative learning magnitude A_- (about -50%), positive time decay τ_+ and negative time decay τ_-. The current conductivity dependency function $f_+(w)$ is optional and can be left at 1.

2.2.3.2 Triplet STDP

The Triplet Spike Timing Dependent Plasticity (TSTDP) algorithm [7] is an extension to STDP that takes the two most recent (instead of one) spike events per neuron into account to compute plasticity. It was proposed to fit experiment data where up to two spike events per neuron connected by a biological synapse were present in the significant portion of the STDP window. Recall from Section 2.2.3.1 that this is a region where STDP fails to reproduce experiment data.

The full triplet rule algorithm consists of four exponential equations that describe the conductivity change Δw as a result of the spike time difference Δt_x between four key combinations of spikes at the presynaptic and postsynaptic end of the synapse. These combinations are shown in Figure 2.8.

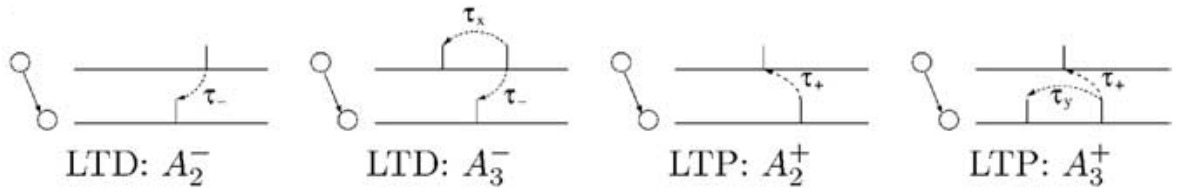

Figure 2.8: TSTDP pairs taken into account [7]

As seen in the figure above, the first and third pair are simply the conventional STDP pairs. The second and fourth set extend from this and take into account one more spike event in the spike event history of the presynaptic and postsynaptic neuron respectively. The four exponential equations are shown below:

$$\Delta w^+ = f_+(w)\cdot(A_{2+}\cdot exp(\frac{-\Delta t_1}{\tau_+}) + (A_{3+}\cdot exp(\frac{-\Delta t_2}{\tau_y})) \qquad \Delta t_1 > 0 \tag{2.14}$$

$$\Delta w^- = -f_-(w)\cdot(A_{2-}\cdot exp(\frac{-\Delta t_1}{\tau_-}) + (A_{3-}\cdot exp(\frac{-\Delta t_3}{\tau_x})) \qquad \Delta t_1 < 0 \tag{2.15}$$

As opposed to STDP this algorithm contains eight parameters to configure: the positive learning magnitudes A_{2+} and A_{3+}, negative learning magnitudes A_{2-} and A_{3-}, positive time decays τ_+ and τ_y, and negative time decays τ_- and τ_x. As with STDP the current conductivity dependency function $f_+(w)$ is optional and can be left at 1. A keen eye will observe that if both A_{3+} and A_{3-} are set to 0 we arrive at the STDP equation set.

To summarize, TSTDP is a more advanced and biologically correct variant of STDP that does not break down at higher spike event rates like STDP does when multiple spikes appear in its *window*. However, it comes at the cost of significantly higher configuration and hardware complexity.

2.2.3.3 Computational Comparison

A overview that summarizes the plasticitiy models by various properties is shown below in Table 2.2.

Table 2.2: Selection of synaptic plasticity models implemented in VLSI [11]

Model	Neural properties used for synaptic plasticity	Config variables
Pair STDP	last spike time	4
Triplet STDP	last-1, last spike time	8
SDSP	spike event, membrane voltage, ion concentration	>8 [25]
BCM	abstracted spike activity into currents	see [24]

2.3 Network Level Properties

The SNN combines the spiking neuron and synapse to form a network through which information in the form of spikes can flow. The layout of this network decides the flow of and response to stimulus provided to the network. Adaptation to stimulus is defined by the network layout, network learning methods, and synaptic level learning methods. This is explained in Section 2.3.2.

2.3.1 Network Layout

Classical neural networks are divided into three *layers* composed of five computation steps as seen in Figure 2.9. First we have an input layer built up from neurons (circles) each performing a certain fixed computation on their (external) input. Their response is propagated into the rest of the network as x. Then there is the hidden layer with connectivity (rectangle) in between that performs computation f and generates response $y_1 = f(x)$, and the output layer with connectivity in between that performs a computation g but whose response $y_2 = g(f(x))$ is presented to the outside world.

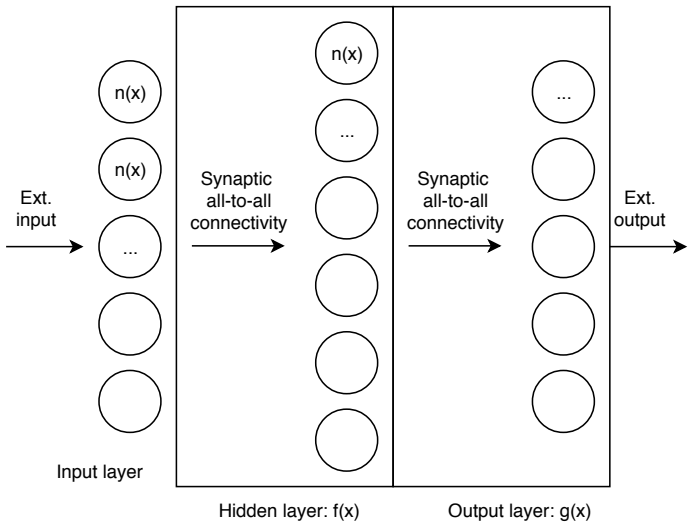

Figure 2.9: Generalized information flow in a multilayer NN

Although the hidden layer might seem unnecessary because it seems like we can perform what f does inside g thereby saving on network complexity this is not the case. Each computation layer is limited to performing a function on its input that is composed of a sum of functions that comply with the UAT (nonlinear, bounded, monotonous). For some classification tasks like the classic XOR problem it is not possible to define a single layer g with UAT compliant functions that solves it. Such a limited network without a hidden layer f is called the perceptron [27], and was fundamentally limited as was later proven. The above does not mean that some generic function $f_{XOR}(x_1, x_2) = XOR(x_1, x_2)$ does not exist but it means that we cannot learn it using networks with one computation layer.

For SNNs, the above concepts are similar but the computational functions that describe neurons, layers, and combinations of layers are less abstract and are defined in terms of *spikes* instead of arbitrary values in the real domain. In this project, a three layer network layout called the Self-Organizing Map is used, that also contains two synaptic all-to-all connectivity layers with one mayor difference being that the second synaptic layer feeds back to the same neurons. It is further explained in Section 3.2.8.

2.3.2 Learning Methods

There is a wide variety of network level learning methods that can be used for learning in SNNs. While the synaptic level learning rules like STDP, TSTDP and others define adaptation in their microscopic scale, learning methods like supervised learning, unsupervised learning, and others define the network level learning approach. These network level methods define the environment in which the whole SNN is being taught, and define the type and amount of help it receives from outside sources.

In unsupervised learning, the task of learning and adaptation in the synaptic level purely depends on the characteristics of data provided, and no signal is present that teaches the network what to do. The data provided to the network also does not contain explicit information called *labels* that inform the network how to process it. Hebbian learning and related synaptic plasticity are considered to be a part of unsupervised learning where the learning that these concepts define happens based on implicit characteristics of the data provided, like spike timings. Another characteristic of unsupervised learning is that the resulting input to output function from learning cannot be compared to a reference results as there is no such defined function present.

In its counterpart supervised learning, there is an external teacher signal available that *teaches* the network how to adapt to input provided. It does this by labeling input data during learning and informing the network how this data should be processed. By computing the difference between the network functionality that the teacher desires and the actual network functionality adaptation strategies can be followed by back-propagation to achieve the desired functionality.

In this project pure unsupervised learning is used, with learning only occurring using spike timing characteristics of data provided to the network.

2.4 The Electrocardiogram

The electrocardiogram (ECG) is an electrical recording of heart activity used in the medical field of cardiology and can be used as a tool to determine cardiovascular health. The output of an ECG is a voltage measurement over time that exists between any combination of two points distributed across the body. This voltage is a component of the electrical field produced by current flow in the heart muscle and the control system of the heart. Since these systems work using the concept of ion transport from A to B one can associate an electrical dipole between high and low points of ion concentrations. This in turn produces the mentioned electric field that is finally measured between any two points.

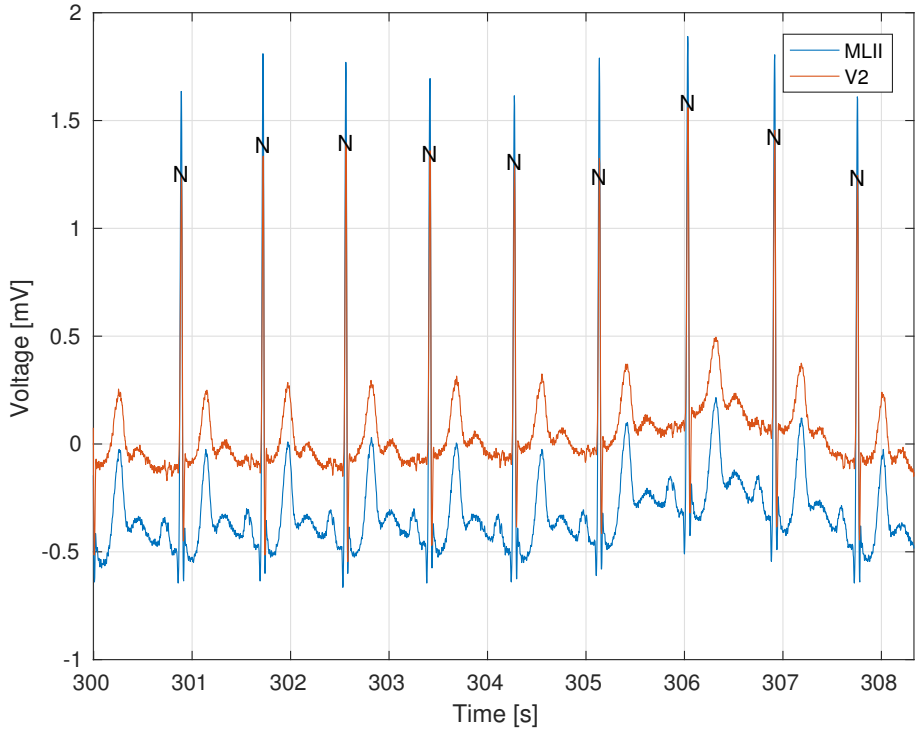

Figure 2.10: Snippet from a ECG recording from the MIT-BIH database [8] showing measurements between two pairs of points named in Table 2.4. Annotations of each beat type are shown at each peak where N stands for a normal beat.

2.4.1 Measurement Setup

A standard ECG setup has ten standardized points or *electrodes* where a conducting pad is positioned on the skin [28]. The term *lead* often found in literature refers to the wire connecting to the electrode. Since electrodes are used for multiple wire connections in the standard ECG setup, the setup can be called the 12-lead setup or 10-electrode setup telling the reader that there are two leads that do not have their own individual electrode.

Table 2.3: Standard 12-lead ECG electrode positions

Electrode name	Electrode placement
RA	On the right arm, avoiding thick muscle.
LA	In the same location where RA was placed, but on the left arm.
RL	On the right leg, lower end of medial aspect of calf muscle.
LL	In the same location where RL was placed, but on the left leg.
V1	In the fourth intercostal space just to the right of the sternum.
V2	In the fourth intercostal space just to the left of the sternum.
V3	Between leads V2 and V4.
V4	In the fifth intercostal space in the mid-clavicular line.
V5	Horizontally even with V4, in the left anterior axillary line.
V6	Horizontally even with V4 and V5 in the midaxillary line.

Based on these ten measurement points the conventional setup measures twelve potential differences to generate a three dimensional map of current flow in the heart. Each lead can be seen as a viewpoint of the three dimensional heart dipole and its surrounding electrical field. Conventionally, the right leg (RL) is only used as a ground and for common-mode noise reduction while the left leg (LL) is used for actual measurement.

The following table shows the setup for the 12-lead ECG configuration:

Table 2.4: Standard 12-lead ECG lead configuration

Lead name	Description
(ML)I	LA - RA
(ML)II	LL - RA
(ML)III	LL - LA
aVR	RA - $\frac{1}{2}$(LA + LL)
aVL	LA - $\frac{1}{2}$(RA + LL)
aVF	LL - $\frac{1}{2}$(RA + LA)
V1..V6	V1..V6 - $\frac{1}{3}$(RA + LA + LL)

The oldest of these leads are I, II and III and combined they form the most basic ECG setup called the 3-lead setup. They are the measurements across the edges of Einthoven's triangle [29]. A graphical description of this measurement triangle is shown in Figure 2.11. The common ECG measurement shape people are most familiar with is shown in the introduction as Figure 1.2 and is usually the result of lead II. Note that with Kirchoff's law it can be proven that I + III = II (or any other combination) so any two leads are sufficient to do a 3-lead ECG measurement. The 3-lead setup is not able to show all important current flow directions in the heart. However it is the setup for which data is most commonly available.

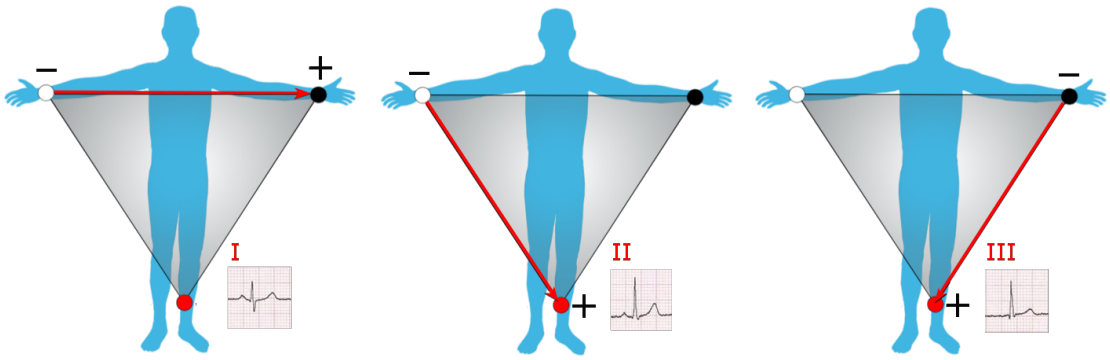

Figure 2.11: Diagrams of the first 3 ECG leads [9]

2.4.2 Features

The process of reading an ECG wave can be seen as an exercise in pattern recognition. Various specific features of the ECG readings are associated to certain cardiovascular diseases. For ease of reading the ECG feature picture is repeated Figure 2.12:

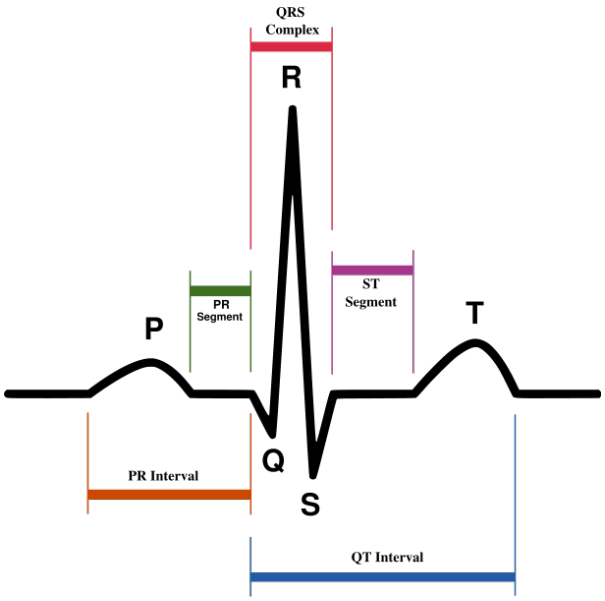

Figure 2.12: ECG features used for categorization [1] of beats

Heart beats and frequency are initiated by the sinoatrial node. This clump of cells generates an action potential triggering the start of the P-wave of each full ECG beat. The complete P-wave is the result of atrial depolarization (generation of a positive membrane potential), expansion and subsequent contraction. The QRS-complex is the result of ventricular depolarization, expansion and subsequent contraction. The peak of the QRS-complex is of extra importance in this document and is called the R-peak.

The T-wave is a recovery phase of the heart, where control and muscle cells reset to their resting potential.

Careful analysis of the shape, occurrence, and temporal location of these features of the ECG wave in various measurement leads helps the cardiologist to diagnose cardiovascular problems. There are multiple orders of beat features that can be analysed for this purpose: *first order* features that are measurements inside a single beat, *second order* measurements that have a two beat scope like the R-to-previous-R (RR) interval, and *third order* measurements such as the RR-minus-previous-RR interval.

An example list of feature anomalies and possible causes is shown in Table 2.5.

Table 2.5: Example selection of ECG features linked to diseases [12]

Feature Description	Possible Causes
Double peak P wave	L/R Atrial enlargement (LAE, RAE)
Wide QRS complex	Ventricular tachycardia (VT)
Abnormal QRS and T, no P wave	Premature ventricular contraction (PVC)
ST segment elevation	Ischemia

2.4.3 Motivation For Automation

The ECG analysis process can be seen as a tedious and error-prone task for a cardiologist. First he or she has to manually find and annotate each beat in up to twelve parallel signals per patient. After finding all beats which by itself is extremely difficult to do perfectly in 'complicated' cases where beats do not even remotely look like the textbook version (see Figure 2.12) they have to be manually annotated with a beat type label. Even though work on the MIT-BIH database [8] started more than 38 years ago, fixes to the manual annotations are being uploaded to this day proving that human analysis for this task is far from perfect.

To improve the throughput to at least real time, maintain or increase the quality of work, and reduce power consumption to wearable levels we propose an architecture presented in the next chapter as a fully automated solution from raw lead data to beat categorization. This system includes raw lead data filtering with DSP, beat detection with DSP, and beat categorization using a neuromorphic hardware-implemented SNN.

Compared to current options for automated ECG analysis like a full DSP solution or a DSP+ANN mix the DSP+SNN approach offers much reduced power consumption, but at the cost of increased system design space complexity.

3 Architecture

In this chapter the architecture of the learning system incorporating an SNN from raw ECG signal input to categorization output is discussed. Let us call this system *the device* from now on.

First in Section 3.1 the state of the art of devices implementing SNNs is provided. Although the application and way of implementation are different, they still provide a reference of recent and not so recent ways of incorporating and making use of SNNs. Nexxt in Section 3.2 the device will be discussed in a top-down block by block fashion. It presents the architectural details of how the biological concepts from Chapter 2 have been used. It contains the implementation details of all device blocks between extracting beats from raw ECG signals and categorization output. Of utmost importance is the core concept of the device called the Self-Organizing Map, and its underlying workings. They are described in Section 3.2.8.

3.1 State Of The Art

The state of the art for this chapter covers other projects that have incorporated neuromorphic SNNs into their devices. In this field of implementations of large scale models, the state of the art includes the following significant works:

The SpiNNaker project [30] aims to deliver a massively parallel million-core computer with the goal of simulating up to a billion neurons in real time [20]. It uses expandable arrays of custom VLSI chips that contain conventional Harvard architecture ARM cores that are able to simulate various neural and synaptic models. Interconnect between these chips uses the AER protocol. This device is able to simulate arbitrary connectivity, plasticity and neural behaviour. However because it is still based on conventional load/store computer architectures it still suffers from traditional computing problems like memory performance limitations, interconnect limitations and high energy per computation. In summary it mimics biological behaviour on the functional level but not on the device level.

The TrueNorth project by IBM [31] attempts to depart from conventional computer architectures and uses a custom made digital CMOS device to implement an array of leaky Integrate & Fire neurons linked together by three-level weighted synapses without delay. These three levels are distributed from maximal inhibition to maximum excitation. The connectivity graph is not arbitrary: only 256 neurons can be targeted by any individual neuron. No plasticity algorithms are implemented on the hardware level. In summary this is an array of biological neurons implemented in custom made hardware allowing for a significant reduction in power consumption but there is a lack of flexibility on the hardware level since synaptic weights and delays are limited and learning has to be done off-chip.

Other projects like NeuroGrid [32], ROLLS [33] and Intel Loihi [34] exist that also explore neuromorphic concepts including SNNs to accelerate computing. For more information please refer to surveys like [20].

Various smaller but more comparable works to this device have created an SOM network topology based on SNNs. These include Ruf [35] and Rumbell [36]. Authors like Bohte [10] have touched upon the concept of mapping using SNNs using Population Offset Encoding.

3.2 System Architecture

The device consists of five global building blocks: the Feature Detector discussed in Section 3.2.3, the Feature Selector discussed in Section 3.2.4, the Input Encoder discussed in Section 3.2.5, the core SNN simulator discussed in Section 3.2.6 and finally the Output Decoder discussed in Section 3.2.7.

When combined, these five blocks are able to categorize ECG signals using a two state process: they can be *trained* by feeding it raw ECG waveforms and can subsequently be *tested* by feeding it individual ECG waves that it is able to categorize. This state system is explained in Section 3.2.1.

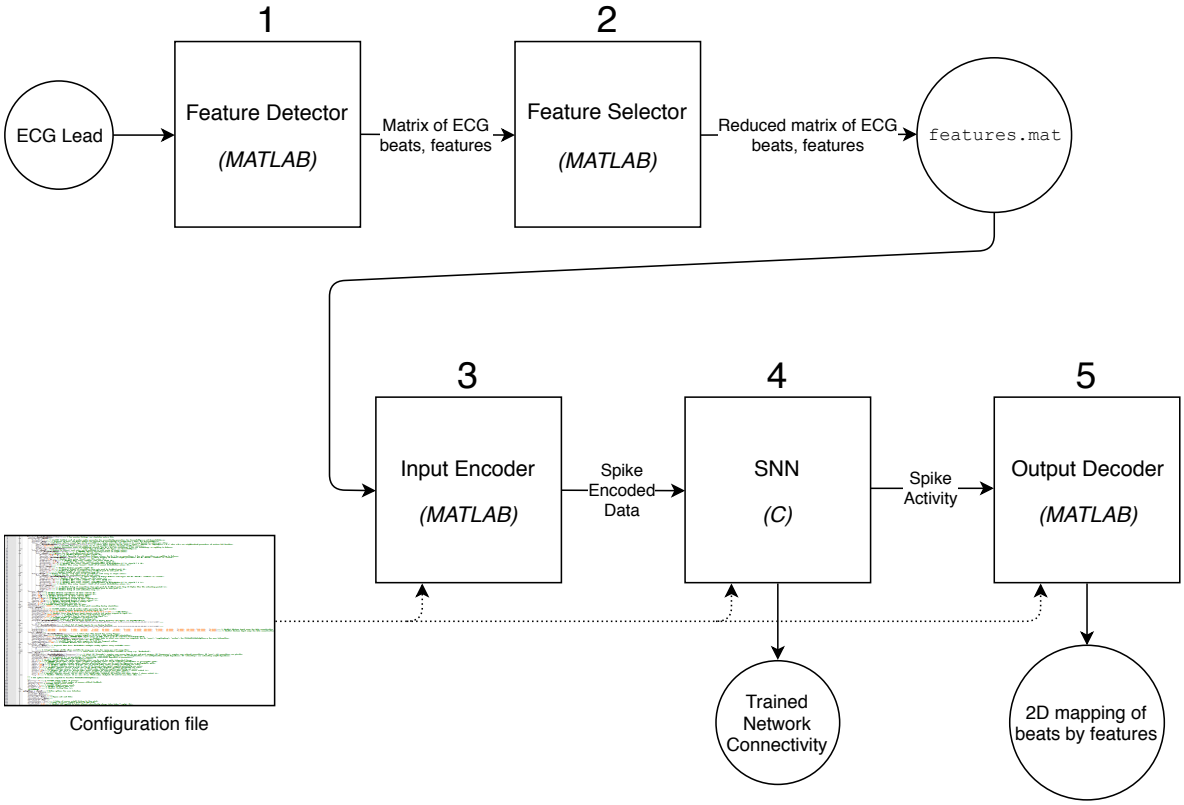

Figure 3.1: Top-level system overview

3.2.1 Training & Testing

From a functionality standpoint the device can be in two states as seen in Figure 3.2: training (1) and testing (2). Before any of these states are entered, the device exists in a blank state with connectivity being defined by the user's configuration (0). When the training state is entered the device is designed to accept raw ECG lead data and feed it to the network. The network will be trained: it adapts its connectivity to the input data provided to extract meaningful information from it. If training results are deemed adequate the testing phase will start. In this phase connectivity is fixed, a specific testing (sometimes called validation in literature) set of ECG beats is provided and the device will categorize these beats, a process that is presented to the user in image or raw data form.

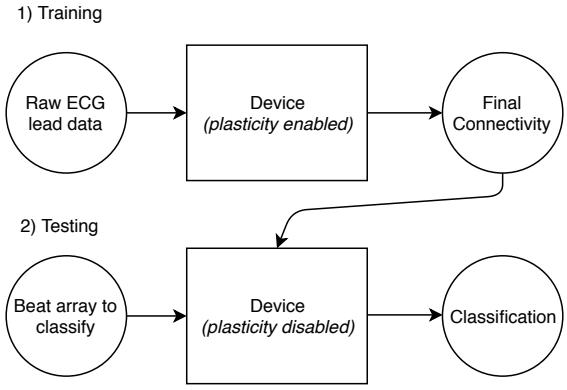

Figure 3.2: Overview of the training and testing states for the ECG use case

3.2.2 Input Dataset

Before covering the five top level blocks of the device, it is important to understand what is being fed into it at the Feature Detector stage. This data is called the input dataset. In the for this thesis relevant literature the most common benchmark dataset for ECG measurements is the MIT-BIH dataset [8] [37]. It is a part of the PhysioBank recording archive and contains half hour lead readings for 48 patients with varying backgrounds, symptoms and underlying conditions. It is employed for training and testing of the device. For each patient three data files are provided: the `.dat` data file that contains lead readings, the `.hea` header file that contains patient and reading properties and the `.atr` file that contains annotations for the data in the `.dat` file.

3.2.2.1 Lead Data

For 46 out of the complete set of 48 patients in MIT-BIH two leads are available inside the `.dat` file: lead II from the classic 3-lead ECG setup and one auxilliary lead, most often a chest lead like V1 to V6. Since the common denominator of these readings is lead II and since that lead is also used in relevant Feature Extraction literature the other remaining lead from these patient file is ignored. The full half hour recordings of lead II of these 46 patients are used as the data set for the device.

3.2.2.2 Lead Data Annotations

The MIT-BIH dataset not only contains voltage readings over time but also manually inserted annotations that describe the type of beat detected at the annotation position as seen in Figure 3.3. Each waveform that has been deemed a beat by cardiologists contains exactly one annotation roughly at the R-peak. These annotations use the PhysioNet format [38].

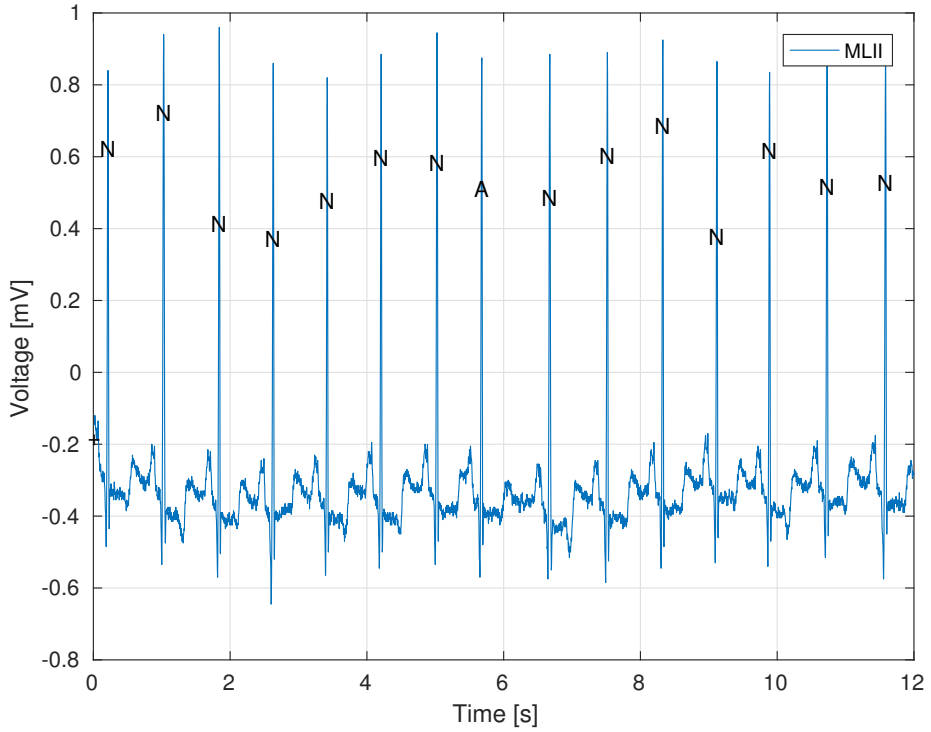

Figure 3.3: First 0.2 minutes of patient 100 in MIT-BIH with PhysioNet format annotations

As seen in the figure above all except the middle beat have been labeled by cardiologists as an *N* standing for *Normal beat* following the PhysioNet standard. The abnormally early beat is an *A* or *Atrial premature beat* (APB) also known as a *premature atrial contraction* [12]. A non-beat annotation + is shown at the very start and denotes a rhythm change. For more information about the possible annotations see the PhysioNet format website.

3.2.3 Feature Detector

The first block in the system is the Feature Detector. The core task of the Feature Detector is to extract the most valuable properties or features from a raw ECG input signal. In the training state of the device, data of a list of patients is presented to the input. For each patient only the MLII lead data is used although multiple leads are supported. This block was designed and implemented in cooperation with Eralp Kolagasioglu [39]. The step by step process of the Feature Detector is as follows and is depicted in Figure 3.4:

1. Data acquisition from a database file from MIT-BIH.
2. Performing a feature detection algorithm on the voltage readings:
   (a) Filtering of the data signal to reduce noise. Noise sources include 50/60 Hz power supply noise, patient movement noise and baseline drift.
   (b) (optional) Transformation to a domain that expresses required features the most.
   (c) Detection of simple features like peaks and zero crossings in the transformed domain that are associated with desired features in the time domain.
3. Construction of the full output data file from the features detected by step 2c.

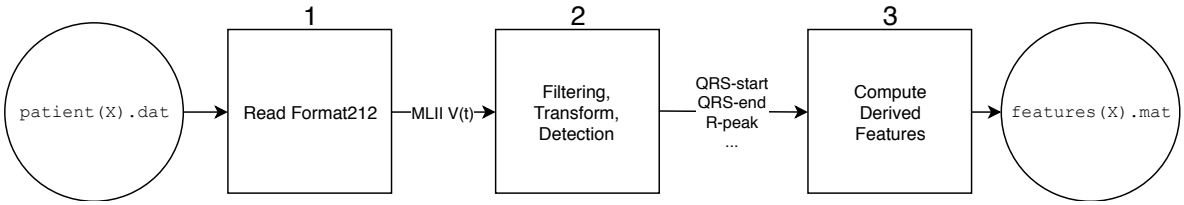

Figure 3.4: Feature Detector block diagram

3.2.3.1 Data Acquisition

The first step is data acquisition. Like most databases in the PhysioBank the MIT-BIH database uses the format 212 encoding [40], graphically explained in Figure 3.5. For each patient it stores 650000 samples of two leads at a measurement frequency of 360 Hz for a total sampled time of 30.09 min. Each sample is a 12 bit two's complement integer that needs to be biased with a zero value and multiplied by a gain defined for each patient reading.

Although a reference implementation called `rdsamp()` for reading the lead voltage files is available [41] it was deemed too slow for use so a custom fast implementation was made. For reading lead annotation files the reference implementation `rdann()` was used.

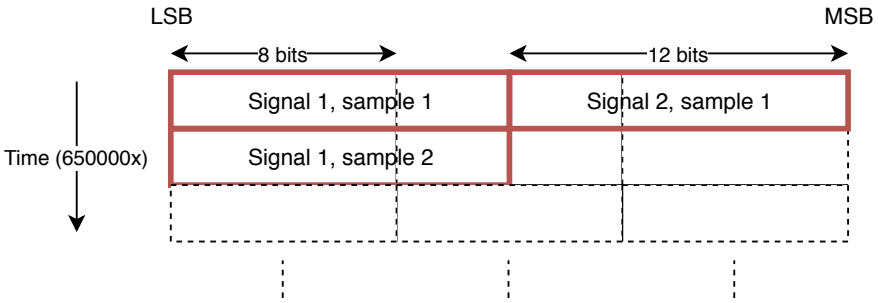

Figure 3.5: Format 212 encoding used for the lead voltage files in the MIT-BIH dataset

Since only the MLII lead is required, only one column of data is selected from the patient files. An algorithmic depiction of the data extraction process is shown in Algorithm 3.1.

Algorithm 3.1 ECG data reading algorithm

```
headerdata = ReadHeaderFile();
if Contains(headerdata.leads,'MLII') then
    samplerows = ReadDataFile();
    for all samples do
        samplerow_12bit = From3x8To2x12Bits(samplerows(i))
        sample_12bit = SelectFrom(samplerow_12bit,'MLII')
        signal_mV(i) = (sample12bit - headerdata.zero) × headerdata.gain
        time(i) = i × headerdata.sfreq
    end for
end if
return signal_mV
return time
```

3.2.3.2 Data Processing

For the next step in the three step process, namely feature detection, as part of the design space exploration (see Section 4.3) a selection of four algorithms were implemented and compared:

Table 3.1: Selection of ECG Feature Detection algorithms

Name	Features Detected	Domain
Pan-Tompkins [42]	QRS-start, QRS-end	Time
Zong [43]	QRS-start, QRS-end	Curve Length
Modified Li [44]	QRS-start, QRS-end, R-peak	Wavelet
Tekeste [45] (impl.: [46])	QRS-start, QRS-end, R-peak, P-start, P-peak, P-end, T-start, T-peak, T-end	QRS: Curve Length, Others: Wavelet

3.2.3.3 The Pan-Tompkins Algorithm

The Pan-Tompkins algorithm [42] is one of the oldest QRS detection algorithms and uses simple filtering and thresholding in the time domain to find the start and end of the QRS-complex as seen in Figure 2.12. It maintains an adaptive threshold that finds R peaks and scans backwards and forwards to find the QRS-complex. Since it is not used in the final design only a summary of the algorithm is provided. Extensive effort was put into mimicing the algorithm from the original paper. For more information please refer to the original paper.

Algorithm 3.2 Pan-Tompkins QRS detector summary

```
signal_mV = ApplyLPF(signal_mV)
signal_mV = ApplyHPF(signal_mV)
signal_mV = Square(signal_mV)
signal_mV = MovingWindowIntegrate(signal_mV)
signal_mV = CorrectForFilterDelay(signal_mV)
for all signal_mV do
    if signal_mV(i) >R_thresh then
        QRS_start(end+1) = ScanBackwards(signal_mV,i)
        QRS_end(end+1) = ScanForwards(signal_mV,i)
        R_thresh = Adapt(R_thresh,signal_mV(i));
    end if
end for
return QRS_start
return QRS_end
```

A plot of the relevant signals in the Pan-Tompkins is presented in Figure 3.6. The first 0.05 minutes of patient 100 are used to generate this image. The top plot shows the unfiltered input signal with QRS-complex start and end marks alpha and omega. The bottom plot shows the filtered signal on which thresholding is performed.

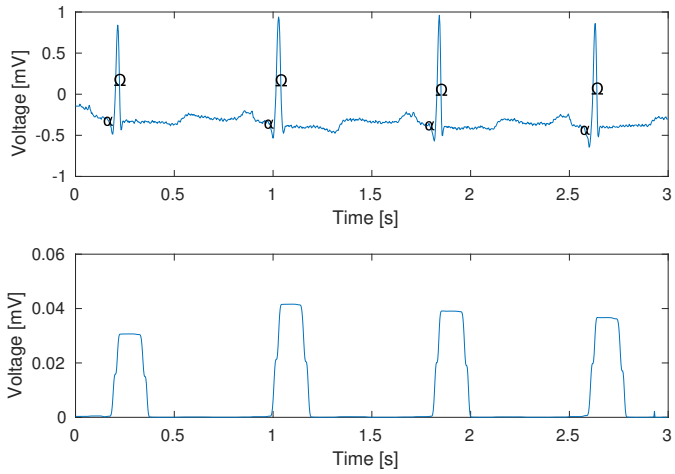

Figure 3.6: Relevant signals in the Pan-Tompkins algorithm applied to patient 100

3.2.3.4 The Zong Algorithm

The central feature of the Zong algorithm [43] is the Curve Length Transform (CLT). The Curve Length Transform calculates the length of a signal in a time window around each point. Before this transform is used a notch filter is applied to the input ECG signal to remove power supply noise. Unlike Pan-Tompkins the algorithm by Zong et al. performs a CLT before applying a similar adaptive thresholding scheme. Since the algorithm described in the original paper does not reveal all the required details and since the cited source code [47] is also slightly different in crucial locations our implementation and its results cannot be directly compared to the results in the paper.

Algorithm 3.3 Zong QRS detector summary

```
signal_mV = ApplyNotchLPF(signal_mV)
signal_mV = CorrectForFilterDelay(signal_mV)
signalCLT_mV = ApplyCurveLengthTransform(signal_mV)
T_thresh = InitialGuess(signalCLT_mV(1...N_learn))
for all signalCLT_mV do
    if signalCLT_mV(i) >T_thresh then
        QRS_start(end+1) = ScanBackwards(signalCLT_mV,i)
        QRS_end(end+1) = ScanForwards(signalCLT_mV,i)
        T_thresh = Adapt(T_thresh,signalCLT_mV(i));
    end if
end for
return QRS_start
return QRS_end
```

A plot of the relevant signals in the Zong algorithm is presented in Figure 3.7. Note that although a completely different domain is used compared to Pan-Tompkins the resulting curve is similar. Note that around the R peak the curve length is maximized: this is the result of the high amplitude changes around the R peak that allow for each R peak detection. Again, alpha and omega are used to denote QRS starts and ends.

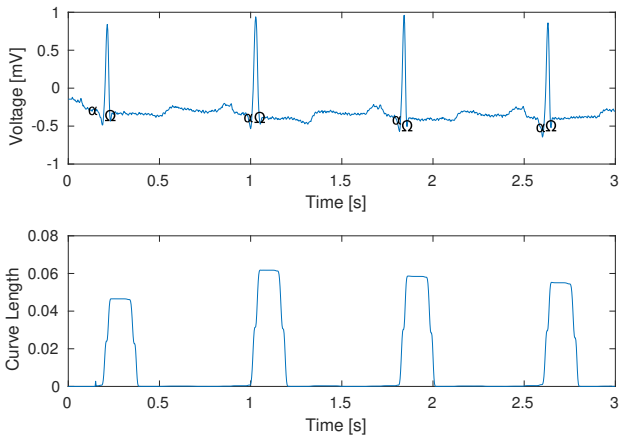

Figure 3.7: Relevant signals in the Zong algorithm applied to patient 100

3.2.3.5 The Modified Li Algorithm

The third candidate is an adaptation of the algorithm presented by Li et al. [44] It first performs the Discrete Wavelet Transform (DWT) on the input signal and then does adaptive thresholding and zero crossing detection in the wavelet domain. In the original publication the Quadratic Spline Wavelet (QSW) [48] is used. Using this wavelet the DWT transformed signal tends to produce zero crossings at sharp changes in the untransformed signal. For ECGs, these zero crossings hint at the locations of QRS-start, QRS-end, R-peak.

To reduce computational cost of the transform the a trous version [49] of the DWT is used in our implementation.

Table 3.2: Wavelet transform specifications of modified Li

Property	Value
Transform Type	DWT, *algorithme a trous*
Low Pass Wavelet $g_1[n]$	[1/8, 3/8, 3/8, 1/8] (QSW)
High Pass Wavelet $h_1[n]$	[2, -2] (QSW)
Transform Depth	4 detail + 1 approximation

The a trous version of the DWT performs the following computation on the input signal x[n] where each wavelet is upsampled by a factor of two compared to the previous one:

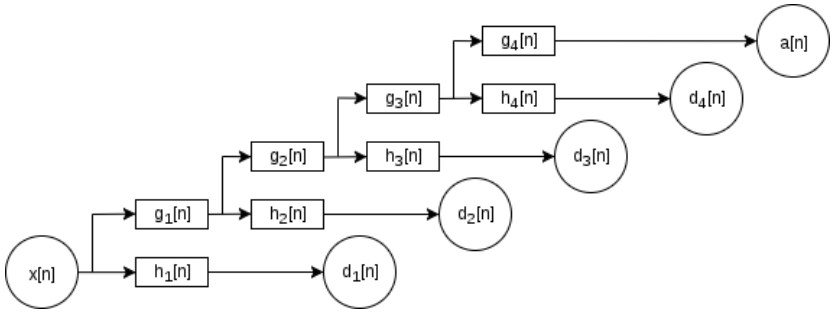

Figure 3.8: The filter bank that implements a 4 detail + 1 approximation DWT *algorithme trous*

After the DWT is performed the signals in Figure 3.9 are returned. As with the previous algorithms the α and Ω text points are used to denote QRS starts and QRS ends that are the end result of the algorithm.

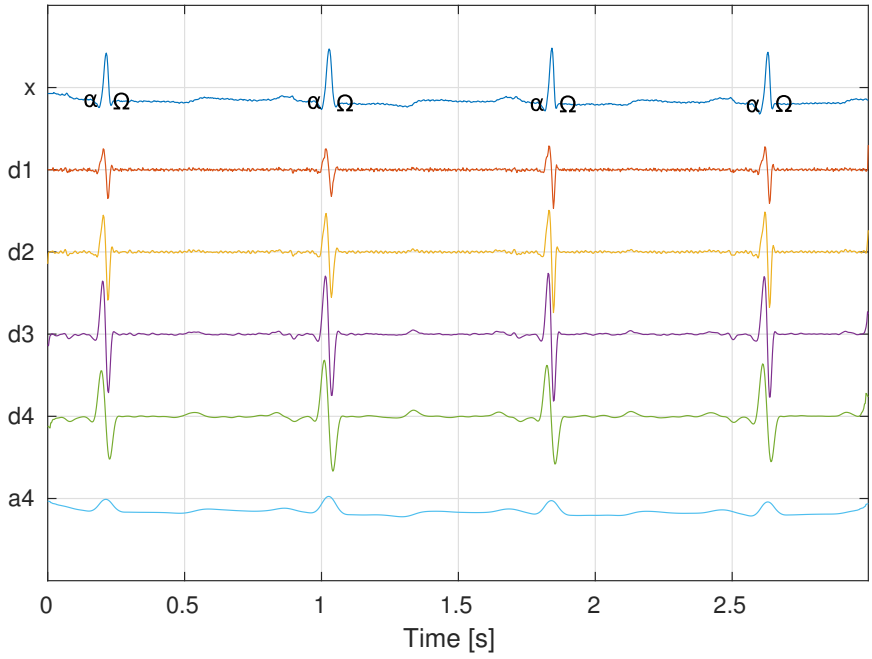

Figure 3.9: Results of a 4 detail + 1 approximation DWT *algorithme a trous*

As with the pictures of other detectors the top signal x[n] is the first 0.05 minutes of patient 100 in the MIT-BIH database. This signal is decomposed into a series of four detail coefficients d[n] and a remainder a[n]. These coefficients represent the occurence of wavelet shapes g[n] in x[n] of increasing temporal scale. This is roughly comparable to Fourier Decomposition where a signal can be decomposed into coefficients of a series of four sine waves of increasing frequencies. One important difference is that the sum of these detail levels and approximation is not equal to the original signal.

Since most ECG features are found at the first four detail levels, the remainder a[n] is ignored and not decomposed any further. Roughly speaking, the R-peak feature in x[n] is then found where nearby high amplitude opposite modulus maxima (MM) are found at all four detail levels d[n] within a certain window. Initial scanning is done at d_4[n] The zero crossing of the line between high positive and high negative detail coefficients is the location of the R-peak. From these points subsequent scanning backwards and forwards until signal amplitude drops reveals QRS-start and QRS-end. A similar approach is used in the Tekeste algorithm explained in Section 3.2.3.6 for detection of the P-peak and T-peak.

Algorithm 3.4 Modified Li QRS detector summary

```
[d1,d2,d3,d4] = DWTWithHoles(signal_mV,coeffs)
for i = 1:length(d4) do                                   ▷ Build reference peak list from d4
    if IsModulusMaximum(d4(i)) AND MoreThanThreshold(d4(i)) then
        if FarFromPreviousPeak(d4(i)) OR OppositeSignOfPreviousPeak(d4(i)) then
            d4peaks(end+1) = i
        end if
    end if
end for
d4peakpairs = RemoveLongPairs(d4peaks)                    ▷ Filter peaks to valid peak pairs
d4peakpairs = RemoveLonePoints(d4peakpairs)                                        ▷ idem
for i = 1:length(d4peakpairs) do
    for j = 4:2 do
        dj = SelectDetailLevel(j)                         ▷ If the peak is also there in d(j)...
        djpeakpairs = SelectPeaks(j-1)                      ▷ ... add it to peak list d(j-1) ...
        if IsModulusMaximum(dj(i)) AND MoreThanThreshold(dj(i)) then
            if FarFromPreviousPeak(dj(i)) OR OppositeSignOfPreviousPeak(dj(i)) then
                djpeakpairs(end+1) = dj(i)                ▷ ... if it matches requirements
            end if
        end if
    end for
end for
for i = 1:length(d1peakpairs) do                          ▷ Get R, QRS from gold list
    R_peak = ZeroCrossing(d1peakpairs(i),d1peakpairs(i+1))
    QRS_start(end+1) = ScanBackwards(signal_mV,R_peak)
    QRS_end(end+1) = ScanForwards(signal_mV,R_peak)
end for
return R_peak
return QRS_start
return QRS_end
```

3.2.3.6 The Tekeste Algorithm

The algorithm proposed [45] and implemented [46] by Tekeste et al. combines the work of Zong on the Curve Length Transform and the novel work similar to the approach by Li to detect QRS-start, QRS-end, P-peak and T-peak features. A block diagram of the algorithm is replicated in Figure 3.10.

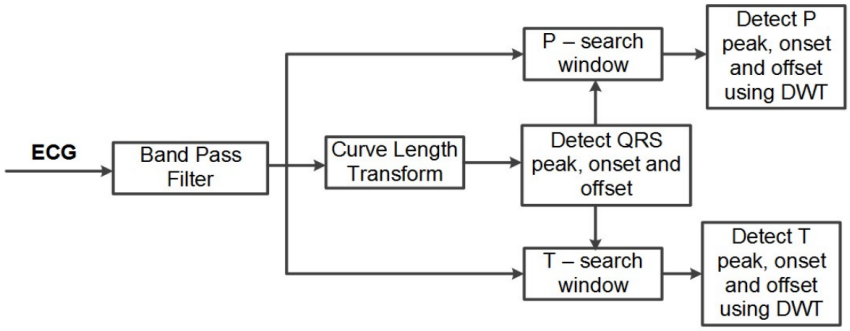

Figure 3.10: Block diagram of data flow of the algorithm proposed by Tekeste et al.

In the implementation of this algorithm in the SNN device the data flow from the raw ECG signal to the QRS detection block is equivalent to the one used for the Zong algorithm. For more details regarding these processing steps see Section 3.2.3.4.

The P-peak and T-peak feature detection part is based on the same concept of the Li algorithm: finding nearby high amplitude opposite modulus maxima (MM) in all detail levels of a wavelet transformed ECG signal. However, unlike Li where these MMs are used to find the R-peak and surrounding QRS-complex, the algorithm by Tekeste attempts to find the much smaller P-peaks and T-peaks by scanning for MM pairs in the region around the QRS-complex found by the Zong algorithm. For more information on the details of our adaptation please refer to the work by Eralp Kolagasioglu [39].

3.2.3.7 Output Matrix Format

For each succesful reading of a patient's `.dat` file a matrix of ECG features is computed. This matrix is dimensioned as m by n. Each column of length m stores the list of features from a single beat in the `.dat` file of a patient. Therefore if the Feature Detector algorithm detected n beats in the readings, the resulting matrix will have the size m by n. This layout is depicted in Figure 3.11.

The final list of ECG features extracted contains $m = 25$ items. This list is composed of the features returned by the algorithms themselves (see Table 3.1) and values derived from those. Derived features include computed heart rate from R peak counts over time, R-peak to R-peak intervals, QRS-end to T-peak and so forth.

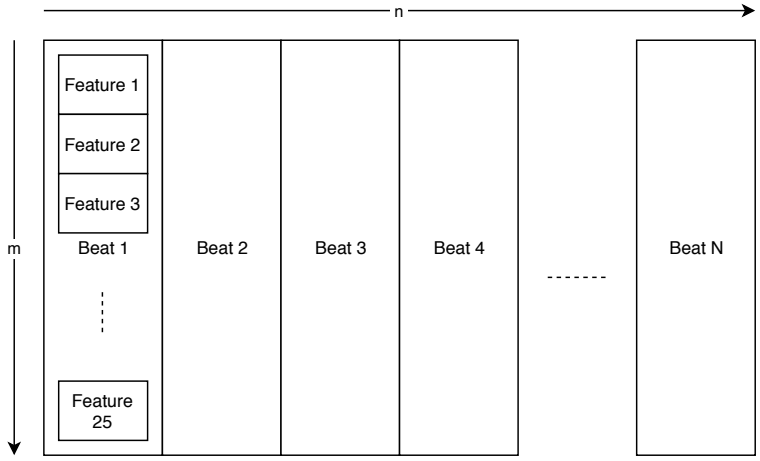

Figure 3.11: Output matrix data format

This format provides an excellent information density increase. To compute this, and give an idea of the scale of this database, let us look at the first patient in MIT-BIH who is a very average patient with average readings. Patient 100 is turned into a matrix of 25 by 2273 32 bit floating point data points or 217 KiB of data, resulting in a compression ratio of 11.7% from the original raw file of 1.95 MB. If needed that figure could even be much lower as the original data only provides 12 bits of precision. The full MIT-BIH dataset is turned into a database of 20.81 MiB of data, resulting in a compression ratio of, again, 11.7% from the original raw file set.

Table 3.3: ECG beat feature list and labels

Index (m)	Name	Comments
1	Heart Rate	
2	R - RMS	not implemented
3	QRS - Width	
4	R - Voltage	
5	QR - Width	
6	RS - Width	
7	P - Exists	
8	PR - Interval	
9	PR - Segment	
10	P - RMS	not implemented
11	P - Voltage	
12	P - Width	
13	T - Exists	
14	PT - Interval	
15	QT - Interval	
16	ST - Segment	
17	ST - Interval	
18	T - RMS	not implemented
19	T - Voltage	
20	T - Width	
21	RR - Prev RR	
22	PP	Basis for heart rate
23	PP - Prev PP	
24	QRS - Lowest Voltage	
25	Annotation	Uses PhysioNet format

3.2.4 Feature Selector

This device compresses the beat array data size to reduce the amount of data processing needed by the rest of the system: the resulting compressed beat array (or parts of it) will be used as the training and testing (see Section 3.2.1) dataset of the device and is fed to the Input Encoder. This component was designed and implemented in cooperation with Eralp Kolagasioglu [39] and subsequently incorporated into the data flow of the device as shown in Figure 3.1.

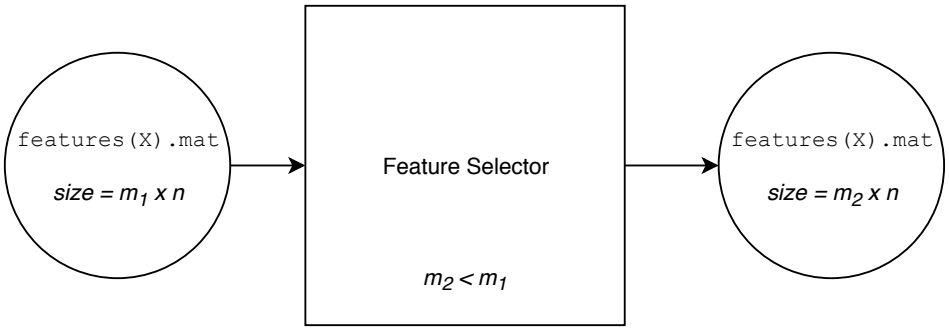

Figure 3.12: Feature selector block diagram

The beat array from the Feature Detector is described in Section 3.2.3.7. Two methods of data compression were implemented and tested: Principal Component Analysis (PCA) and correlation matrix analysis. Both attempt to convert a possibly correlated set of data into a reduced set of data that is able to statistically describe the full set of data to a user defined extent. From this it follows that the ideal reduced set is the set where the correlation between each set item to all other set items is minimized and correlation between set items and nonset items is maximized.

3.2.4.1 Correlation Matrix Inspection

The correlation matrix or the related covariance matrix are statistical tools that can be used to discover inherent relationships between items in a set of measurements. A computation of the correlation matrix of the Feature Detector output matrix of MIT-BIH patient 100 is shown in Figure 3.13.

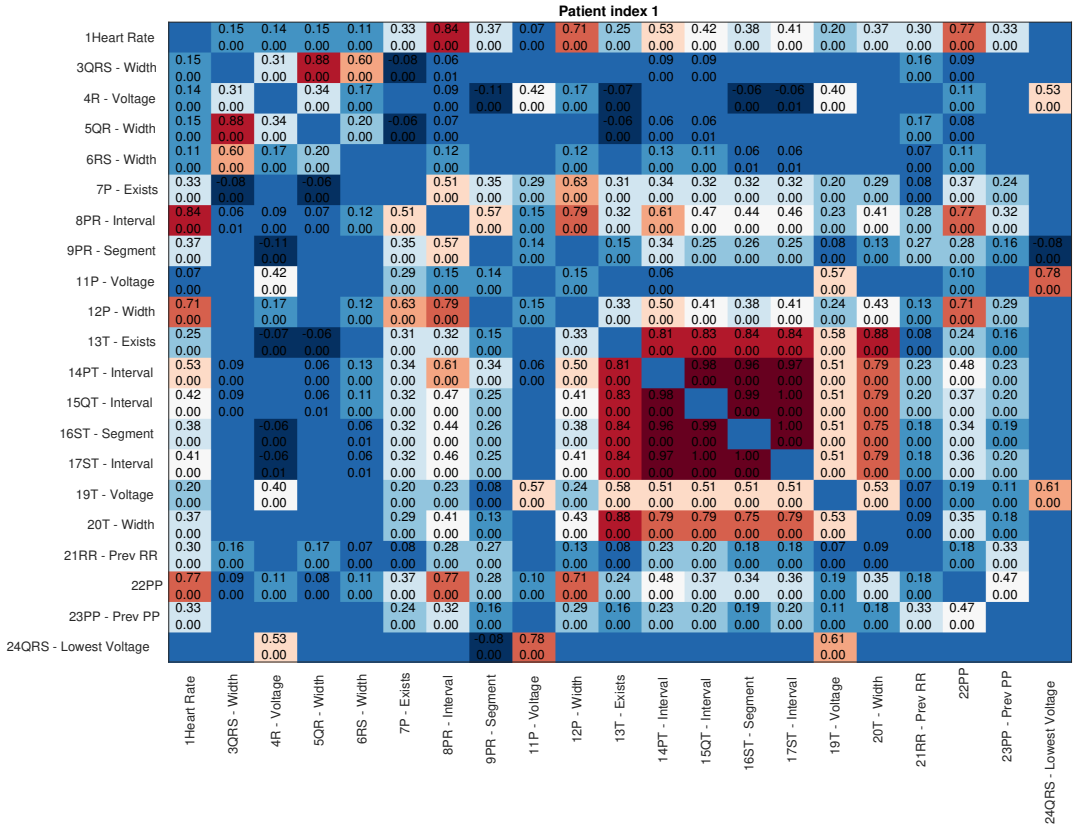

Figure 3.13: Correlation matrix computation of all detected features of patient 100 (MIT-BIH patient index 1)

In this figure a high positive correlation between two features (items in the set) is shown as a red block while a high negative correlation between two features is shown in dark blue. Only correlations with a P value of less than 0.01 have their R value (top item) and P value (bottom item) written in their corresponding box. Therefore the targets of this method are the blocks with saturated red or blue colors and text written in them.

Looking at this result it is possible to manually delete features from the output matrix by inspection. For example features 13 up to and including 20 are consistently correlated through the whole half hour recording of this patient. Therefore all but one of these features can be removed and the remaining features can be statistically derived from the behaviour of the selected feature.

3.2.4.2 Principal Component Analysis

Principal Component Analysis (PCA) is a statistical procedure that can be used to remove correlated pairs within a set of observations. It applies algorithms like eigenvalue decomposition and singular-value decomposition (SVD) to the correlation matrix shown in Figure 3.13 to reduce the amount of items in the set needed. If applied properly it informs the user of how much information is lost as a function of how many (and which) items are deleted. This is a more automated approach compared to manual correlation matrix reading. This option was implemented, investigated, and ultimately discarded in favor of matrix inspection by Eralp Kolagasioglu [39].

3.2.5 Input Encoder

The Input Encoder is tasked with converting the list of ECG features from an analog format (implemented as floating point) to the language of SNNs: the action potentials or spikes. This is done in a real time fashion where each beat is considered a sampling point of that real time signal.

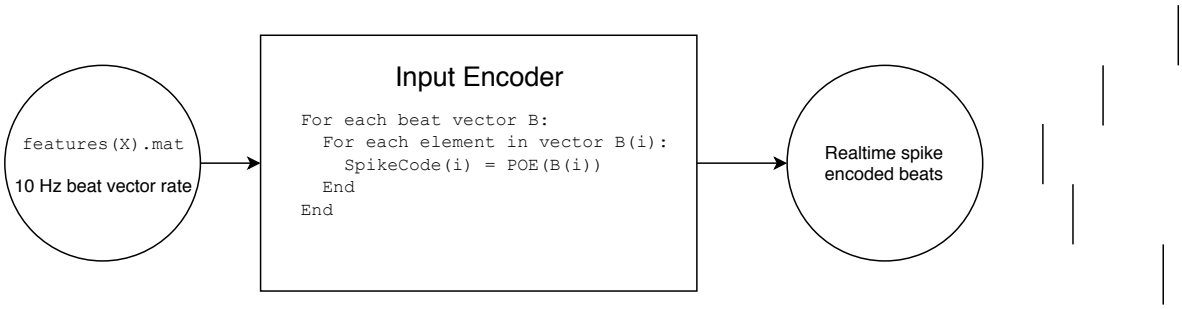

Figure 3.14: Input Encoder block diagram

Since the output of this block is directly connected to neurons in the Core SNN Simulator we need to look at how we can stimulate these neurons and the subsequent layers. For the computationally simple neuron models used (Izhikevich and Leaky Integrate & Fire) it is possible to provide input in three distinct ways:

1. Generating current waveforms from input data using DSP logic and providing them as inputs to the input layer neurons of the SNN. This way one can say that the input neurons become part of the Input Encoder as a separate additional filter to the input data.

2. A special case of the above option is accepting current waveforms and feeding them through to the input neurons directly. In this way the Input Encoder is implemented in an SNN.

3. Generating spike patterns in a separate DSP Input Encoder block and using these spikes directly as the outputs of the input layer neurons of the SNN. These neurons can be considered as fake placeholders that do not generate activity themselves and whose output connectivity is used to transport spikes to subsequent layers.

Although multiple encoding schemes were implemented and tested for the format of the ECG output data matrix returned by the Feature Detector only the *population offset encoding* (POE) [10] scheme is used. Since we believe that this algorithm produces a satisfactory temporal spike encoding, coupled with project time constraints, no alternatives were investigated.

3.2.5.1 Population Offset Encoding

This method uses the third option in the list provided above. The POE scheme accepts values in the range of [0 ... 1] in real numbers and encodes this information into the timing of spikes generated by a population of neurons. Spike rate plays no role here as each neuron spikes only once for each input signal sample. In our implementation this scheme is applied to each feature from each beat individually, resulting in up to 25 POE blocks in parallel.

Beats are provided to the device in real time with a frequency of 10 Hz. For each beat presented to the device all selected features of that beat are encoded using POE blocks. In the time domain this results in a scheme that is set up comparable to Figure 3.15. Note that this figure presents a generic configuration customized for explanatory purposes not used in the device.

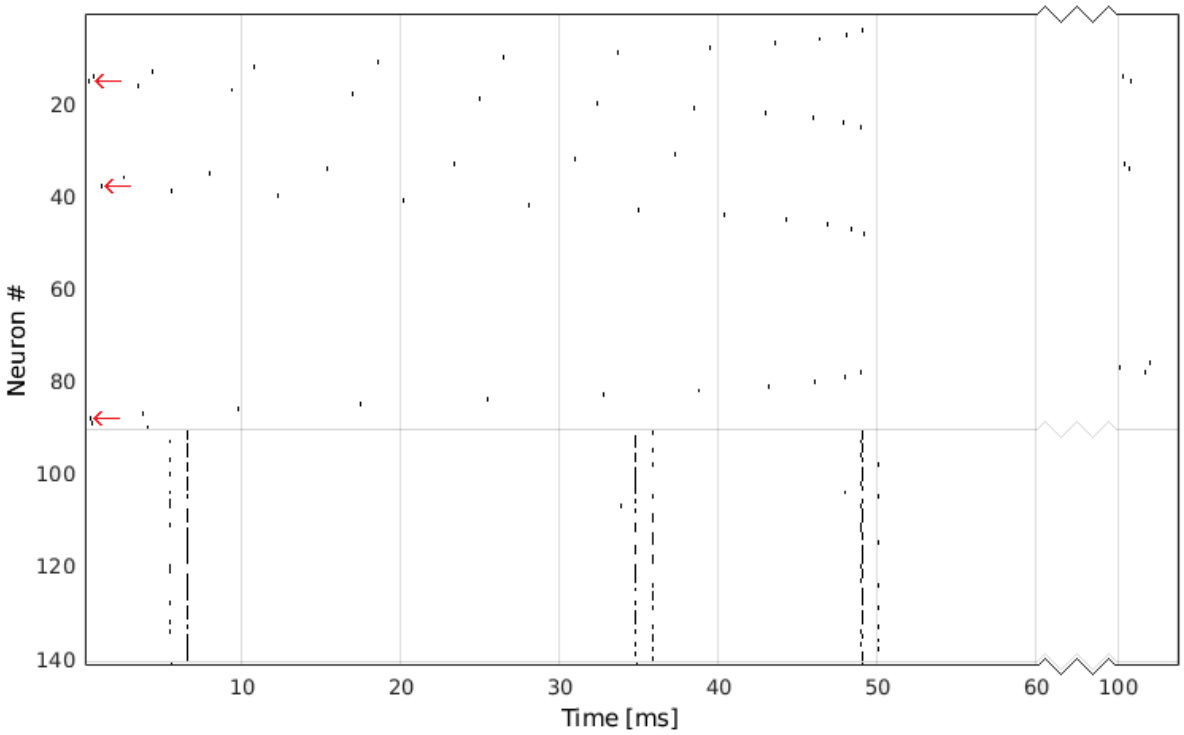

Figure 3.15: Realtime flow of spike timing based information through time with POE. In this example, the input neuron range is from 1 up to 90, divided into 3 POE blocks of 30. The red arrows highlight the most sensitive neuron in each POE block: POE block 1 encodes a value of 0.6 with neurons [1 ... 30], POE2 encodes 0.4 with [31 .. 60], and POE3 encodes 0.95 with [61 ... 90]. Neurons responding to the input are numbered 91 up to 140.

Each beat that is composed of up to 25 items depending on Feature Selector settings is encoded into spikes, where each individual feature is sent into its own POE unit in parallel. All of these blocks are configured so that they start and stop generating output at the same time. This active time is a percentage of the input signal period and is called *maxoffset*. In Figure 3.15, the input signal frequency is 10 Hz and *maxoffset* is set to $\frac{1}{2}$ resulting in an active time of 50 ms. Depending on network delay this leaves between 50 ms and 100 ms of time for the network to respond to this pattern. When the next input signal sample is presented, all neuron capacitances are reset and the whole process repeats.

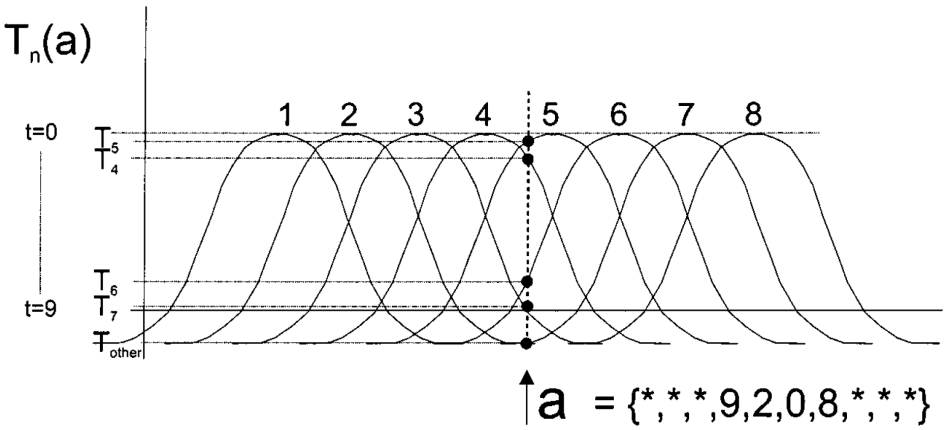

Figure 3.16: Single population conversion function from the real domain into action potentials [10]. In this example, $N = 8$ neurons are used to encode a value from 0 (leftmost on x-axis) to 1 (rightmost on x-axis). The example value a to encode has a value of 0.5 and is encoded into the spike times shown. The values in the curly brace array denote the time to spike of each neuron where lower is faster.

For each of the parallel encoders the POE scheme from Bohte et al. [10] is used, as shown in Figure 3.16. Each consecutive encoder is assigned a consecutive identically sized group of neurons from the input neuron group where encoder 1 is assigned input neurons 1 up to N, encoder 2 is assigned neurons N+1 up to 2N, and so forth. For the maximum of 25 items, 25N neurons are used.

We will now zoom into a single POE unit that is tasked with encoding consecutive values of one specific feature. For this feature the range of values to be expected is computed before any encoding is done. The minimum and maximum values found are usually assigned the value 0 and 1 in the encoding scheme input. When encoding, the feature value presented is normalized to this range (can be below 0 or above 1) and assigned a value in the range (a in the image). For all neurons assigned to this unit the time to spike is computed by Equation 3.1.

$$a(i) = exp(-\frac{(inputvalue - sensitivitycenters(i))^2}{receptivespacing}) \tag{3.1}$$

$$a(i)_{discrete} = round(a(i) \times maxoffset) \tag{3.2}$$

Each consecutive unit neuron has a certain consecutive value to which it is most sensitive (spikes earliest) stored in *sensitivitycenters(i)*, i loops through all the unit neurons, *receptivespacing* modifies the width of all the bell curves, and recall that *maxoffset* determines the maximum active time as seen in Figure 3.15 where it is set to $\frac{1}{2}$.

In the image the time to spike $a(i)_{discrete}$ ranges from $t = 0$ (fastest) to $t = 9$ (slowest). These step sizes and counts determine the resolution of the output signal.

One more variable to configure exists and is called *activationminimum*. In Figure 3.16 this is presented as population neurons spiking at time '*' for example value a. This spike time means that the neuron is so unsensitive to the value a to be encoded that its extremely late response spike time (later than 9 in the image) is considered noise and not encoded. In Figure 3.15 this feature is also implemented and configured so that spikes generated after 49 ms are not generated. Note that using this option violates the one spike per neuron per signal sample property of POE.

3.2.6 SNN Simulator

The SNN Simulator performs a simulation of the neuromorphic component in the device: the SNN. It is a simulator that uses discrete time steps of 0.1 ms to solve the equations of the neurons and synapses in the network.

Before simulation starts, the network itself is generated by the Network Generator (not shown in the system overview) that introduces a noise factor into instances of the device: although the distribution of the initial conductivity values of synapses is known, the actual values samples from this distribution are computed with the help of a random number generator. To ensure consistent results, the seed to this random number generator is set to a fixed configurable value. This noise factor will be investigated in Section 4.7.

The core loop of the simulator that performs the time stepped calculations is the only part that has been written in C, and has been designed in such a way that it can be easily converted into neuromorphic hardware in the future. It is currently capable of being hardware synthesized into digital logic without loss of functionality.

Due to the discrete time step based simulation architecture, a slight loss of spike timing accuracy has resulted. This type of simulation architecture is also known to introduce errors into differential equations that it 'solves' through discrete time.

Experiments have been performed with variations in simulation time step, which have shown thatthe current step of 0.1 ms is a good balance between computational load and simulation accuracy.

3.2.7 Output Decoder

Finally the Output Decoder applies various analysis techniques to output neuron spiking activity that help to quantify network response. These techniques can be divided into two groups: a group that returns image-based analysis like the actual mapped points of the SOM, and a group that compresses output activity into scalar Figures of Merit (FOM). Only the last group will be discussed here.

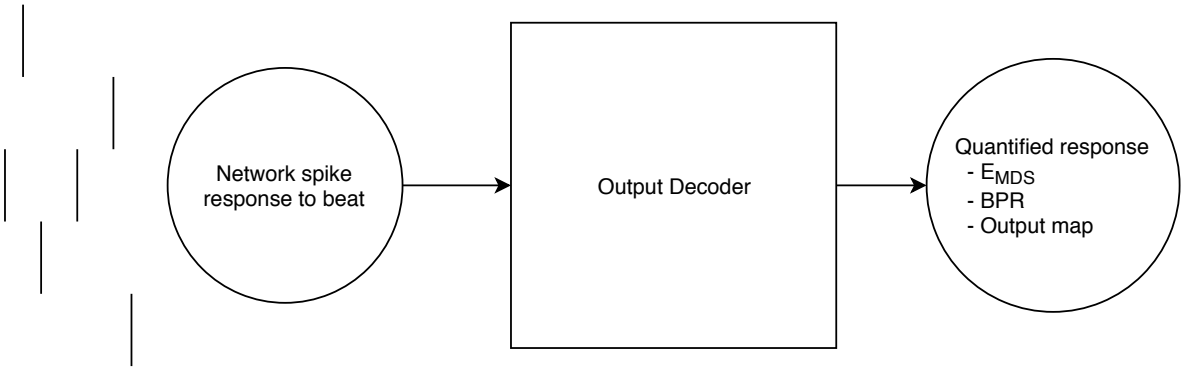

Figure 3.17: Output Decoder block diagram

The scalar FOMs that are used in this device are the Multidimensional Scaling Error (E_{MDS}) and the Beat Pair Recognition (BPR).

3.2.7.1 Multidimensional Scaling Analysis

The E_{MDS} [50] is a tool that can assess the ability of a SOM [35] to map its input dataset to its output. Effectively what an SOM performs is a dimension reduction process: starting from the dimensions of a beat (see Section 3.2.4) in its predefined input space it is tasked to reduce these n dimension to a two-dimensional map, which is its output space. The quality of this dimension reduction, also called map formation, can be assessed my many tools including the aforementioned E_{MDS} that is used in this project.

Ideally, the distance metric between two data points in the input space and output space should be perfectly related. This means that identical points in the input space should be mapped to equal points in the output map. Similarly, maximally distant (again dependent on the distance metric used) should be mapped to maximally distant points in the output map.

The E_{MDS} is computed as follows:

$$E_{MDS} = \sum_{i=1}^{N} \sum_{j=1}^{j<i} (F(i,j) - G(M(i), M(j))) \tag{3.3}$$

In this equation, N is the total amount of items in the dataset (beats in this case), *F(i,j)* is a normalized distance or dissimilarity computation function in the input space

for any beat pair i and j of beats, *M(i)* is the location in the output space of beat i, and *G(M(i),M(j))* is a normalized distance or dissimilarity computation function in the output space for beat pair i and j transformed by M.

Essentially, for all possible combinations in the dataset of size N, the difference or *error* between the normalized input space and output space coordinates is computed and summated. Usually, as is the case for this project, the end result is normalized for the total amount of combinations iterated through to be able to compare results between different dataset sizes.

Both *F(i,j)* and *G(M(i),M(j))* require a suitable normalization function so that the maximum possible distance reliably returns 1 and the minimum distance returns 0. For the input dimension *F(i,j)* the normalized sum of normalized differences of all beat features is taken. The normalization factor for beat features is identical to the optimized data range for that feature (see Section 4.5.1.2). For the output dimension the maximum possible distance in the toroidal map was taken as the normalization factor (see Section 3.2.6) which is equal to half the diagonal length of the map.

A value of zero for E_{MDS} is interpreted as a perfect mapping result since output distances between all beats are perfectly related to input distances between all beats. For a purely randomly generated mapping a value of 0.5 for E_{MDS} is expected. Values above 0.5 require performance worse than just rolling a die and are practically useless.

3.2.7.2 Beat Pair Recognition

This brings us to the next FOM: Beat Pair Recognition, or *BPR*. This should be combined with E_{MDS} as that alone does not reveal if the device does not recognize certain beats. In the state of the art, failure of a SOM to generate a *coherent* output is either ignored or a maximum possible E_{MDS} of 0.5 or 1 is assigned to that item.

In this thesis the *BPR* figure is used. It reveals the percentage of *pairs* of beats that are coherently detected. Coherence in this context means that a single interconnected population of neurons with a maximum size of 25 out of 100 map neurons is allowed to spike first. This population of neuron that spikes first is called the winner. If multiple unconnected neuron groups spike first at the same time, if the single population is too large, or if no neuron spikes at all in response to a beat, the response is deemed *incoherent*. For each *pair* of coherent beats the *BPR* is incremented.

As an example computation of the *BPR* is shown in Table 3.4.

Table 3.4: Example computation of the *BPR*

Property	Value	Notes
N	100	Number of beats in testing set
N_{pair}	4950	Number of possible pairs: 1 + 2 + ... + 99
Coherent responses	60	Number of beats recognized
Coherent pairs	1770	Number of pairs for which E_{MDS} can be computed
BPR	35.8%	(1770/4950)*100

3.2.8 The Self-Organizing Map

Standard multilayer feed forward networks as described in Section 2.3.1 are conventionally meant to be used in combination with supervised learning protocols. However, the scope of the bigger project that this thesis is a part of is limited to unsupervised networks. Therefore one type of neural network designed for unsupervised learning has been selected as the network topolgy and it is called the Self-Organizing Map (SOM), derived from the Kohonen Map named after its inventor [51]. The way that the SOM is integrated into the final device is depicted in Figure 3.18.

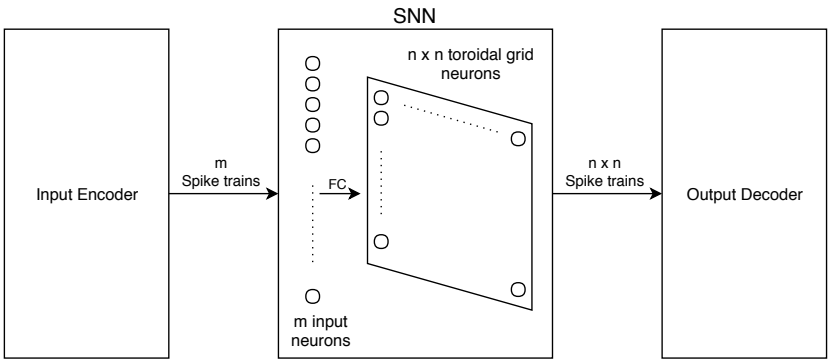

Figure 3.18: System overview of SNN with accompanying input and output data flow

The multilayer properties of neural networks described in Section 2.3.1 do not directly translate to the SOM layout since there is only an input layer and an output layer with a single computation like f in between. However a complex feedback network from each output layer neuron to all other output layer neurons is present that provides a second layer of computation like g.

The input layer is implemented as a row of neurons that directly feed binary spike trains received from the Input Encoder to their output as explained in Section 3.2.5. They do not perform any computation on their input. The output layer is a two dimensional grid of neurons that receive spikes from the input layer. Their response to the input layer is fed to the Output Decoder and are analyzed to compute E_{MDS} and BPR.

The first computation f implemented by the synapses from the input layer to the output layer is where learning happens: these synapses implement STDP or TSTDP plasticity and adapt their conductivity based on spike timing activity of the neurons connected at both ends.

3.2.8.1 SOM Feedback Function & Spike Based Mapping

Up to this point, apart from g, we have basically described a simple single layer SNN perceptron. The core functionality of the SOM is inside this feedback of the output layer g. Each neuron in the 2D grid is connected to all other neurons in the grid (except itself like an XOR) with a conductivity that scales with the distance between each pair of neurons. The neighbourhoud function described in this thesis is called the Mexican Hat function and is described and investigated by [52].

It is computed by three parameters a, b, and r that determine its negative magnitude, negative spread, and radius respectively:

$$hat = (1 + a)exp\Big(\frac{-d^2}{2r^2}\Big) - (a)exp\Big(\frac{-d^2}{2(br)^2}\Big) \tag{3.4}$$

When configured correctly, *hat* describes the feedback conductivity between two neurons in the 2D output layer depending on the distance r between these two neurons. The output layer is shown in Figure 3.18 as a n by n grid of neurons. For low r, potentiation will occur. For higher r, depression will occur. For even higher r, no significant conductivity is present. This characteristic allows neurons that are activated by the input layer to potentiate close neighbours in an attempt to help them also spike to the same or related input patternsn. Should a neighbouring neuron also spike as a result of this feedback, the synapse from the input layer to that layer will then be strengtened. This process allows neighbouring neurons to respond to similar, but slightly different, input patterns. This in turn describes the concept behind mapping in a spiking Self-Organizing Map.

Device Design Space Exploration

4

The device described in Section 3.2.6 can be configured and simulated in an infinite amount of ways assuming infinite computing power and storage is available. Think of each of those configuration options as a *dimension* in a certain space where the available range of values to set that option to is the range of that dimension. If we add up all of these dimensions to a space we call that a *design space*.

This design space of this device is made extremely complex, or in other words difficult to find a usable set of dimension values, by the fact that there are so many of these dimensions available. This rules out approaches like an exhaustive search of possible combinations of dimension values. It also makes navigating the space towards the global maximum difficult since every single dimensional value change can result in all other dimensions taking up different optimal values or ranges. Another problem is the fact that for some device blocks, especially in the SNN, the optimal ranges of various dimensions in this space is extremely small, as will be shown in this chapter.

This chapter presents an exploration of the above mentioned difficult to crack design space. The exploration reveals a subspace that produces a neuromorphic learning device that is independently capable of creating and improving a 2D mapping of a patient's ECG beats, using the biological properties discussed in Chapter 2, and is implemented using the architecture discussed in Chapter 3.

[53]

4.1 State of the Art

To the knowledge of the author no previous work is available on unsupervised categorization of beats in the MIT-BIH database, and design space exploration thereof. In the more general field of ECG classification using various type neural networks numerous works are available, but results are difficult to compare due to differences in input data sets and output metrics used.

For a state of the art comparison of more general ECG classifiers, but not categorizers, please refer to Eralp Kolagasioglu [39], Table 2.4. It lists various software and hardware implementations of classifiers and shows their accuracy, learning method, hardware costs primarily.

For a comparison of SOM map formation metrics, but with different dataset, please refer to [36] and [35]. These give a point of comparison of the performance expressed in E_{MDS} that can be expected from a SOM that can be used to assess performance of this device. Directly comparable metrics to the *BPR* are not covered in the state of the art due to differences in the ways that SNN response is assessed (see Section 3.2.7.2).

The state of the art of the Feature Detector stage comprisis mainly of the authors covered in the architecture section (see Section 3.2.3). Comparison is difficult since essential details including database preference, patient selection, lead selection, and beat type to detect selection are different and underreported in comparable works.

4.2 Chronology & Approach

To efficiently explore this high dimensional space it is essential that independent dimension groups are uncovered and explored separately, thereby drastically reducing the amount of points in the design space volume to cover. For this reason a two way split was put in place at the start of the exploration. From a top-level view looking at the five main blocks in Figure 3.1 the combined Feature Detector and Feature Selector are the most independent group of blocks available. They are also the two blocks that are easiest to test and compare to golden values. Splitting the device up this way reduces the design space volume V in the following way as a function of exploration point count N, with example values for all N being 100:

$$V_{before} = N_{FS} \times N_{FS} \times N_{IE} \times N_{SNN} \times N_{OD} \rightarrow 10000000000 \tag{4.1}$$

$$V_{after} = N_{FS} \times N_{FS} + N_{IE} \times N_{SNN} \times N_{OD} \rightarrow 1010000 \tag{4.2}$$

Therefore the workflow will be as follows:

1. Firstly, these two blocks were the first to be explored, and their configuration was fixed. This fixed configuration will therefore present a fixed dataset to the remainder of the network for the remainder of the exploration. For the remainder of this document this dataset will be called `features.mat`.

2. Secondly, using `features.mat` a usable baseline for the Input Encoder, SNN, and Output Decoder was obtained. This process starts in Section 4.5.1 and it leads to a baseline configuration shown in Table 4.5.

3. Finally, this baseline and its surroundings in the design space was explored using parameter sweeps in various dimensions of the Input Encoder, SNN, and Output Decoder. The results of these sweeps are shown in Section 4.7.7.1.

4.3 Feature Detector

The first block to be explored is the Feature Detector described in Chapter 3.2.3. As selecting and optimizing of data is done in the Feature Selector we only want this block to obtain as much correct data from raw ECG data as possible. Optimal configuration of this block, independent of the state of other device blocks, is relatively easy to find as there is a defined golden result available and its output, in the correct range of possibilities, does not vary much.

The analysis functions at our disposal are the Pan-Tompkins algorithm, the Zong algorithm, the modified Li algorithm and the Tekeste algorithm. As the work for the Tekeste option was performed by project member Eralp Kolagasioglu [39], please refer to that document for an exploration of the Tekeste based Feature Detector. The other three algorithms were algorithms written by or contributed to by the author of this document.

Although Tekeste is the only option available if features from P, T, and QRS are to be used (see Table 3.1) in subsequent device blocks it is useful to consider the other

three algorithms. This is because Tekeste uses Zong for QRS and a Li based technique for P and T detection. Also, computation and hardware cost need to be taken into account as they vary wildly between algorithms: for some use cases it might be better to only use QRS related features for learning and use the simple Pan-Tompkins algorithm instead. Therefore, for completeness sake, a shorter cost and performance analysis will be done here for these other three algorithms even though they will not be used in this document.

4.3.1 Analysis of QRS Detectors

Each algorithm except for Li was replicated from the original publication and adapted to our dataset. For the Li algorithm, remember that extensive modifications were done due to lack of details in publication and possible optimizations.

For each algorithm tuning was performed to optimize the following figures of merit: the number of correct beats detected or the true positives (TP), the incorrect detections or false positives (FP), and missed beats or false negatives (FN). All three algorithms were applied to the `.dat` file of every patient that had an MLII lead.

The reference used for testing the TP, FP and FN is the R-peak location of each beat. This R-peak is at the location of the annotations manually added to all patient data files by a human professional in the MIT-BIH dataset.

This reference is used as follows: *if a QRS-start is before an R-peak and if the matched QRS-end is after the same R-peak but before the next R-peak then a detection is true.* So the test for the QRS detectors is actually only finding the correct time range where the R-peak is. Checking for correctness of the actual points of QRS-start and QRS-end is only done by manual inspection and automated bounds checking for reasonable values.

This limitation comes from the fact that MIT-BIH only stores R-peaks reference values. In the ideal case we would like to have a database that stores QRS-start, R-peak, QRS-end, and others for all beats for all patients. To the author's knowledge only one such database exists, called the QT database, but it is of a much smaller size with only around 3600 fully annotated beats in total [54] compared to 105210 partially annotated ones in MIT-BIH.

Table 4.1: Performance of dedicated QRS feature detection algorithms

Algorithm	Count	TP	FP	FN	TPR	PPV	Complexity [39]
Reference (mean)	2287						
Zong (mean)	2291	2262	29	25	98.86%	98.66%	High
Pan-Tompkins (mean)	2279	2250	29	38	98.47%	98.79%	Lowest
Li (mean)	2274	2265	9	22	99.10%	99.58%	Highest
Reference (sum)	105210						
Zong (sum)	105387	104047	1340	1163			
Pan-Tompkins (sum)	104822	103477	1345	1733			
Li (sum)	104606	104205	401	1005			

Table 4.1 shows that the total number of beats in all files is 105210 with an average of 2287 per file. At first glance all the algorithms seem capable of detecting between 99.4% and as much as 100.2% of all beats. However, false positives need to be removed from this figure and false negatives also need to be considered.

After filtering, the modified Li algorithm is deemed the most effective because it has the highest precision (PPV) of 99.58%, meaning that its detections are more likely to be true, and because it has the highest sensitivity (TPR) of 99.10%, meaning that is it capable of finding most beats in the first place.

From a complexity standpoint the Pan-Tompkins algorithm deserves special mention. It has much lower computational demands, is easier to implement in hardware and is more easy to tune than both Zong and especially Li. Looking at these figures it can be deduced that Zong has no clear advantages compared to Pan-Tompkins. However, this table only shows the average performance for an average type of patient with average-shaped beats. For a more detailed per beat type analysis, please see [39].

Full algorithm performance results can be found in Appendix A.1.4, Appendix A.1.2 and Appendix A.1.1.

4.3.2 Analysis of P, QRS, T Detector

Work on the only candidate of this type of detector, Tekeste, is done by Eralp Kolagasioglu [39]. For completeness the results of the final testbench applied to the same dataset as for the other QRS detectors is provided. It is summarized in Table 4.2.

Table 4.2: Performance of the QRS block of the Tekste algorithm

Algorithm	Count	TP	FP	FN	TPR	PPV	Complexity
Reference (mean)	2297						
Tekeste (mean)	2291	2269	22	29	98.71%	98.97%	High
Reference (sum)	105682						
Tekeste (sum)	105380	104359	1021	1323			

Note that the total amount of beats is slightly different than in Table 4.1. This is because a different list of PhysioNet beat types was selected (see Section 3.2.2.2) as the reference set of beats to find.

Also note that the reference data only contains information as to where the R-peak is: this is at the location of the manual annotation. No information is provided for the true location of the P-peak and T-peak or if there even is one. The lack of these peaks is not uncommon. Testing for correctness of the P and T related feature detection was done by inspection.

As was to be expected the implementation of Tekeste performs very similarly to the author's implementation of Zong when looking at QRS detection. Some minor differences can be attributed to minor differences in tuning, implementation details, and the aforementioned different selection of beats to be found.

Full algorithm performance results can be found in Appendix A.1.3.

4.4 Feature Selector

The second block to be explored is the Feature Selector. As discussed in the Architecture chapter it receives output from the Feature Extractor in the data format explained in Section 3.2.3.7. Its job is to select the most useful features (rows) from this matrix and send it to the Input Encoder.

Two algorithms were investigated by Eralp Kolagasioglu [39]: PCA and manual correlation matrix optimization. The PCA option was investigated but dropped as results were not deemed an improvement compared to the simpler manual approach. Manual correlation matrix analysis was performed independently by both researchers.

4.4.1 Manual Inspection Process

Manual correlation matrix analysis was performed on the first 20 patients of the MIT-BIH database. This limitation was put into place as a step up to using the full database set and also served to speed up the DSE of subsequent blocks. This set of 20 corresponds to patient files named 100 up to and including 121. We start with a data matrix row selector containing all 25 features or rows represented by their indices:

```
i = [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25]
```

Recall that these indices are listed in Table 3.3. Then, we remove the manual beat annotations as they would only be available in scenarios where annotated data would be available for training. Note that they can still serve as backup features in case learning needs access to more data. We also remove features that were not implemented and verified in time as noted in Section 3.3. These indices remain:

```
i = [1   3 4 5 6 7 8 9    11 12 13 14 15 16 17    19 20 21 22 23 24   ]
```

Afterwards, the correlation matrix with corresponding probability values is investigated. Analysis of it shows that the features obtained from the P-wave, QRS-complex, and the T-wave tend to form groups of significant correlation ($p<0.01$, $r>0.8$). There rarely is any significant correlation between these three groups. This effect is most pronounced in the P-wave and T-wave group.

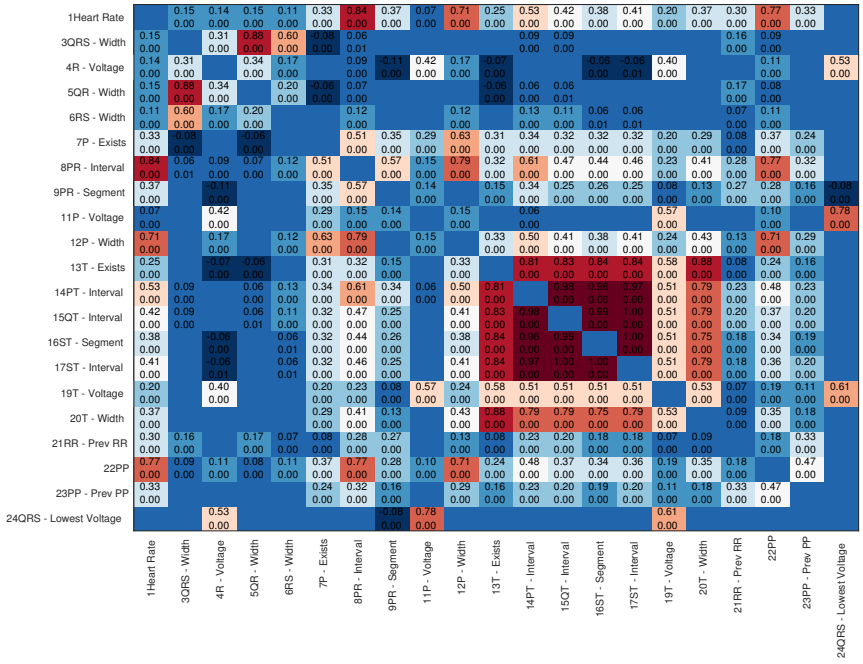

Figure 4.1: Correlation matrix for patient 100 showing correlation blocks (p<0.01) for P-exists and T-wave only

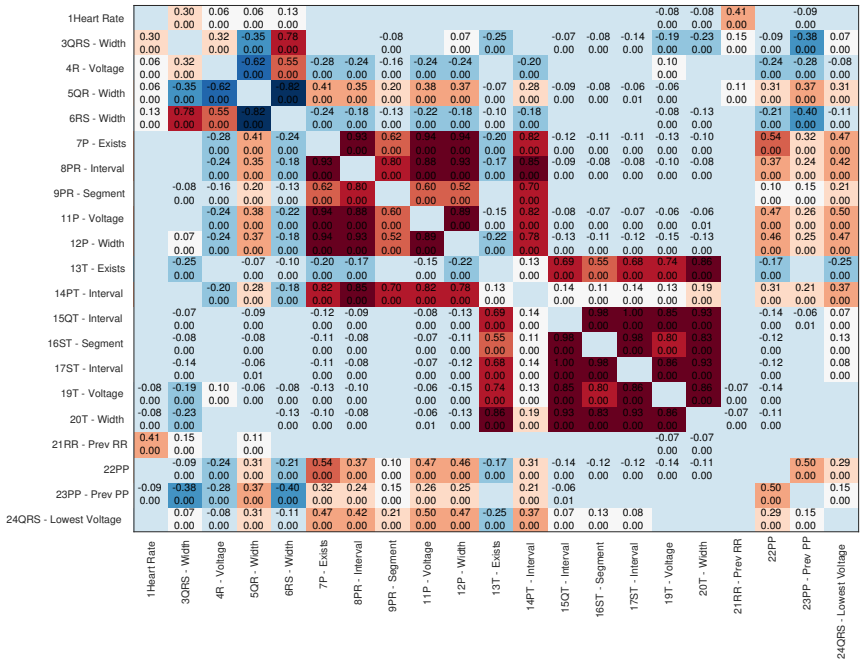

Figure 4.2: Correlation matrix for patient 102 showing rare occurence of correlation blocks (p<0.01) for P-wave, QRS-complex, and T-wave features

In the above figures two common cases of these correlation grouping expressions are shown. Patient 100 shown in Figure 4.1 is a case where the mentioned correlation blocks are not all present. Only the T-wave block is red meaning high correlation exists within this group and superfluous information is present. Only weak correlation between the P-exists and P-width features exists. In patient 102 shown in Figure 4.2 all three groupings are present.

The boolean values P-exists and T-exists, which show if the P-wave and T-wave are present or not, are the most common properties inside these groups that are highly correlated to other properties of P and T respectively. Therefore we will remove them.

```
i = [1   3 4 5 6   8 9    11 12    14 15 16 17    19 20 21 22 23 24   ]
```

Further analysis of the P-wave, QRS-complex, and T-wave correlation groups of the first 20 patients reveal that only 1 patient (5%) has significant correlation in QRS-complex (index 3 up to 6), 9 patients (45%) have significant correlation in P-wave (index 8 to 12), and all patients (100%) have significant correlation in T-wave (15-17). Furthermore 16 patients (80%) have significant correlation in a bigger T-wave portion (14-17), but this is not universal and cannot be removed for all patients. For T-wave more similar even bigger index groups up to 14-20 with high correlation are present in smaller patient counts. Raw analysis data can be found in Appendix A.2.1.

It has been decided to remove all but one of the highly correlated 14-17 index group that is present in 89% of all patients.

```
i = [1   3 4 5 6   8 9    11 12    14              19 20 21 22 23 24   ]
```

4.5 Baseline Observations

For the remainder of the device the exploration consists of the second and third steps mentioned in Section 4.2. Final results of the first part are fixed and referred to as `features.mat`.

4.5.1 Input Encoder

The third block to be discussed is the Input Encoder. It is responsible for translating a continuous stream of beat features to the language that the fourth block, the SNN, understands. This block is the first block to be explored as part of the second step in Section 4.2 and usage of `features.mat` is implied.

For the specific use case of the ECG beats only one encoding scheme was explored in the end: population offset encoding (POE). That is not the end of the story as this algorithm contains various configuration options.

4.5.1.1 Population Offset Encoding

For POE three variables can be configured and are explored here: *receptive spacing* (1), *maximum offset* (2), and *minimum activation level* (3).

The first variable, *receptive spacing*, determines the selectivity of the neurons in each POE block. Increasing this value decreases the range of input values that each neuron is sensitive to. While this value is set to 1.5 in the original publication of the scheme [10] the sensible range in our device is in the orders of magnitude [0.1 ... 10]. For the lower limit of 0.1 every neuron is at least 50% activated for any input value, while above the upper limit of 10 the neurons get so sensitive that dead spots in the input range become problematic. The average activation time is another problem experienced when reaching into the lower limits of this variable: a preference for spiking early arises in the middle of the input range making these values' spike pattern a more potent candidate for plasticity. A flat dotted line in the figures below is preferred.

Examples of the low and high cases are given in Figure 4.3 and Figure 4.4. Initially a value of 1 is selected.

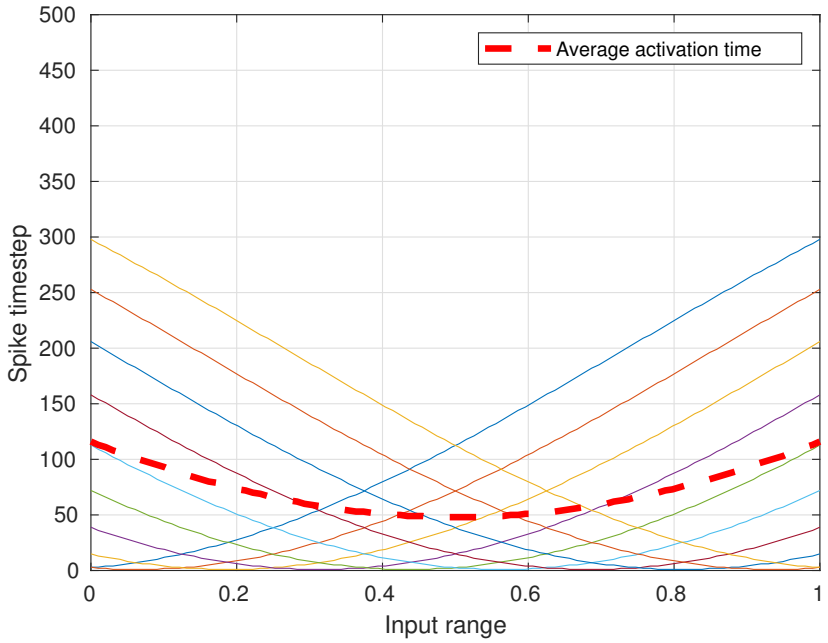

Figure 4.3: POE spike encoding function with low (0.1) receptive spacing. In this image the spike time range of $a(i)_{discrete}$ is 500 steps. $a(i)_{discrete}$ is explained in Equation 4.3

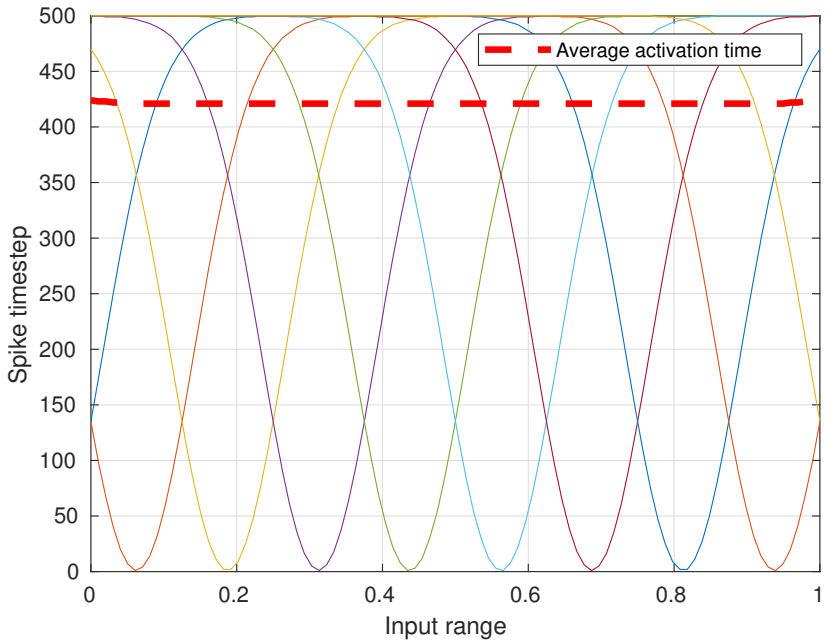

Figure 4.4: POE spike encoding function with high (10) receptive spacing. In this image the spike time range of $a(i)_{discrete}$ is also 500 steps.

The second variable, *maximum offset*, determines the duty cycle of network spike exposure. Combined with the input signal sampling frequency and receptive spacing this determines the resolution of the output signal of the Input Encoder in simulation environments where discrete time steps are simulated. This resolution can be calculated using the POE conversion function from Equation 3.1.

$$a(i) = exp(-\frac{(inputvalue - sensitivitycenters(i))^2}{receptivespacing}) \tag{4.3}$$

$$a(i)_{discrete} = round(a(i) \times maxoffset) \tag{4.4}$$

With this equation we can compute the smallest *inputvalue* change that changes the spike timing of the most sensitive neuron by the smallest step possible: changing $a(i)_{discrete}$ by 1. The POE output value corresponding to that step is computed using Table 4.3. This table contains values used in the final setup of exploration step two.

Table 4.3: Computation variable values

Variable	Value	Comments
maxoffset	0.5	50% duty cycle
sim timestep	0.1 ms	Simulation time step
sampling step	100 ms	From 10 Hz ECG beat rate
POE steps	500	Time steps in 50% duty of sampling step
POE exp. step (a)	$\frac{1}{500} = 0.002$	POE exponent step for 1 timestep
receptivespacing (b)	0.1 ... 10	Usable range, lit. uses 1.5
population size (c)	10	Single POE block

First we need to determine the most sensitive neuron responding to a certain value change. For an *inputvalue* that changes infinitesimally by x the most sensitive neuron index i_{max} (note i in Equation 4.3) can be found at the roots of the second derivative of the exponential function in Equation 4.3. This is where the rate of change in the exponential is highest. This neuron index is not the neuron that is tuned to the value nearest to *inputvalue* because that one is at or near the maximum of the exponential.

$$\frac{d\left(e^{-\frac{x_{max}^2}{b}}\right)}{dx^2} = 0 \tag{4.5}$$

$$|x_{max}| = \frac{\sqrt{b}}{\sqrt{2}} \tag{4.6}$$

$$i_{max} = x_{max}c \tag{4.7}$$

Using the neuron offset i_{max} from the center neuron of the population we can obtain the *inputvalue* step in_{step} that causes this most sensitive neuron to change the smallest possible step a. We compute this by dividing the smallest possible timestep a by the rate of change of the most sensitive neuron.

$$x_{dt} = \frac{d\left(e^{-\frac{x^2}{b}}\right)}{dx} \tag{4.8}$$

$$x_{dt} = -\frac{2xe^{-\frac{x^2}{b}}}{b} \tag{4.9}$$

$$i_{maxrate} = x_{dt}(x_{max}) \tag{4.10}$$

$$in_{step} = \frac{a}{i_{maxrate}} = 0.00074(b = 0.1), 0.0023(b = 1), 0.0073(b = 10) \tag{4.11}$$

So within the normalized range of *inputvalue* there is enough but not spectacular resolution available. Keep in mind that this is a far-reaching best possible case (using these configuration variables) since this in_{step} means that only one neuron of the whole population will change its spike time by the smallest possible step. Potential resolution problems can be e.g. mitigated by optimizing the input value range and by keeping b low.

The third variable, *minimum activation level*, did not seem to have any measurable effect on performance during this stage of the exploration and was left at -1, meaning full participation of all neurons. However, original literature [10] does mention a significant improvement in categorization performance as would be expected by preventing unrelated late-spiking neurons from spiking at all.

4.5.1.2 Dataset Composition

The composition of the training dataset has to meet certain demands for it to be useful as SNN training data. One such demand is that it needs to include the full scope of data points that are expected to be fed to the SNN after training. If this is not the case and data outside of the training data scope is presented to the network, one generally cannot predict how the network will respond. This kind of scoping problem has been known for a long time and is, among others, also discussed in the field of autonomous driving by Pomerleat et al [55].

For this project, to limit scope, during testing only values that have also been presented to the network during training will be used. In addition to that solution the SNN is helped by optimization of the minimum and maximum values corresponding to 0 and 1 in the POE blocks. These normalization values are set the minimum and maximum values found in the full training set. This ensures that the Input Encoder makes maximum use of the available dynamic range of its output and that unknown values are not able to be converted. As seen in Equation 4.8 a tighter fit around the predicted data range results in a higher input range resolution.

One also needs to make sure that overstimulation does not occur. Since an SNN learns or modifies itself to better respond to the data provided if a similar pattern of activity is presented multiple times in a row, overlearning can occur: the network will then respond strongly to the overlearned pattern while very weakly responding to other patterns. Similarly if the data does not present a balanced share of activity of patterns to be learnt it might overlearn to the overrepresented data.

Shown below in Table 4.4 is the count of occurrence of each type of beat in the output of the Feature Detector as it is fed to the Feature Selector:

Table 4.4: Count of beat types in data set returned by Feature Selector

Beat Type (uint8)	Beat Type [38]	Count
0	*False Positive*	1070
33	!	250
47	/	7021
65	A	2542
69	E	104
70	F	784
75	J	83
76	L	8062
78	N	74520
81	Q	21
82	R	7214
83	S	2
86	V	6852
97	a	69
101	e	16
102	f	969
106	j	223
124	|	42
Sum		109844

As is obvious from this table the normal beat (N) type is severely overrepresented. If this data set is used as the training set directly the amplitude of adaption to N beats will be much greater than other beats. However, rare occurrences of non-normal beats tend to be more clinically significant [8] and useful for further diagnosis.

Recall that per beat fed to the network an array of properties of that beat is used. Inside of each group of beats with identical Beat Type there is still significant spread in properties present. To stimulate the SNN with an as complete input set as possible it is essential to not only highlight rare beats but also determine an orthogonal set of each beat type.

Although the author is aware of these problems this subject was not investigated further due to time constraints.

4.5.2 Neuron Model

We want our SNN to be able to approximate any continuous function from input to output as a result of learning. A mathematical theorem called the Universal Approximation Theorem deals with this subject for simple multilayer feed forward networks. Summarizing, it states that for this type of network to be a universal approximator of continuous functions either of the following criteria should be met:

1. The activation function of hidden layer nodes of the network should be bounded, nonlinear and monotonously increasing [56]. No rules apply to the amount of nodes needed.

2. The amount of available nodes in hidden layers is unbounded with some general requirements on activation function of these nodes [57]. Result quality is very dependent on node activation function.

In this thesis only the first solution is considered while the second one is considered to be purely of theoretical use. The activation function of our neurons should therefore be a bounded, nonlinear and monotonously increasing function.

We will take a closer look at the relation between input current and output current (in the form of discrete charge packets per second or spike rate) for the two biological models available in this project. In Figure 4.5 and Figure 4.6 the typical transfer function of the Leaky Integrate & Fire and Izhikevich neurons respectively are presented.

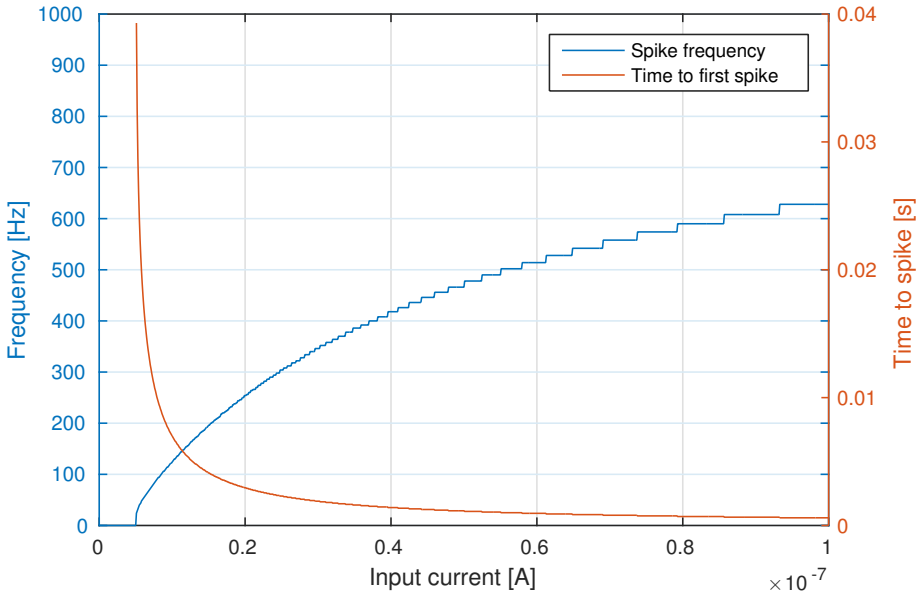

Figure 4.5: Typical input to output relationship for the Leaky Integrate & Fire neuron

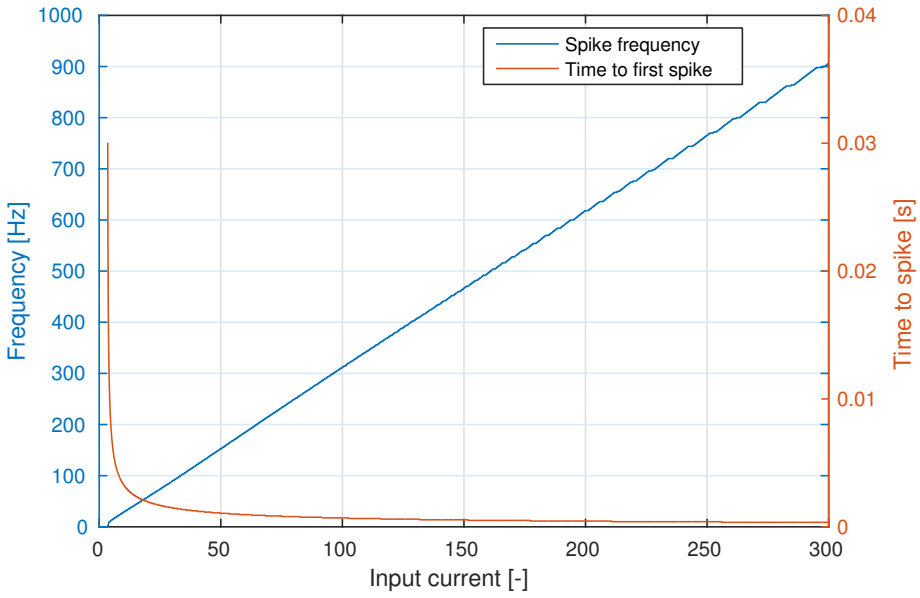

Figure 4.6: Typical input to output relationship for the Izhikevich neuron

As can be seen from the images above, the Leaky Integrate & Fire model presents the nonlinearity that we are looking for. Although the Izhikevich model is capable of producing nonlinear membrane voltages over time its input to output relation is not nonlinear. Therefore from here on, the neuron model that will be used is the Leaky Integrate & Fire model. Other advantages are its lower computational complexity and configuration complexity (see Table 2.1).

The variables to configure for this model are as follows: membrane capacitance C_{mem}, membrane resistance R_{mem}, refractory period T_{ref}, reset membrane voltage V_{mem0}, and threshold membrane voltage V_{thresh}.

4.5.2.1 Constraints & Considerations

Initially, the combination of R_{mem}, C_{mem}, and V_{thresh} is configured so that spiking activity is in the biological time scale, with possible spike frequencies up to 1 kHz [58]. The range for V_{mem} being V_{mem0} up to V_{thresh} was not biologically inspired and set to 0 mV up to 50 mV. Refractory period tuning is covered in Section 4.5.3.2.

From this starting point membrane charge leakage determined by $R_{mem}C_{mem}$ has two other constraints related to the spike acitivity schedule presented in Figure 4.10. First, accumulated charge during the active input phase (50 ms, see Figure 4.5.1.1) should not leak away too fast during each exposure as this prevents learning from the full spike pattern. Second, as this device is not made to learn from relationships between beats, accumulated network charges need to leak away before the next sample is presented.

This rate of forgetting information/charge determines how much history of each pattern is relevant. Note that this forgetting phenomenon also occurs between actually generated spikes in STDP which is covered in Section 4.5.3.3.

4.5.3 Synapses & Learning Rules

4.5.3.1 Weight Storage

In state of the art hardware implementations of SNNs either low resolution digital storage of synaptic *conductivity* or *weight* storage or continuous analog storage [33] is used. A common digital approach is storing weights as 2 or 3 bit signed integers. This allows for reduced area costs of synapse hardware which is a component that tends to scale with more than linearly (up to square) with neuron count. It also reduces the complexitiy of learning rule updating blocks since they can perform calculations in a lower precision.

Research tends to show that below ten weight levels spread across excitatory and inhibitory connections is enough. As a result in digital synapses limiting the precision to i.e. 3 bit two's complement is a common occurrence.

In this project the effects of changing digital synaptic weight storage formats and severely limiting usable weight levels were not investigated but a simulated analog synaptic weight storage system was implemented. All synaptic weight related computation and storage was performed both in exclusive 32bit and exclusive 64bit IEEE754 floating point formats [59] and no differences in SNN device behaviour were noted. Unless otherwise noted 64bit IEEE754 is used in this project.

4.5.3.2 Learning Rules

The simulator can be used to investigate both second order (STDP) and third order (TSTDP) spike timing based learning rules. Spike frequency based rules like BCM were not investigated further or implemented. Given that the STDP rule is much cheaper to implement (see Section 2.2.3) and easier to configure than the TSTDP one we should find out in which scenarios TSTDP provides an advantage instead of just using it because of its higher biological accuracy and increased functionality.

Our first area of interest is the spike activity frequency range in which both algorithms are usable. Recall that the experiments that the STDP rule were derived from used isolated single presynaptic and postsynaptic spikes [6]. The STDP rules do not define response to multiple recent spike events on either end of the synapse. That is where TSTDP comes in (see Section 2.2.3.2).

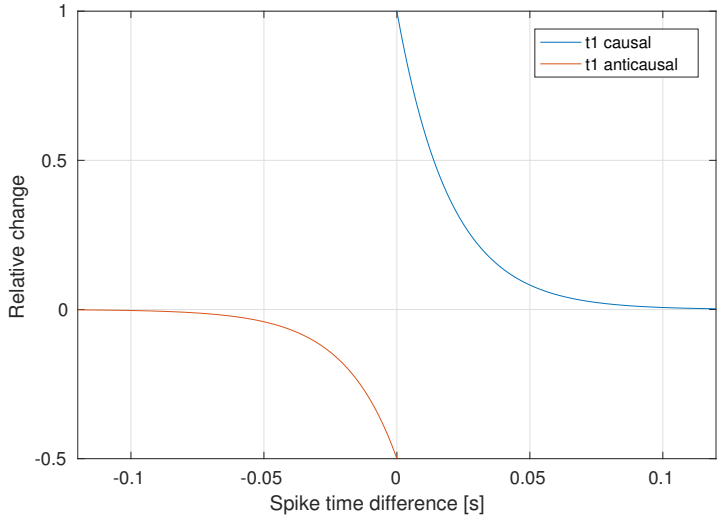

Figure 4.7: Typical second order (STDP) learning window showing conductivity change as a function of spike time difference. STDP configuration parameters are $\tau_+ = \tau_- = 20$ ms

In Figure 4.7 a typical STDP window is provided. Recall that this is a mathematical fitting of original data shown in Figure 2.7. Line $t1causal$ describes the potentiation window while line $t1anticausal$ describes the depression window. Because of the exponential relationship between spike time difference Δt and rate of change the range of significant change for i.e. 1% of maximum is ∼4.6 times the configured decay constant. Let us call this the *STDP window*. When there are multiple spike events found in the STDP window in both directions around any spike event simply applying the STDP rule to all possible pairs leads to unpredictable learning: the STDP model does not describe what would happen biologically. This leads to what can be called the pingpong effect.

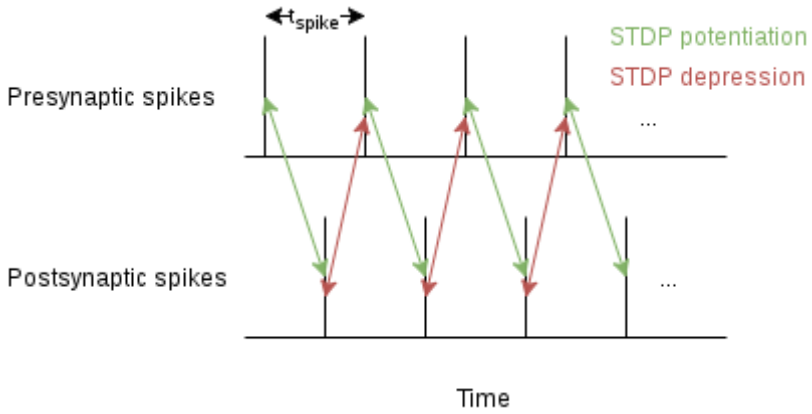

Figure 4.8: Diagram showing spike activity of two LIF neurons connected by an STDP synapse. Presynaptic neuron produces spikes at t_{spike} and postsynaptic neuron responds accordingly

In the scenario depicted in Figure 4.8 the spike timestep of the presynaptic neuron is t_{spike} in the range of the STDP time of effect. When blindly applying the STDP rule to all available pairs the net result can vary from no change at all to depression even though a causal relationship exists between each consecutive pair of spike events. This problem presents itself most clearly when investigating the conductivity of a synapse over time if a sawtooth pattern is observed there and steady learning in either direction is expected.

A frequency analysis based depiction of this effect is shown below in Figure 4.9.

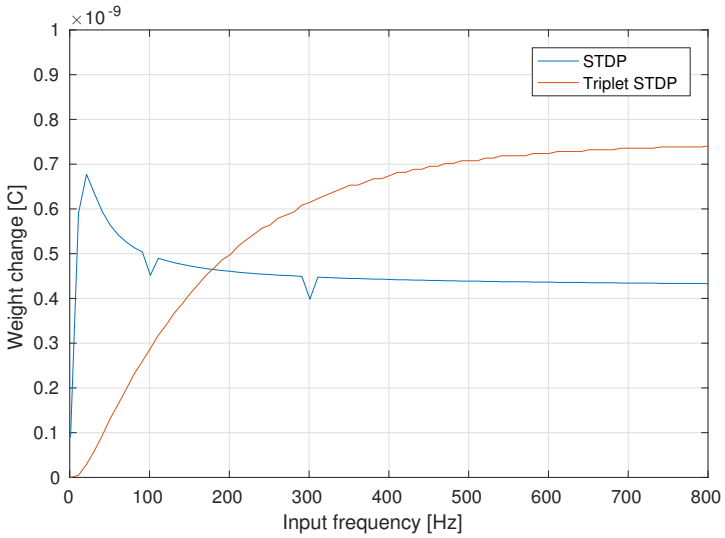

Figure 4.9: Comparison of conductivity change behaviour as a function of input frequency for STDP and TSTDP synapses.

In the above figure a similar simulation setup is used as in Figure 4.8: two LIF neurons are connected by either an STDP or TSTDP synapse. The presynaptic neuron fires at a timestep of t_{spike} for one second. The t_{spike} variable is varied across the biological time scale of spike frequencies [58]. The postsynaptic neuron is programmed to initially respond in a one-to-one fashion. In this simulation the STDP window parameters from the original paper are used [6]: $\tau_+ = 33.7$ ms, $\tau_- = 16.8$ ms. The problem of the pingpong effect is clearly evident in the blue line: as causal spike activity increases, learning performance only does so up to 21 Hz. After that STDP potentiation and depression changes equalise in amplitude. This is unexpected and wasteful behaviour. Please note that the amplitude of what happens after the peak depends on the relation between the STDP amplitudes A_{plus} and A_{min}.

Note that the unexpected dips in the STDP line are due to finite simulator timestep effects: one notable example is that if spikes are produced at a timestep of t_{spike} for one second the total amount of spikes $\frac{1}{t_{spike}}$ tends to jump at some increments of t_{spike} and will not increase for others. Another side effect of this test setup is that it is not always the same ping or pong that occurs last, thereby slightly altering the final conductivity change value.

4.5.3.3 Approach to using STDP

To avoid the uncertainty that using STDP results in when unknown frequency spike rates are present, one can design their SNN in such a way that high frequency spike trains are avoided or that deletion of STDP spike pairs from the logbook of neurons occurs. The first method is implemented in the final design.

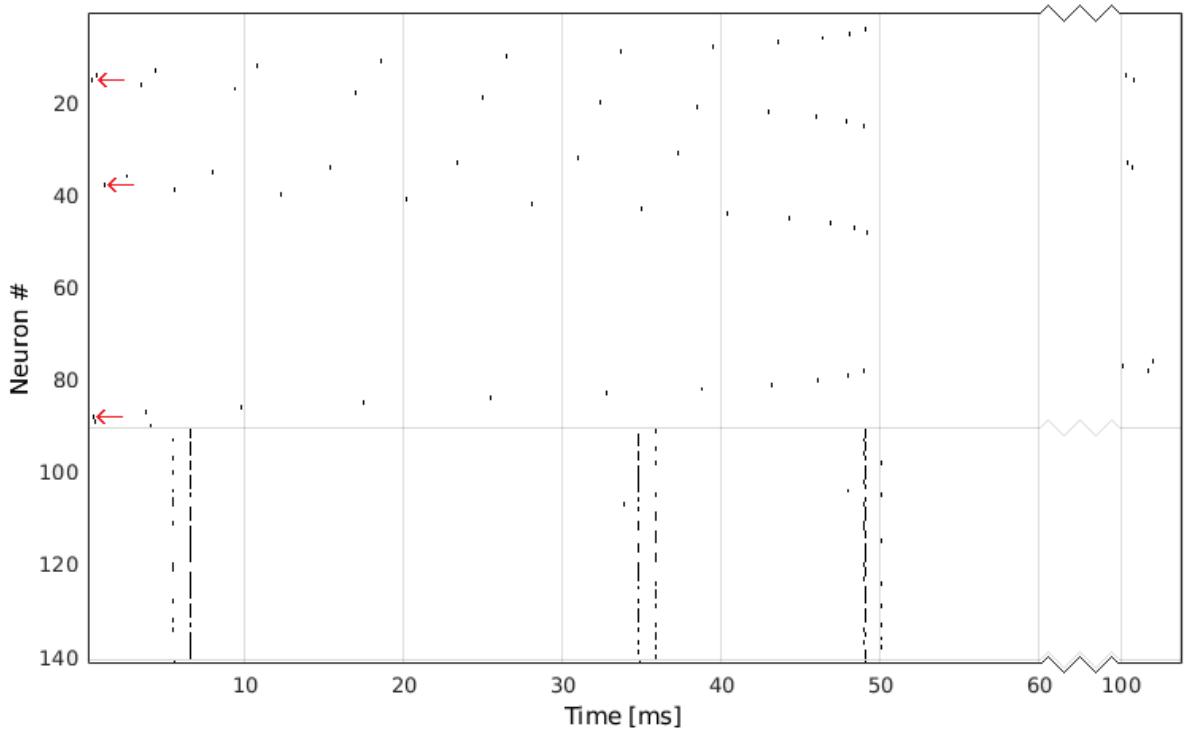

Figure 4.10: Realtime flow of spike timing repeated from Figure 3.15. Note the spacing between input and output layer spike time ranges. Red arrows highlight the neuron most sensitive to the input value encoded by each POE population.

The final device will use the spike timing schedule as can be seen in Figure 4.10. The following measures were taken to ensure that STDP provides enough functionality in the final device and TSTDP is not needed.

First, the two neuron populations (input layer U and output layer V) in the final network layout are configured in such a way that they cannot generate high frequency spike trains. The input layer is limited to up to (recall *activationminimum*) one spike per input signal sample. Since the input sample step is 100 ms the maximum spike frequency of that layer is only 10 Hz. The output layer responding to that is forced to only respond up to one time per input pattern by setting the refractory period t_{ref} of all its neurons to the active period of the POE (50 ms in the image).

Second, using knowledge of the boundaries of spike timing from the setup above we can tune the STDP window width to prevent unwanted significant anticausal plasticity. The significant portion of the STDP window needs to be narrower than 50 ms minus maximum output layer propagation delay so that any spike pairs emanating from sep-

arate beats do not lead to significant plasticity change. It is convenient to also take the STDP considerations mentioned in Section 4.5.2.1 into account now. The window width determines how much of the POE population shapes are taken into account during learning. One could say that input neurons spiking much later or much sooner are not relevant to the spike event caused in the output layer, and therefore learning should not occur there.

Combining these two considerations, initially a value of 8 ms for τ_+ and τ_- was chosen so that no inter-beat false learning occurs and that for spikes in the middle of the active period are significantly affected by the whole 50 ms input pattern.

4.5.3.4 When to upgrade to TSTDP

On the other hand if continuous learning is required with spike rates exceeding the pingpong onset point or if no strict learning regime preventing spike bursts exists one could opt for TSTDP. As explained in previous chapters, TSTDP accounts for two spike events in each direction of time and is able to fix the pingpong problem. TSTDP is also more biologically accurate describing what happens when higher spike rates occur. If the added cost of TSTDP is not an objection there is no real need to stick with STDP as TSTDP is a superset of STDP and can replicate all behaviour of it.

The TSTDP synapse is also an excellent candidate to be used to experiment with spike rate based systems. However, in this thesis the rate dependency is not of further importance.

4.5.4 Network Topology

For the network topology an enumeration of guidelines was used to obtain a baseline configuration:

- Initially, a 10 by 10 grid of neurons as the output layer is used, primarily because it provides adequate map formation space, and because it is the most often investigated grid size in literature. This is combined with an input layer population of 16 times 10, where each consecutive 10 neurons are connected to a POE block.
- Ensure that input layer to output layer initial conductivity distribution is both not too low and not too high. Too low initial weights inhibit further usage of these synapses during training, while too high values can early and permanent cause short-circuiting. For now, only the uniform distribution is used and investigated.
- Check if the sum of all input neuron to output grid neuron synaptic conductivities is proportional and not too much more than the spike charge required to generate a spike in the output layer. Recall that spiking is caused by an instantaneous charge of 50 pC (from $Q = C_{mem}V_{spike}$) arriving, where each spike arriving contributes an amount of charge dictated by its conductivity. If this balance is not present overactivation or short-circuiting across all of the output map can occur, especially when feedback conductivity is added.
- Make sure that when using feedback synaptic delay, the delay is larger than the refractory period of neurons. Otherwise feedback activity is ignored.
- Tune the mexican hat function so that at least one but not more than 2 units in a radius from each neuron are potentiated for a 10 by 10 grid. This tends to result in a optimal population of inter-potentiating neighbours of roughly 1-25 items, which is the upper limit of *BPR* testing (see Section 3.2.7.2).
- Make sure that for each grid neuron in the output layer the sum of potentiation (positive) and depression (negative) from the feedback layer is proportional to the same type of sum of connections from the input layer. If the strength of feedback synaptic layer is too low compared to the input synaptic layer short-circuiting can occur in the input synaptic layer. If input connectivity is lacking the circuit may lack enough spike activity required for learning.
- Ensure that the maximum allowable conductivity does not restrict synape

4.6 Obtained Baseline & Workflow

The analysis described in this chapter up to this point are what results in the baseline configuration of the device for the Input Encoder, SNN Simulator, and Output Decoder. Speaking in design space terms, this is a selection of points and ranges along the critical design space dimensions of the device. A summary is provided in Table 4.5. Recall that this is part two of the workflow described in Section 4.2 and will be summarized here.

For this baseline and further configurations two Figures of Merit (FOM) will be investigated: E_{MDS} and BCR. Both of these figures are explained in Section 3.2.7.1. To summarize, E_{MDS} defines the quality of the output mapping generated where lower is better, and BPR defines how many beats the SNN recognizes and places on the map.

Table 4.5: Baseline configuration as starting point for sweeps

Index	Configuration Name	Value	Notes
1.1	Input layer (U) Type	1D layer	See Section 3.2.8
1.2	Input layer feedback	None	See Section 3.2.8
1.3	Input layer size	160 neurons	See Section 4.4
1.4	Output layer (V) type	2D layer	See Section 3.2.8
1.5	Output layer feedback	Full XOR feedback	See Section 3.2.8
1.6	Output layer size	100 neurons	See Section 4.5.4
2.1	U to V initial distribution	Uniform	See Section 3.2.8
2.2	UV conductivity limit	20 pC	See Section 4.5.4
2.3	UV min. initial conductivity	3 pC	See Section 4.5.4
2.4	UV max. initial conductivity	5 pC	See Section 4.5.4
2.5	UV plasticity	Enabled	See Section 4.5.4
2.6	V to V initial distribution	Mexican hat	See Section 3.2.8
2.7	VV distribution tuning	"2dgridgauss 4, 3, 2.5"	See Section 3.2.8
2.8	VV conductivity scaling	10 pC	See Section 4.5.4
2.9	VV plasticity	Disabled	See Section 4.5.4
2.10	Conductivity generation seed	Fixed, configurable, 4	See Section 3.2.8
3.1	Neuron model	Leaky Integrate & Fire	See Section 4.5.2
3.2	Neuron capacitance	1 nF	See Section 4.5.2
3.3	Neuron resistivity	15 MOhm	See Section 4.5.2
3.4	Neuron threshold	50 mV	See Section 4.5.2
3.5	Neuron refractory period	50 ms	See Section 4.5.3.2
4.1	Input encoding scheme	POE	See Section 4.5.1
4.2	Input encoding layout	16 POE blocks of 10	See Section 4.4
4.3	POE receptive spacing	1	See Section 4.5.1.1
4.4	POE max offset	0.5	See Section 3.2.5.1
4.5	POE activation minimum	-1	See Section 4.5.1.1
5.1	Training dataset	Patient 100, all 2272 beats	See Section 4.5.1.2
5.2	Training dataset repeats	5	See Section 4.5.1.2
5.3	Testing dataset	Patient 100, first 100 beats	See Section 4.5.1.2
5.4	Testing dataset repeats	1	See Section 4.5.1.2
6.1	Plasticity algorithm	STDP	See Section 4.5.3.2
6.2	Plasticity implementation	TSTDP	See Section 4.5.3.2
6.3	STDP A_{2+}	50 fF	See Section 4.5.3.2
6.4	STDP A_{2-}	100 fF	See Section 4.5.3.2
6.5	STDP τ_{+}	8 ms	See Section 4.5.3.2
6.6	STDP τ_{-}	8 ms	See Section 4.5.3.2
6.7	Triplet properties	All A to 0, all τ to ∞	See Section 2.2.3.2

4.7 Critical Dimension Exploration

Starting from the discovered point or volume in the design space described in Section 4.6 we will conduct a more detailed analysis of critical configuration dimensions. From the starting volume the values of certain critical combinations of up to three dimensions/options will be *sweeped* in a sensible range and performance of each combination is measured by the multidimensional SOM scaling error E_{MDS} and BPR (see Section 3.2.7.1).

No detailed performance figures for the initial set point are given due to extreme FOM variability as a result of the initial conductivity generation. Although the initial weights can be set to a fixed value by fixing the random seed (option 2.10), resulting in a fixed FOM, changing this seed still throws the FOM all over the place. This effect of the initial conductivity generation will be used as the *noise* robustness test in the remainder of this chapter. For more information about this process see Section 3.2.6.

To reduce the amount of points required a divide-and-conquer method has been used, comparable to what was done previously by breaking up the device into a Feature Detector/Selector part and the remainder, see Section 4.2.

4.7.1 FOM Interpretation

In this section the E_{MDS} and BPR will be compared to the state of the art as well as reasonably possible. For comparison, any value for E_{MDS} more than 0.1 can be considered bad enough to be insignificant. Values below 0.02 can be considered competitive. For BPR, although a value of 100% would be preferred, values above 90% can be considered good and above 95% considered competitive.

4.7.2 Sweep Result Color Scheme

The colors in the figures that follow in this section combine the E_{MDS} and BPR values into one. E_{MDS} is used directly and has a range of [0 ... 1] by definition where 0 is a perfect score and 1 is the worst possible score. BPR was used in the range of [0 ... 1] where 1 means a perfect 100% BPR value. The final color is then computed as follows:

$$color = \frac{E_{MDS} + (1 - BPR)}{2} \tag{4.12}$$

The value $color$ is then mapped to a colormap that smoothly transitions between three points: pure green for $color = 0$ (best possible result), grey for $color = 0.3$ (average result), and pure red for $color = 1$ (worst possible result). A BPR of zero is treated as an invalid result and mapped to $color = 1$.

For example, an average E_{MDS} of 0.1 combined with a good BPR of 90% results in color = 0.1 which is a relatively bright green. However, an average E_{MDS} of 0.1 combined with a very bad BPR of 10% results in $color = 0.5$ which is a relatively bright red meaning a bad result even though E_{MDS} is good.

4.7.3 Input Encoder

The first block to be explored in this fashion is the Input Encoder. It was done first since it is the earliest block in the device flow of information (see Figure 3.1) and its changes should therefore have the biggest effects on all other blocks. Changes as a result of this section are summarized in Table 4.6.

Table 4.6: Summary of Input Encoder changes as a result of sweeps for this block

Option	Old Value	New Value
4.1	POE	same
4.2	16 POE blocks of 10	same
4.3	1	same (1.5 possible)
4.4	0.5	same
4.5	-1	0.025

To arrive at these results a total of 276 combinations were tested. From these, 156 (56.5%) are shown and discussed in this section. All sweep results, full configuration information for all sweeps, and analysis logs can be found in Appendix A.2.2.

4.7.3.1 POE Activation Minimum & Noise

First, the effect of POE activation minimum (option 4.5) was investigated. Knowing from baseline investigations that the initial conductivity values generated by the random generator seed (option 2.10) have a huge effect on FOM, we performed a two-dimensional sweep for 4.5 and 2.10 to average out its effect. The setup for this sweep is shown in Table 4.7.

Hypothesis: option 4.5 values slightly above zero reduce the amount of insignificant spike events. This should help the SNN with learning. Too high option 4.5 values reduce information density and should be avoided.

Table 4.7: Input Encoder sweep 1 setup information. Total combinations = 32

Option 4.5 Values (8x)
-1 *(current)*
0.000
0.025
...
0.250

Noise (4x)
1
2
3
4

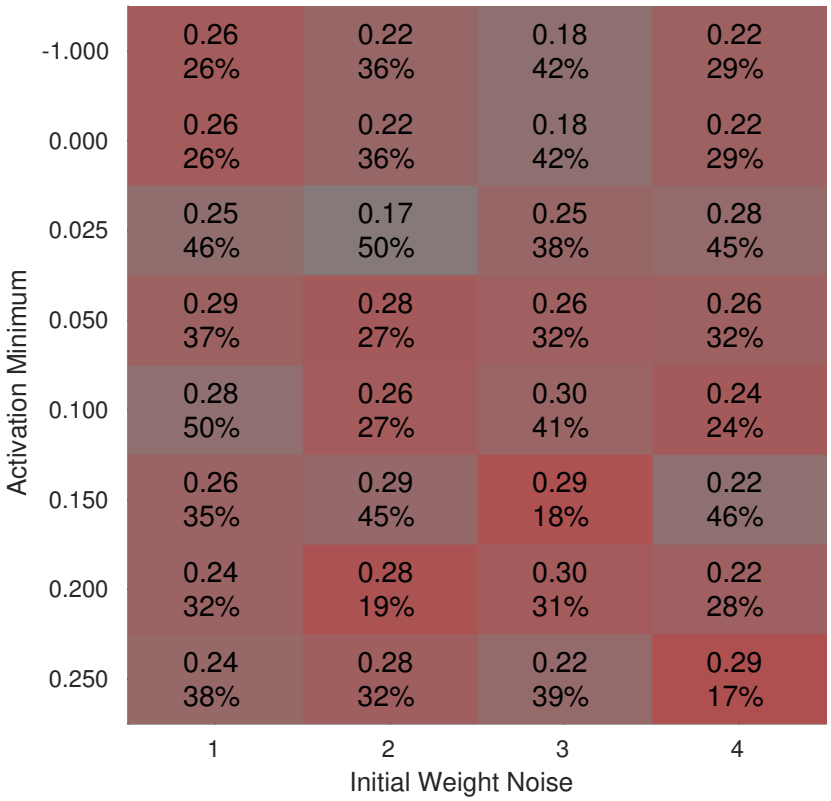

Figure 4.11: Sweep results for this combination. Along the vertical dimension is sweep variable 1 (option 4.5). Along the horizontal dimension is sweep variable 2 (option 2.10). Top text values show the E_{MDS} while bottom text values show the *BPR*. Color values are explained in Section 4.7.2.

The resulting E_{MDS} and BPR distribution is seen in Figure 4.11. Each colored box represents one combination of both sweep variables for a total of 32. Since for this sweep the horizontal dimension is the noise test, the mean and variance of each row is the most important piece of information of this matrix. The next-best data part to look at are individual columns with equivalent starting conductivity values.

From this exploration we can learn that in this whole subspace E_{MDS} is not very competitive with non-spiking solutions. However, although not shown in the figure, there is a clear improvement of the E_{MDS} through time as a result of the full device architecture meaning that the concepts of SNN based learning do work. E_{MDS} values obtained by testing without training (recall Section 3.2.1) tend to show values around 0.5.

Apart from the above global observations there is a slight increase in BPR and general *performance color* (see Section 4.7.2) for low values of option 4.5 above zero up to roughly 0.2.

Conclusion: increase option 4.5 slightly from -1 (effect not used) to 0.025. More investigation in this value range is needed.

4.7.3.2 POE Activation Minimum Zoom & Noise

Now we will zoom in on the best region of option 4.5: low values around 0.025.

Hypothesis: for this experiment results comparable to the last experiment are expected, but with a bigger higher performance region around 0.025.

Table 4.8: Input Encoder sweep 2 setup information. Total combinations = 48

Option 4.5 Values (12x)
-1
0.000
0.005
...
0.025 *(current)*
...
0.050

Noise (4x)
1
2
3
4

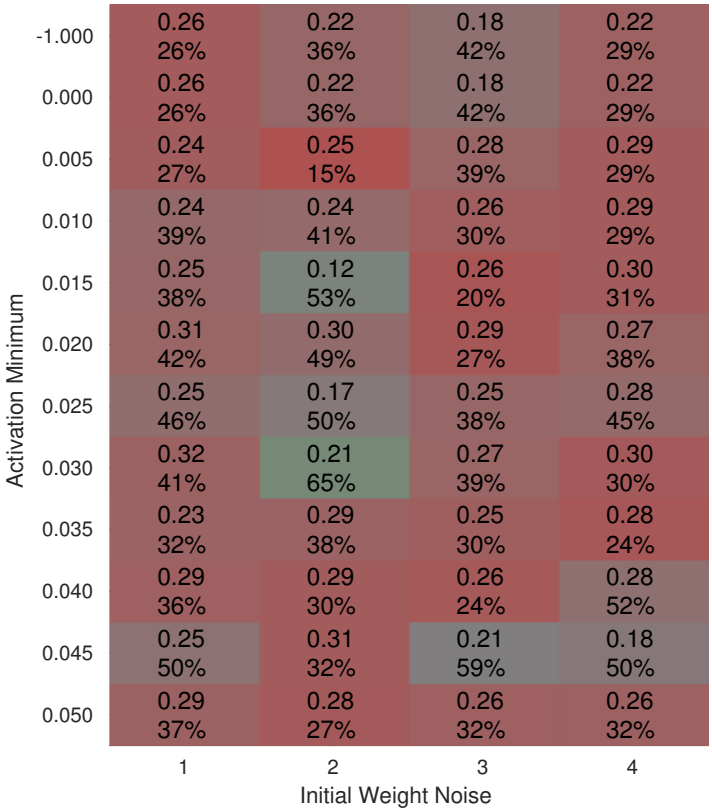

Figure 4.12: Results for the sweep of option 4.5 (down) versus option 2.10 (right).

The resulting FOM shown in Figure 4.12 slows a slight improvement in the region around 0.025 when looking at row averages.

Conclusion: there is still slight preference for the region around 0.025 but no improvement from that point. Option 4.5 will be kept at 0.025.

4.7.3.3 High POE Receptive Spacing & Noise

Now we will take a look at the high range of the receptive spacing option of the POE algorithm. This is option 4.3 in the baseline table. In the original publication this values was set to 1.5, and a usable range of [0.1 ... 10] is presented in Section 4.5.1.1.

Hypothesis: at the higher limit of the presented range performance should decrease due to the earlier mentioned effects. Values around 1 or 1.5 should perform comparably.

Table 4.9: Input Encoder sweep 3 setup information. Total combinations = 44

Option 4.3 Values (11x)
1 *(current)*
1.5
2
...
10

Noise (4x)
1
...
4

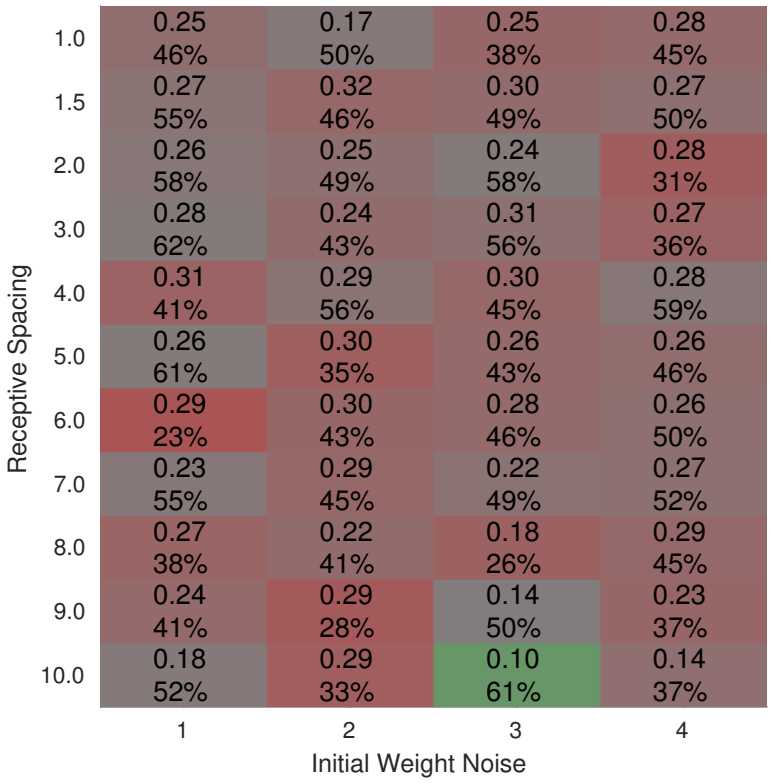

Figure 4.13: Results for the sweep of option 4.3 (down) versus option 2.10 (right).

As seen in Figure 4.13 there is a slight improvement in E_{MDS} and sometimes even *BPR* for extremely high values (9 and 10) of option 4.3 at the cost of lowered *BPR*. In the lower receptive spacing value range no significant effects were observed.

Conclusion: extremely high receptive spacing values like 9 and 10 perform slightly better than the order of magnitude below them, albeit only for some noise cases. How-

ever, no major noise independent effects are observed across this range at this step resolution.

4.7.3.4 Low POE Receptive Spacing & Noise

Next we will take a look at the low range of the receptive spacing of the POE algorithm. Recall that a usable range of [0.1 ... 10] is presented in Section 4.5.1.1. No results are given for the very lowest values (below 0.4) since mapping performance completely breaks down there.

Hypothesis: at the lower limit of the presented range performance should decrease due to the earlier mentioned effects. An expected high point of option 4.3 should be around 1 to 1.5.

Table 4.10: Input Encoder sweep 4 setup information. Total combinations = 32

Option 4.3 Values (8x)
0.4
0.7
1 *(current)*
...
1.5

Noise (4x)
1
...
4

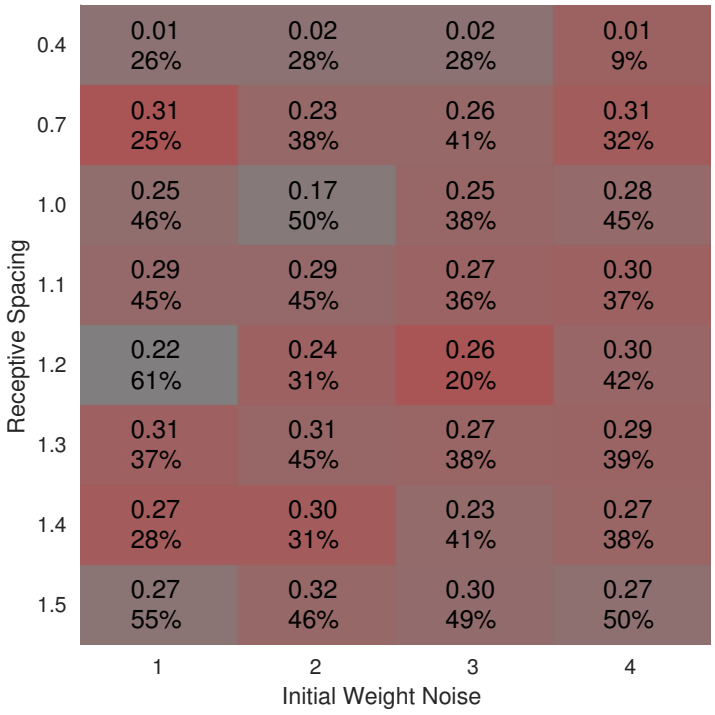

Figure 4.14: Results for the sweep of option 4.3 (down) versus option 2.10 (right).

For the top row in this region E_{MDS} improves to good values far below 0.1, but this is misleading as this excellent mapping performance only applies to a small selection of the training dataset.

Conclusion: at receptive spacing values below 1, especially 0.4, *BPR* drops. At other parts no conclusion could be drawn. Maybe a slight preference for values around

1 to 1.2 can be observed across the remainder of the sweep, as denoted by the block colors.

4.7.4 UV Connectivity

The next block of configuration options to investigate is the connectivity between the input layer and the output layer. As with the choice of the first block to investigate, this is the next device piece in the flow of information through it. Changes as a result of this section are summarized in Table 4.11.

Table 4.11: Summary of UV conductivity changes as a result of sweeps for this block

Option	Old Value	New Value
2.1	Uniform	same
2.2	20 pC	60 pC (up to 70 pC possible)
2.3	3 pC	same (2.5 pC possible)
2.4	5 pC	4.5 pC (up to 4.7 pC possible)

These results were obtained by simulating a total of 1003 combinations. From these, 553 (55.1%) are shown and discussed in this section. All sweep results, full configuration information for all sweeps, and analysis logs can be found in Appendix A.2.3.

4.7.4.1 Minimum & Maximum UV Initial Conductivity Limits & Noise

The first test subject is the range of the uniform distribution used to generate the U to V layer synaptic conductivity. In this experiment the lower (option 2.3) and upper (option 2.4) limit of this distribution are tested. For this layer the first attempt will be the lower sensible range of values where between 50 (1 pC per spike) or 5 (10 pC per spike) spikes are required to generate a spike at the other end of a synapse. Recall that in the simulator synaptic *conductivity* can be expressed in terms of the amount of charge per spike a synapse transports to the other end.

Hypothesis: for values where the upper limit is lower than the lower limit, the device should fail. For too high UV conductivity values short-circuiting is expected meaning that uncontrolled mass activity will occur in the V layer due to any kind of activity in the V layer. For a too low range a lack of V layer excitation is expected.

Table 4.12: UV conductivity sweep 1 setup information. Total combinations = 200

Option 2.3 Values (10x)
1 pC
...
3 pC *(current)*
...
10 pC

Option 2.4 Values (10x)
1 pC
...
5 pC *(current)*
...
10 pC

Noise (2x)
1
2

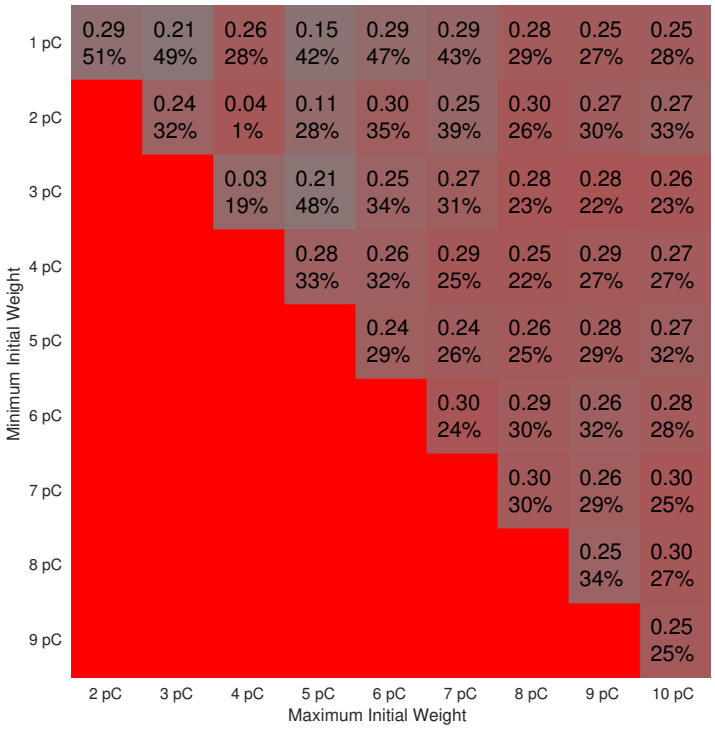

Figure 4.15: Results for the sweep of option 2.3 (down) versus option 2.4 (right). Block colors and text values are averages of all noise runs. Note that a pure red block without text means that any one of the noise runs failed to recognize any beats.

The results in Figure 4.14 show that there is a vague optimal range of initial conductivity around lower = [2 ... 3] and upper = [4 ... 5] when looking at E_{MDS}. However, as with the data Section 4.7.3.4, the E_{MDS} is only low because the amount of recognized beats (BPR) is low.

Conclusion: the simulation failure triangle is present as expected and for other extreme combinations of both options no good performance could be found. The only possibly interesting part is around the initial point found using qualitative methods.

4.7.4.2 Minimum & Maximum UV Initial Conductivity Limits Zoom & Noise

We will zoom into the good part of Section 4.7.4.1. Recall that a low E_{MDS} ideally below 0.1, and a high BPR, ideally above 90%, is considered a good result.

Hypothesis: a more detailed blob of high performance combinations is expected.

Table 4.13: UV conductivity sweep 2 setup information. Total combinations = 200

Option 2.3 Values (10x)
1 pC
1.333 pC
...
3 pC *(current)*
...
4 pC

Option 2.4 Values (10x)
3 pC
3.222 pC
...
5 pC *(current)*

Noise (2x)
1
2

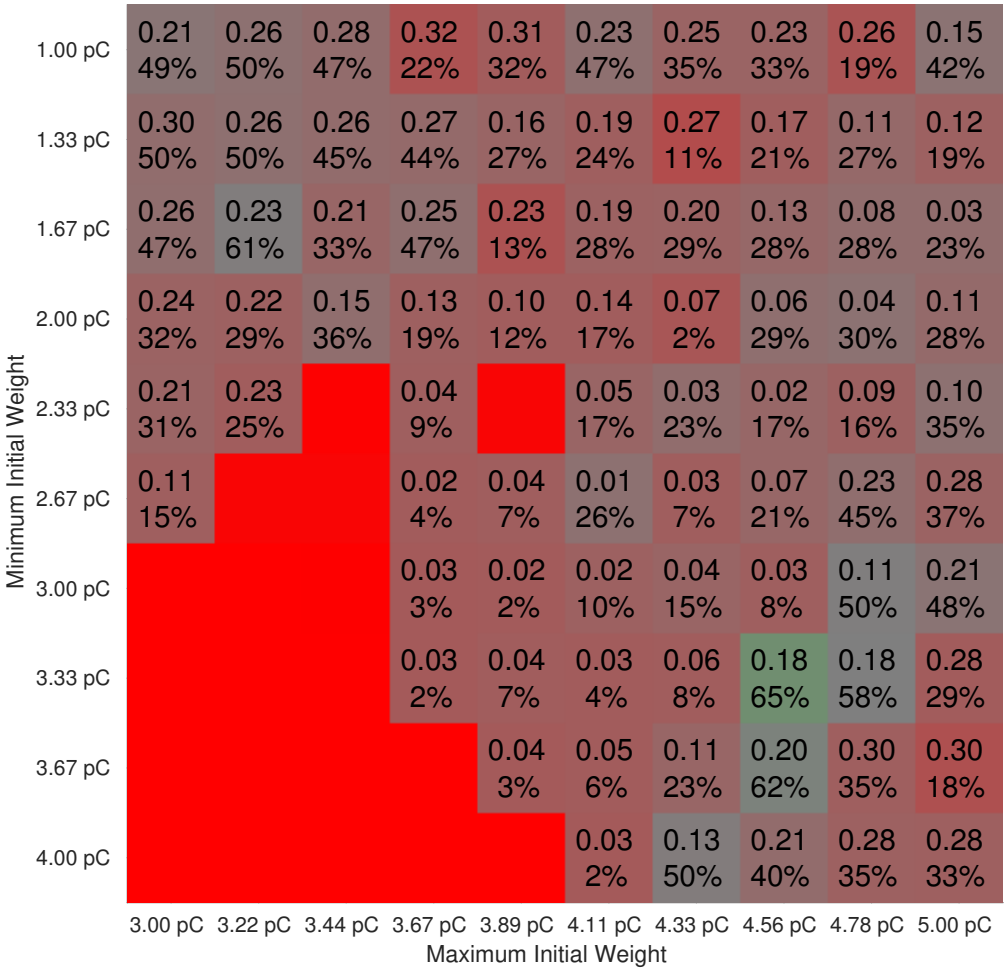

Figure 4.16: Results for the sweep of option 2.3 (down) versus option 2.4 (right). Results are averages of all noise runs. Again, note that a pure red block without text means that any one of the noise runs failed to recognise any beats.

Figure 4.16 shows that the optimal value blob is indeed more detailed. The greenest part with the best mapping performance unfortunately has a low *BPR*. Focus should be put on the area to the right of the green blob where E_{MDS} is only a bit worse at about 0.15 but *BPR* increases to up to 49.56%.

Conclusion: a more detailed small blob of good candidates is present. No ideal combination of very low E_{MDS} (< 0.1) and very high *BPR* ($> 90\%$) is found yet. Suggest reducing option 2.4 from 5 pC to 4.5 pC from now on.

4.7.4.3 Maximum Overall UV Conductivity Limits Zoom & Noise

The next topic of interest is the maximum possible conductivity a synapse in the UV layer can obtain through learning (option 2.2). This will be swept between 10 pC and 100 pC. From earlier experiments the lower value has been proven to severely inhibit the learning process, and that although synapses with conductivity more than 50 pC are equivalent when it comes to spike transport, a higher allowed limit during learning is sometimes needed to arrive at a better learning result.

Hypothesis: for maximum conductivity up to about 50 pC better learning should occur. Too high limits might cause short-circuiting if mass overlearning happens.

Table 4.14: UV conductivity sweep 3 setup information. Total combinations = 153

Option 2.2 Values (51x)
10 pC
...
20 pC *(current)*
...
100 pC

Noise (3x)
1
...
3

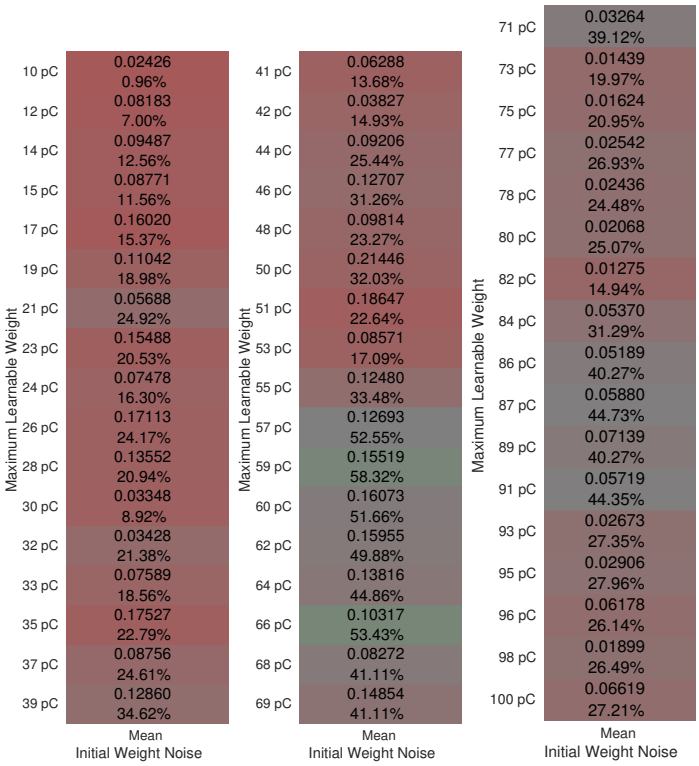

Figure 4.17: Results for the sweep of option 2.2 (down). Results are averages for all noise runs.

From Figure 4.17 it is clear that the initial conductivity was previously set to a too low value and that high maximum conductivity is advantageous for learning. For values far above (70 pC+) the one-to-one spike transport boundary of 50 pC *BPR* tends to drop down a bit.

Conclusion: as expected more available conductivity range set by option 2.2 allows for better learning. Too high allowed maximum conductivity tends to cause short-circuiting. Suggest increasing maximum conductivity from 20 pC to at least 60 pC. A value of 60 pC is chosen from now on.

4.7.5 VV Connectivity

For the next sweep block the shape and magnitude of the feedback connectivity will be modified and its effects investigated. Again, this is the next device piece in the flow of information through it. Changes as a result of this section are summarized in Table 4.15.

Table 4.15: Summary of UV conductivity changes as a result of sweeps for this block

Option	Old Value	New Value
2.6	Mexican hat	same
2.7	"4, 3, 2.5"	same
2.8	10 pC	14 pC

To arrive at these results a total of 721 combinations were tested. From these, 453 (62.8%) are shown and discussed in this section. All sweep results, full configuration information for all sweeps, and analysis logs can be found in Appendix A.2.4.

4.7.5.1 VV Layer Mexican Hat Feedback Shape & Noise

As part of block 3, the shape of the Mexican Hat pattern (option 2.7), but not the magnitude (option 2.8), will be investigated first. Recall from Section 3.2.8 that this pattern is tuned by parameters a, b, and r. Values below 2 of any of the three tuning variables have shown dissapointing results during the qualitative phase and are not included in the sweeps.

Hypothesis: for small variations in Mexican Hat pattern generation huge differences in device performance should occur. Current set point should be nearly optimal.

Table 4.16: VV conductivity sweep 1 setup information. Total combinations = 189

Option 2.7 Values (27x)
`2dgridgauss 2.0, 2.0, 2.0` *(a, b, r)*
...
`2dgridgauss 4.0, 3.0, 2.5` *(current)*
...
`2dgridgauss 4.0, 4.0, 3.0`

Noise (7x)
1
...
7

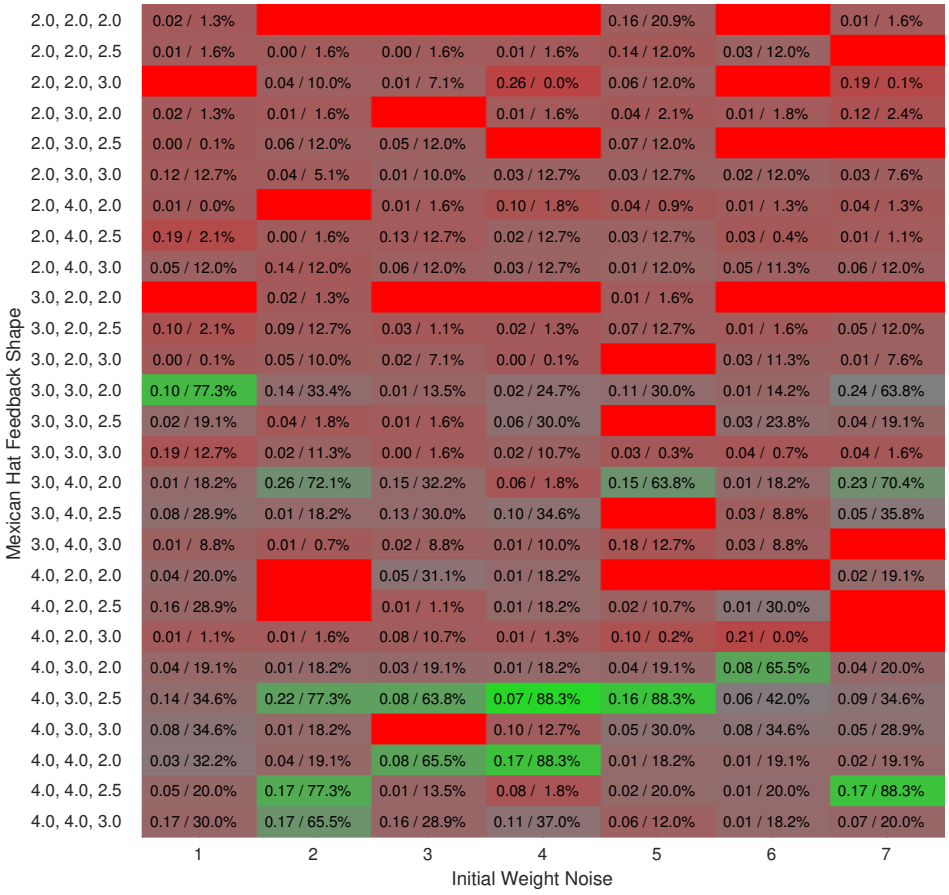

Figure 4.18: Results for the sweep of option 2.7 (down) versus option 2.10 (right). Values are shown side by side for improved readability.

As seen in Figure 4.18 this is a complicated subject that requires closer inspection. No linear relationships are present. Closer inspection reveals that the worst performers are anything with $a = 2$, anything with $b = 2$, or r below 2. For higher values of a and b, good candidates are `3,4,2`, `4,3,2.5`, and maybe `4,3,2.3`.

Conclusion: as expected option 2.7 is an extremely critical dimension to explore because high performance variation occurs due to small changes. The initial baseline of `4,3,2.5` is already a good candidate when limited to 0.5 increments in tuning variables.

4.7.5.2 VV Layer Mexican Hat Feedback Shape Zoom & Noise

Next the ideal point of feedback shape `4,3,2.5` is investigated more closely by zooming in.

Hypothesis: a more stable region than in the previous sweep should be found around `4,3,2.5` when reducing r step size from 0.5 to 0.1.

Table 4.17: VV conductivity sweep 2 setup information. Total combinations = 104

Option 2.7 Values (13x)
`2dgridgauss 4.0, 3.0, 1.9` *(a, b, r)*
...
`2dgridgauss 4.0, 3.0, 2.5` *(current)*
...
`2dgridgauss 4.0, 3.0, 3.1`

Noise (8x)
1
...
8

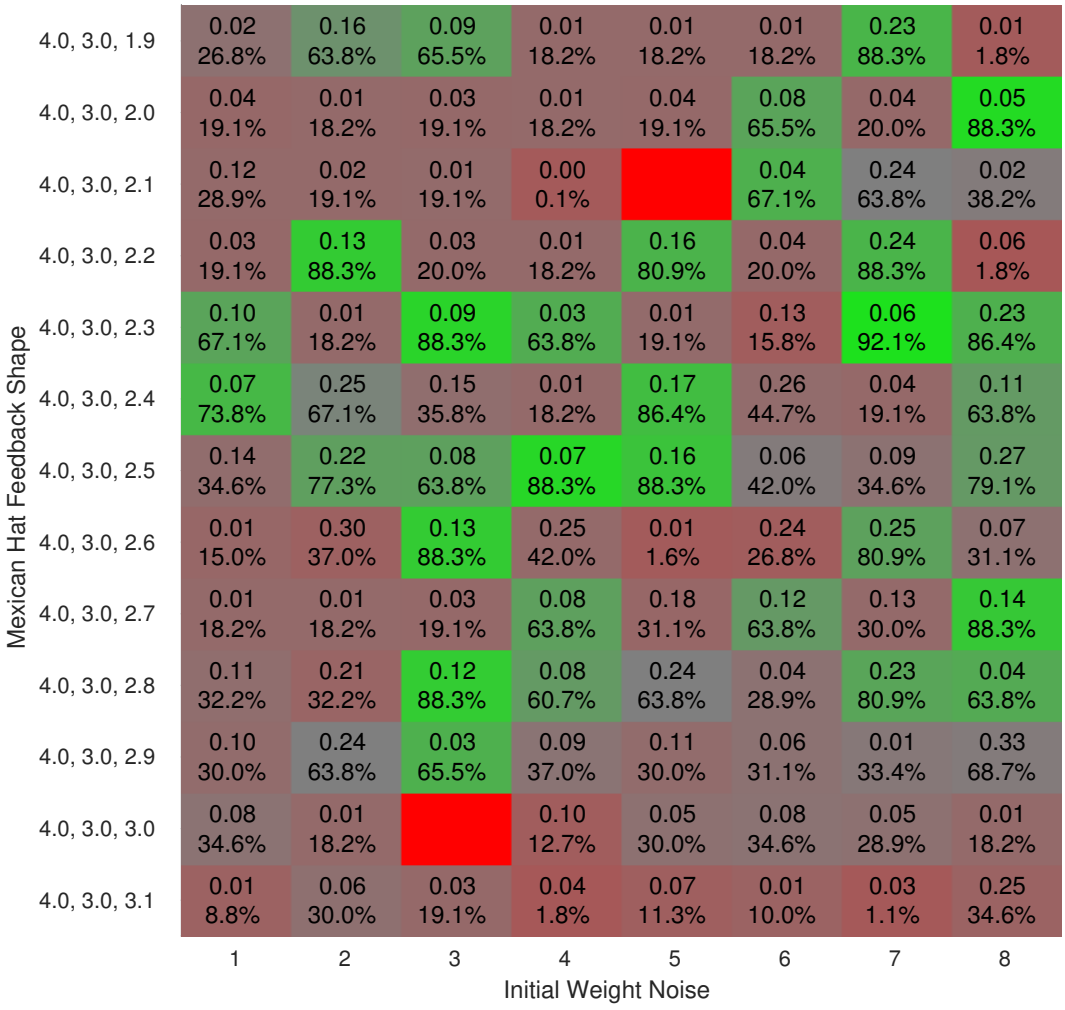

Figure 4.19: Results for the sweep of option 2.7 (down) versus option 2.10 (right).

From Figure 4.19 it is clear that zooming does provide a more stable region of performance. Observing a trend requires some serious squinting. Also there are still rare cases where performance completely breaks down due to a change in initial conductivity generation or miniscule changes in feedback connectivity (red blocks).

Conclusion: a more stable region was found around `4,3,2.5`, but looking at the combination of low E_{MDS} and high BPR no improvement is observed by changing r from 2.5. Shape will stay at `4,3,2.5`.

4.7.5.3 VV Layer Mexican Hat Feedback Amplitude & Noise

With the feedback *shape* fixed at `4,3,2.5`, we will focus our attention to the scaling factor applied to the shape obtained (option 2.8). During initial conductivity generation, this value is used directly as a multiplier of the normalized Mexican Hat shape generated. The resulting total UV conductivity magnitude should ideally be tuned in relation to total VV conductivity that we are changing now.

Hypothesis: it is expected that this parameter, like previous VV parameters, has a large effect on performance since the VV layer is a defining factor in SOM map formation.

Table 4.18: VV conductivity sweep 3 setup information. Total combinations = 160

Option 2.8 Values (20x)
1 pC
...
10 pC *(current)*
...
20 pC

Noise (8x)
1
...
8

Figure 4.20: Results for the sweep of option 2.8 (down) versus option 2.10 (right).

In Figure 4.20 a clear trend appears, showing that a correct relationship between UV conductivity magnitude and VV conductivity magnitude is of great importance. Detailed simulations of cases in the lower and upper show that there is widespread short-circuiting and open-circuiting in the UV layer respecitively.

Conclusion: the feedback amplitude has a large effect on device performance. The initial value of 10 pC was adequate but is close to the fall-of area where performance breaks down. A safer higher value of 14 pC was chosen instead.

4.7.6 STDP Window

The fourth and final block options to explore is related to the STDP window. Changes in this block relate to the rate of learning and the balance between potentiation and depression. Changes as a result of this section are summarized in Table 4.19.

Table 4.19: Summary of STDP changes as a result of sweeps for this block

Option	Old Value	New Value
5.3	Patient 100, first 100 beats	Patient 100, first 500 beats
6.3	50 fF	40 fF
6.4	100 fF	78.9 fF or 793 fF
6.5	8 ms	same
6.6	8 ms	same

To arrive at these results a total of 4732 combinations were tested. From these, 2633 (55.6%) are shown and discussed in this section. All sweep results, full configuration info for all sweeps, and analysis logs can be found in Appendix A.2.5.

4.7.6.1 STDP Window Amplitude & Ratio

The first two options to check are the positive and negative magnitudes of the STDP window. By checking a range of values for both of these, the relationship between potentiation amplitude and depression amplitude is also tested.

Hypothesis: from earlier testing a higher depression than potentiation factor is expected to work best.

Table 4.20: STDP sweep 1 setup information. Total combinations = 147

Option 6.3 Values (7x)	**Option 6.4 Values (7x)**	**Noise (3x)**
1 fF	1 fF	1
... (logarithmic)	*... (logarithmic)*	...
10 000 fF	10 000 fF	3

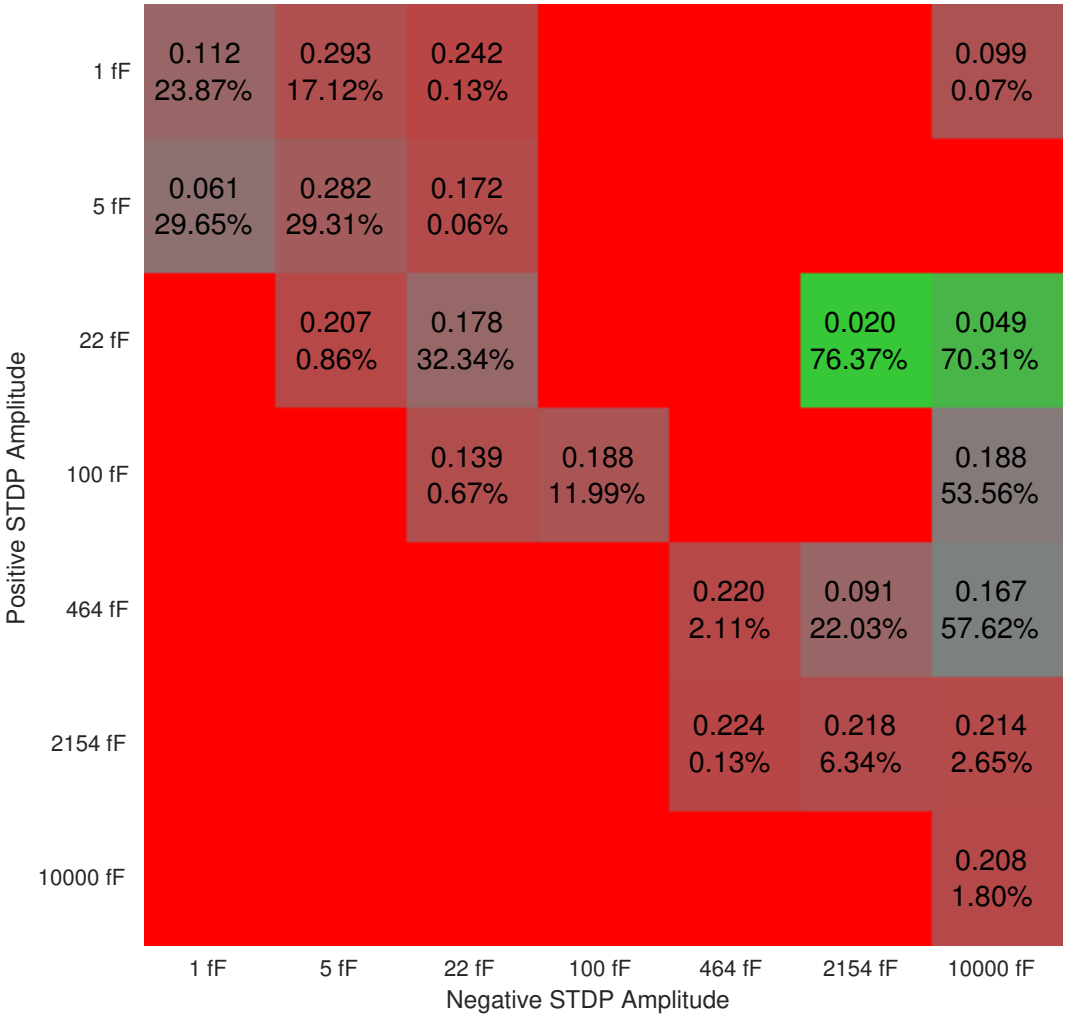

Figure 4.21: Results for the sweep of option 6.3 (down) versus option 6.4 (right). Note that a pure red block without text means that any one of the noise runs failed to recognize any beats.

In Figure 4.21 two things can be observed. First, the general failure of the device when potentiation is stronger than depression. Second, one very specific region of interest appears: the one where depression is between 100x and 500x stronger than potentiation, and only when the potentiation amplitude is equal to 22 fF (at this resolution).

Conclusion: the hypothesis was true, but a much smaller region of good performance than expected was found.

4.7.6.2 STDP Window Amplitude Zoom 1 & Ratio

Next, we attempt to zoom into the green area found in the previous experiment by removing one order of magnitude from the vertical dimension and increasing the amount of steps from 7 to 9.

Hypothesis: a bigger green blob is expected. Also the device is only expected to work when depression is stronger than potentiation.

Table 4.21: STDP sweep 2 setup information. Total combinations = 243

Option 6.3 Values (9x)	Option 6.4 Values (9x)	Noise (3x)
1 fF	1 fF	1
... *(logarithmic)*	... *(logarithmic)*	...
1000 fF	10 000 fF	3

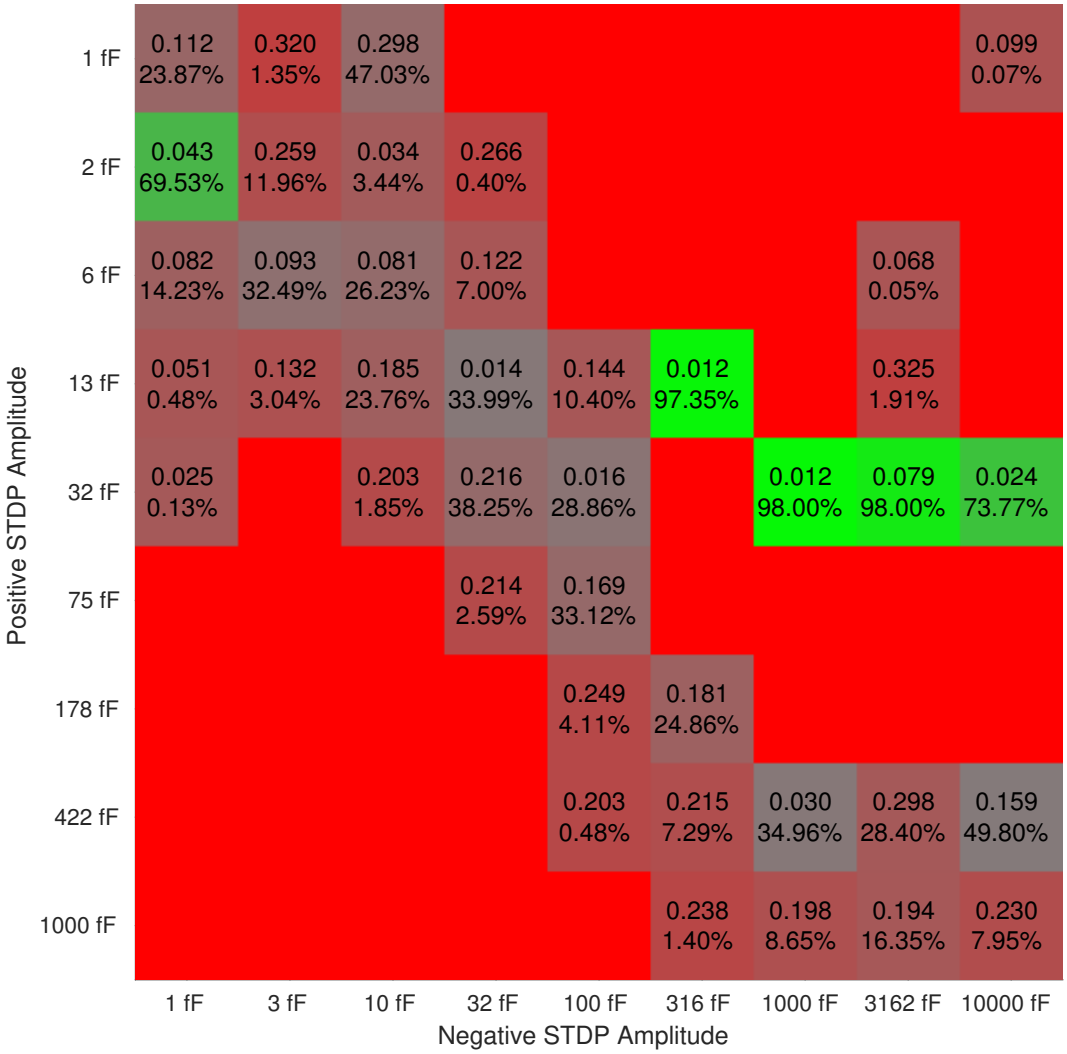

Figure 4.22: Results for the sweep of option 6.3 (down) versus option 6.4 (right).

From Figure 4.22 it is clear that tuning the device for the green area to the right will be very difficult as even the smallest step at this resolution means the difference between an almost perfect score and nothing at all.

One green data point also appeared, this time with much more mild learning rates. It is at 2 fF over 1 fF.

Conclusion: a bigger green blob is found. Also the device still one works when depression is stronger than potentiation.

4.7.6.3 STDP Window Amplitude Zoom 2 & Ratio

Again, we attempt to zoom more into the green area found in the previous experiment by removing one order of magnitude from the vertical dimension.

Hypothesis: an more detailed green blob is expected in the region found in the previous experiment.

Table 4.22: STDP sweep 3 setup information. Total combinations = 243

Option 6.3 Values (9x)	Option 6.4 Values (9x)	Noise (3x)
10 fF	1 fF	1
... *(logarithmic)*	... *(logarithmic)*	...
100 fF	10 000 fF	3

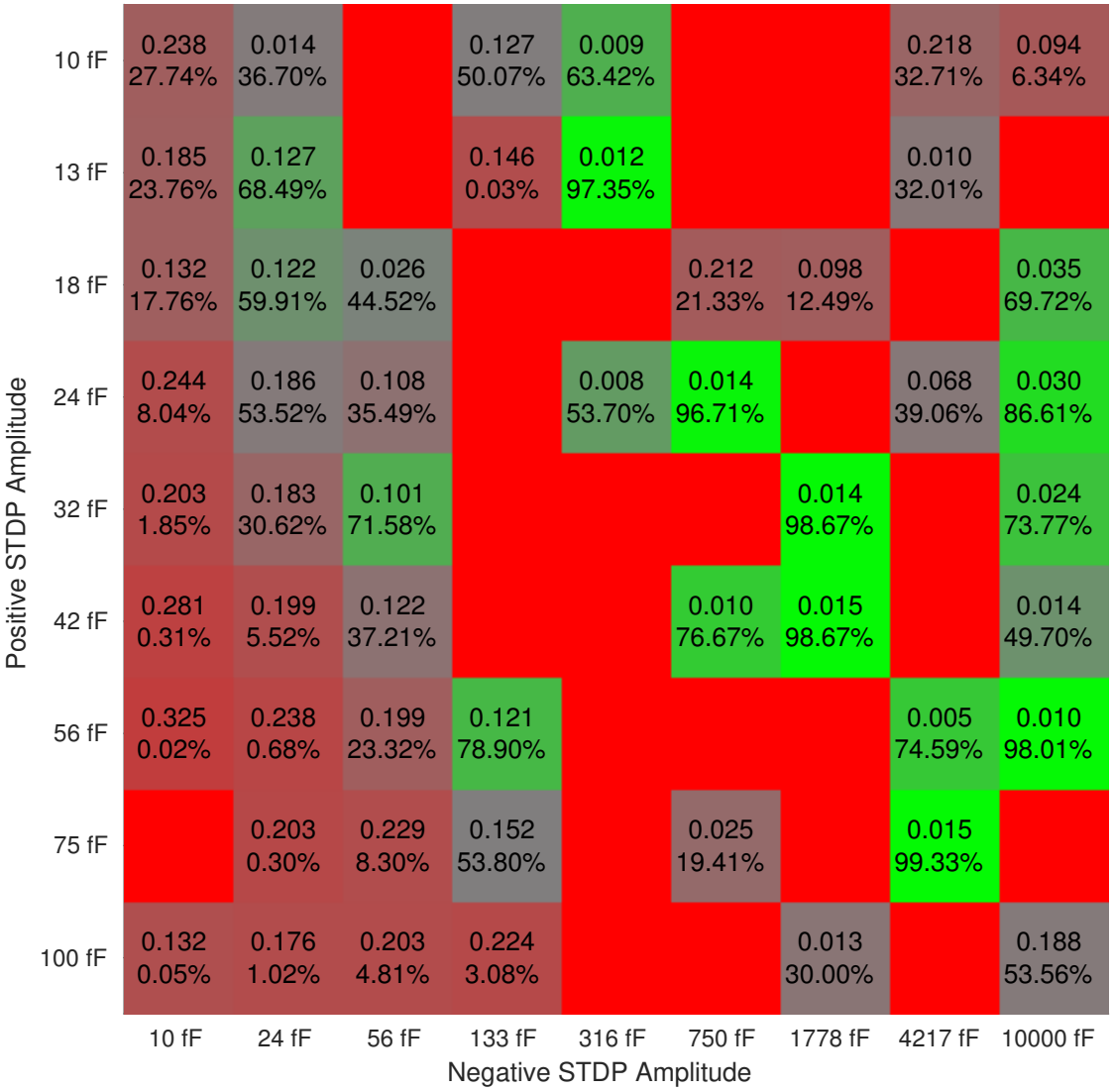

Figure 4.23: Results for the sweep of option 6.3 (down) versus option 6.4 (right).

When zooming in the blob shape of the image is becoming more difficult to observe. It might require moving your chair back a couple of meters. As was expected from the previous sweep, the device is very sensitive to minute changes in depression amplitudes. Repeatedly zooming in does not seem to hide that problem in any way.

At the higher depression amplitudes in the right half of the image very rapid removal of unnecessary branches in the UV connectivity occurs. For the extreme values in the order of magnitude of 10 000 fF it is not uncommon for any synaptic conductivity amplitude to decrease to zero instantly by a single close-timed anticausal spike pair.

Conclusion: an even bigger but non-continuous blob was found. It is expected that the learning rate in this area tends to cause overshooting of its optimal learning result more than it should.

4.7.6.4 STDP Window Amplitude Zoom 2 More Data & Ratio

Instead of zooming in even further the performance of the device with a more complex testing database is investigated. The amount of beats from patient 100 in the testing datbase is increased from the first 100 to the first 500 (out of 2272).

Hypothesis: with a bigger testing dataset the amount of overshoot might decrease.

Table 4.23: STDP sweep 4 setup information. Total combinations = 300

Option 6.3 Values (10x)
10 fF
... *(linear)*
100 fF

Option 6.4 Values (10x)
10 fF
... *(logarithmic)*
10 000 fF

Noise (3x)
1
...
3

Figure 4.24: Results for the sweep of option 6.3 (down) versus option 6.4 (right).

With the larger testing database the amount of total failures (BPR equals zero) is drastically reduced in the blob region of interest. Now a two-way split appears between slow learning and average results to the left of the red part, and fast learning and excellent but varying results to the right of the red part.

Hypothesis: the bigger dataset has revealed a more complete picture of the blob. In fact, two blobs or areas are present now. We will investigate both.

4.7.6.5 STDP Window Amplitude High A_{2-} Zoom & Ratio

The first region of Figure 4.24 to be explored is the high depression amplitude region. The goal is to find an optimal combination of option 6.3 (A_{2+}) and option 6.4 (A_{2-}) in that region.

Hypothesis: an optimum is expected around 40 fF for 6.3 with currently unknown values of 6.4.

Table 4.24: STDP sweep 5 setup information. Total combinations = 900

Option 6.3 Values (10x)	Option 6.4 Values (30x)	Noise (3x)
10 fF	100 fF	1
... *(linear)*	... *(logarithmic)*	...
100 fF	10 000 fF	3

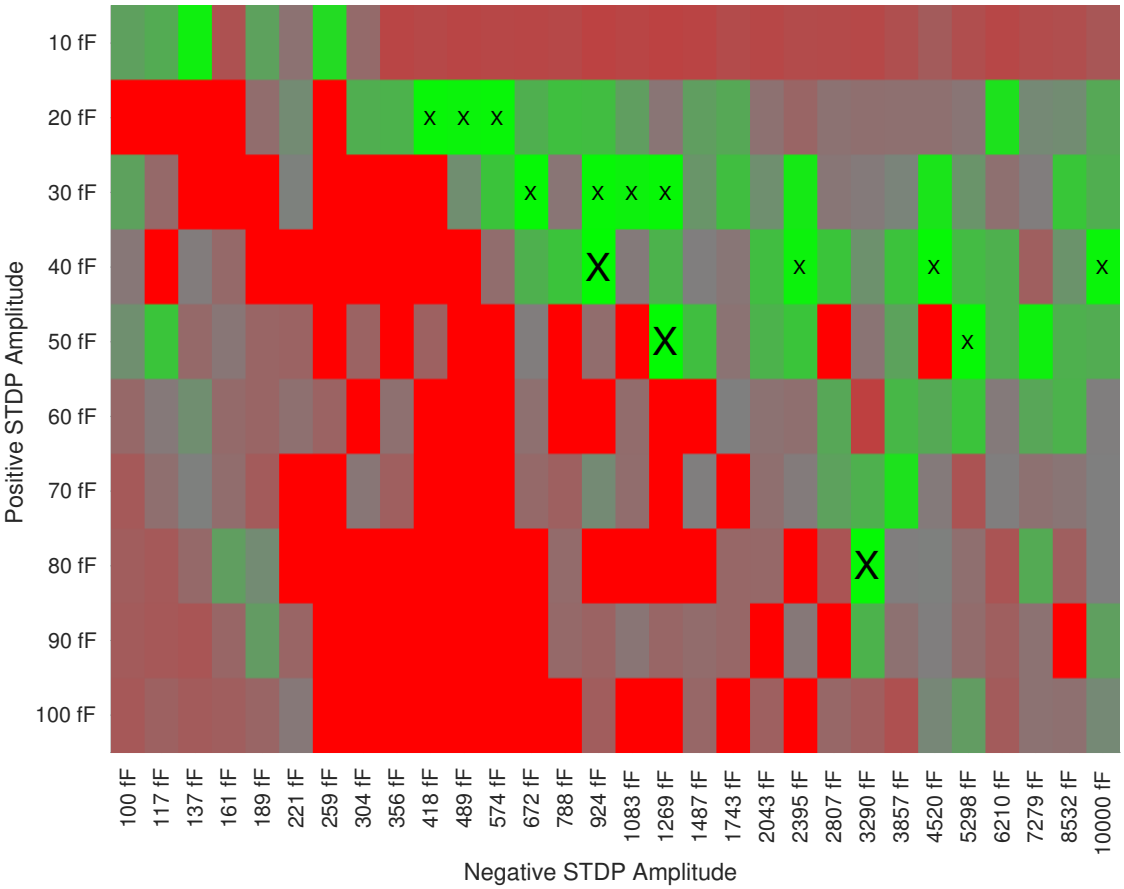

Figure 4.25: Results for the sweep of option 6.3 (down) versus option 6.4 (right). A *big* **X** has been placed at combinations where the E_{MDS} is lower than 0.05 and the *BPR* is higher than 99%. For combinations where E_{MDS} is also lower than 0.05 but *BPR* is between 95% and 99% a *small* **x** has been placed. Full result data can be found in Appendix A.2.5, line 676.

The green blob in the fast learning area behaves roughly the same in this zoom region as in did before. The center row of the green area tends to be at roughly 40 fF.

Conclusion: an optimum was indeed found at 40 fF for option 6.3. The range of values of option 6.4 that yield the most reliable learning are between roughly 400 fF and 1400 fF in the fast area.

4.7.6.6 STDP Window Amplitude Low A_{2-} Zoom & Ratio

Next, the low region was investigated more thoroughly, but this time the value of option 6.3 was limited to 40 fF decided on in Section 4.7.6.5.

Hypothesis: for the low depression amplitude range combined with 40 fF for option 6.3 reliable but worse performance than in the fast region is expected.

Table 4.25: STDP sweep 6 setup information. Total combinations = 400

Option 6.4 Values (100x)
40 fF
... *(linear)*
150 fF

Noise (4x)
1
...
4

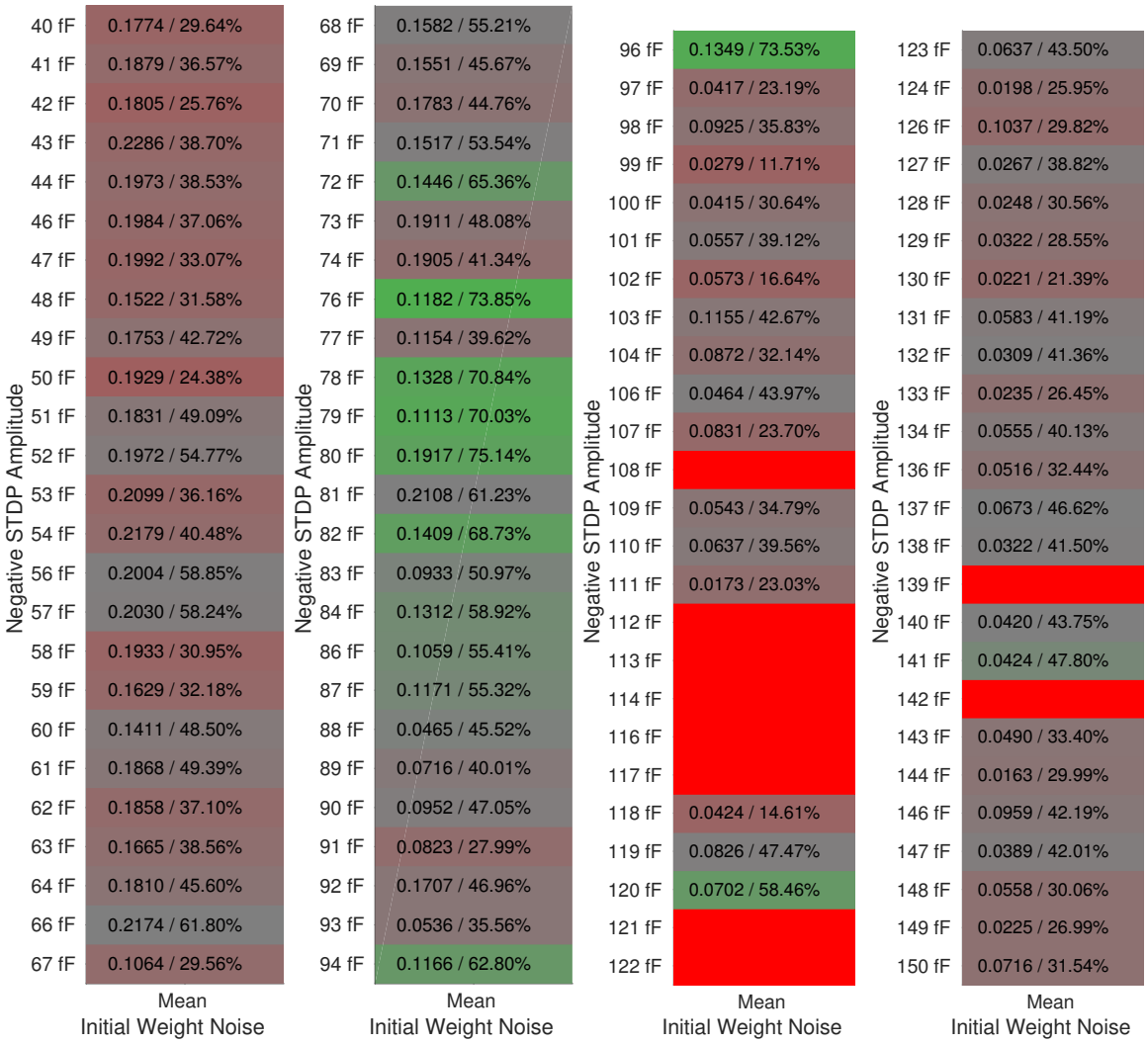

Figure 4.26: Results for the sweep of option 6.3 (down). Results are averages for all noise runs.

For the region of depression amplitude starting from the point where the amplitude is equal to the potentiation amplitude up to the red valley in Figure 4.24 where performance breaks down, an optimal region is found where the ratio of potentiation and depression is roughly $\frac{1}{2}$.

Conclusion: reliable but worse performance was in fact found in the aforementioned $\frac{1}{2}$ region. Peak performance across the four noise runs is shown in Table 4.26.

Table 4.26: Peak performance settings and FOM in the slow learning region

Setting	Value
Testing Dataset	Patient 100, first 500 beats
A_{2+}	40 fF
A_{2-}	78.9 fF

Figure of Merit	Value
E_{MDS}	0.1113
BPR	70.03%

4.7.6.7 STDP Window Amplitude High A_{2-} Zoom & Ratio

Similar to Section 4.7.6.6, the high region combined with 40 fF for option 6.3 was investigated.

Hypothesis: for the high depression amplitude range combined with 40 fF for option 6.3 unreliable but better performance than the low region is expected.

Table 4.27: STDP sweep 7 setup information. Total combinations = 400

Option 6.4 Values (100x)
500 fF
... *(linear)*
1000 fF

Noise (4x)
1
...
4

Figure 4.27: Results for the sweep of option 6.4 (down). Results are averages for all noise runs.

For the region of depression amplitude where depression is between 10 times and 25 times stronger than potentiation narrow bands of excellent performance are observed. These regions are however mixed with configurations where performance completely breaks down, even causing an all or nothing scenario in *BPR* between 793 fF and 794 fF for option 6.4.

Conclusion: extremely unreliable but excellent performance was found in the investigated region. For option 6.4 equal to 793 fF, across an average of four noise runs, all 100% of beats were recognized and mapped to the output map with an error of just 0.0169. However, an increase of just 0.13% in this configuration option causes beat recognition to drop to zero. Peak performance across the four noise runs is shown in Table 4.28.

Table 4.28: Peak performance settings and FOM in the fast learning region

Setting	Value
Testing Dataset	Patient 100, first 500 beats
A_{2+}	40 fF
A_{2-}	793 fF

Figure of Merit	Value
E_{MDS}	0.0169
BPR	100%

4.7.6.8 STDP Window Widths

The effect of modifying STDP window widths was also investigated. These experiments can be found at the end of Appendix A.2.5 and can be summarized as follows: as long as both τ_+ and τ_- are within 1 ms of each other and both are between 5 ms and 30 ms no significant difference in learning performance is found. These experiments were performed in the slow but reliable region of A_{2-}, with A_{2-} equal to 100 fF instead of 78.9 fF.

4.7.7 Training Regime

For this final section of exploratory experiments all baseline modifications up to now will be collected and applied to the device. Afterwards, experiments on bigger datasets and training set epoch counts will be performed. Full results, configuration data, and experiments with even larger datasets can be found in Appendix A.2.6.

4.7.7.1 Training Dataset Repeat Count With Slow Learning

For this experiment the amount of repetitions of the training dataset (option 5.2) and its effects are investigated. Up until now this configuration parameter was set to 5 because it was observed that the rate of network conductivity change converges to zero around this point.

Hypothesis: it is known that reducing the repetition count below 5 leads to worse performance. Values above 5 should not affect results.

Table 4.29: Training sweep 1 setup information. Total combinations = 110

Option 6.4 Values (11x)
1
...
10

Noise (10x)
1
...
10

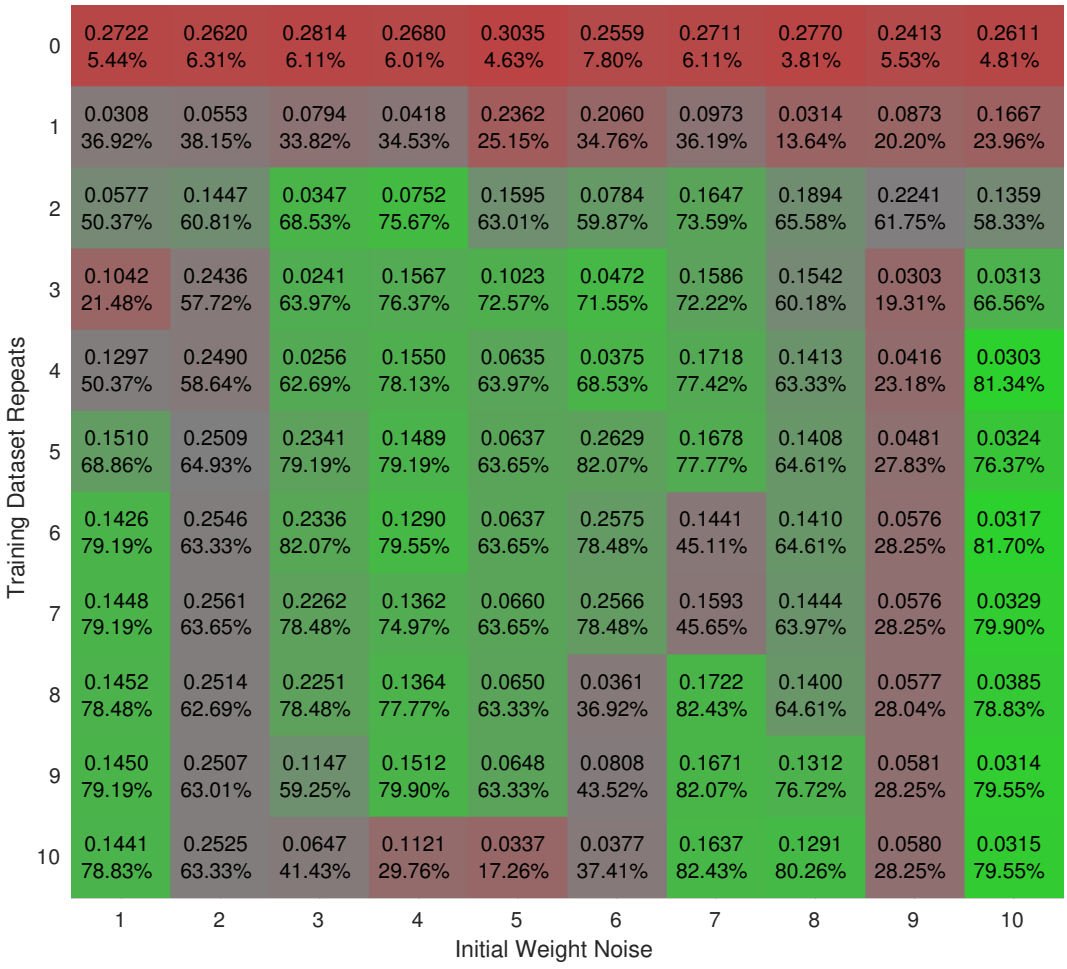

Figure 4.28: Results for the sweep of option 5.2 (down) versus option 2.10 (right). Each column shows one instance of the device through time with a specific noise pattern affecting it.

In Figure 4.28 each column represents Figure of Merit changes as the amount of training set exposure repeats increases. Baseline performance without learning is shown in the top row. Compared to the baseline each instance improves its performance using unsupervised learning, with most instances achieving an E_{MDS} of 0.1 and a BPR of 80%. However most instances suffer from learning instability.

Conclusion: for a repetition count of up to five most instances consistently improve performance. For higher repetition counts learning instability and drops in peformance are expected.

4.7.7.2 Training Dataset Repeat Count With Fast Learning

For this experiment, in contrast to Section 4.7.7.1, fast but unreliable learning is used and the effect of training exposure time is investigated again. Except for the the change of $A_{2\_}$ from 78.9 fF to 793 fF no changes were made.

Hypothesis: again, it is known that reducing the repetition count below the original value of 5 leads to worse performance. Compared to slow learning, more variation but better performance is expected.

Table 4.30: Training sweep 2 setup information. Total combinations = 110

Option 6.4 Values (11x)
0
...
10

Noise (10x)
1
...
10

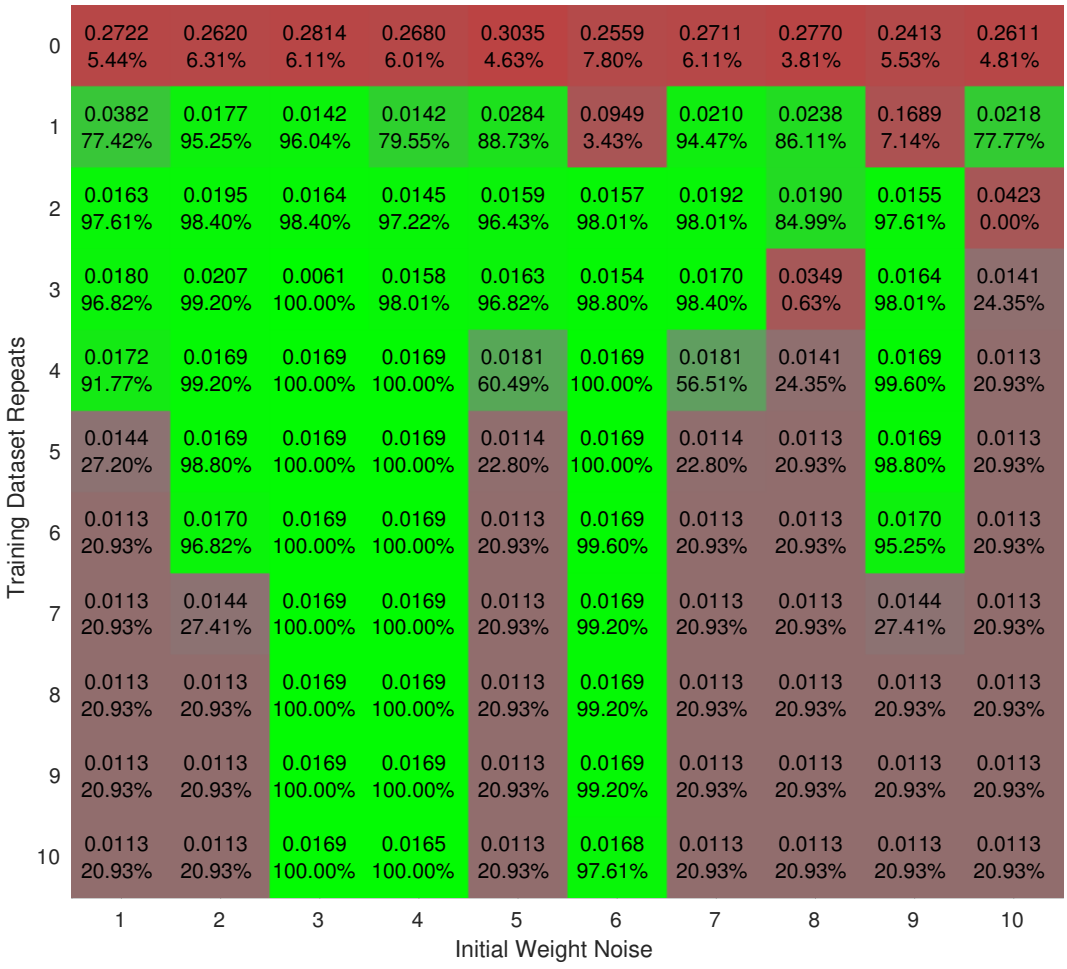

Figure 4.29: Results for the sweep of option 5.2 (down) versus option 2.10 (right). Each column shows one instance of the device through time with a specific noise pattern affecting it.

Compared to the experiment in Section 4.7.7.1, fast learning instances of the device achieve their peak performance in only 3 database repetitions on average. However, only 3 out of 9 instances with competitive performance, meaning E_{MDS} below 0.02 and BPR above 95%, retain their result after repeated training sessions.

Conclusion: this time, for a repetition count of up to three 9 out of 10 instances consistently improve performance to much higher levels than slow learning would. For higher repetition counts learning more dramatic instability and drops in peformance are expected.

5 Conclusion & Future Work

5.1 Conclusion

In this thesis a system architecture for a device incorporating a neuromorphic Spiking Neural Network (SNN) has been presented. As a result of the design space exploration that has been performed for this device it can be concluded that accurate categorization of electrocardiogram (ECG) beats using an unsupervised SNN with the Self-Organizing Map (SOM) layout and using simple Spike Timing Dependent Plasticity (STDP) as its learning rule can be performed.

The neuromorphic device consists of five top-level blocks. The first block is the Feature Detector. For this block four algorithms from literature were implemented and tested, with modifications ranging from minor to extensive. Using our own implementations and testing regime the best-performing algorithm achieves a True Positive Rate (TPR) of 99.10%, a Positive Predictive Value (PPV) of 99.58%, and 104606 total detected beats when applied to the full length of the MIT-BIH database that contains MLII leads, which contains a total of 105210 beats.

The second block is called the Feature Selector. For this block, two algorithms were implemented and tested, namely Principal Component Analysis (PCA) and correlation matrix inspection. In this document only correlation matrix inspection is performed. Depending on the acceptable amount of information loss, this algorithm reduces the data set to feed to the remaining three blocks by 21% without loss of performance of the whole device.

The next block is the Input Encoder. For this block only one algorithm was implemented and tested: Population Offset Encoding (POE). This algorithm was implemented using non-neuromorphic concepts, and is an inversible, fully temporal, unique one-to-one, high resolution encoding scheme from the real domain to the spike domain. For POE, an extensive design space exploration of all its conventional tuning paramters has been performed.

The fourth block is the SNN itself. For this block only one network layout was tested thoroughly: the SOM. This is an unsupervised categorization network layout that is combined with STDP as the learning rule. For this network layout 28 key configuration options were extensively tested. The Multidimensional Scaling Error (E_{MDS}) and Beat Pair Recognition (BPR) metrics of the output layer were used as Figures of Merit (FOM). Making extensive use of the divide-and-conquer approach combined with exhaustive exploratory sweeps a detailed characterization of this block has been presented with well-documented behaviour changes as a result of various configuration changes. This block is implemented while keeping neuromorphic hardware implementability in mind, and is also fully digitally hardware syntesizable, conforming with the goal of designing a low power wearable version of this device. For this block, when looking at the FOM, two local maxima were found: one region where slow but

reliable learning occurs, and one region where fast but unreliable learning occurs. In this region an E_{MDS} of 0.1113 and a BPR of 70.03% is achieved with excellent UV layer noise robustness. The other local maximum found uses fast learning and achieves an E_{MDS} of 0.0169 with a BPR of 100%. However, hardness to configuration parameter changes and UV layer noise is of great concern at this maximum.

The final block is the Output Decoder that decodes spiking information generated by the output layer of the SNN from spikes back to the real domain. This is the actual block that calculates the E_{MDS} and BPR for the device.

5.2 Future Work

As mentioned throughout this document many additional steps could be taken to improve the device. These steps could help the device by lowering compute complexity, lowering hardware cost, improving FOMs, and so forth.

First, for the Feature Detector phase, the author believes that more exhaustive tuning of available configuration options could improve the performance of the algorithms presented. Also, a more optimal combination of QRS detection and P, T detection algorithms could be used since the current implementation of P, T detection (Tekeste) uses information generated by an inferior QRS detection scheme (Zong) and combines these to generate the full dataset.

Also for the Feature Detector phase and the device as a whole, information from multiple parallel leads could be used for categorization, in similar manner to which multiple features of beats from a single lead are used in parallel now. Multiple leads provide a richer picture of the behaviour of the heart and could therefore be used to improve categorization.

For the Feature Selection phase, a more robust automated approach like PCA should be able to outperform the manual approach currently used. Recall that this approach was abandoned due to unsatisfactory results.

For the Input Encoder, dataset composition and ordering could be taken into account. Currently no effort is taken to emphasize important types of beats in the training dataset that should receive a higher than average degree of learning. The lack of this selection process also makes frequently occurring very similar beats prone to receive overlearning. Although it is briefly touched upon in this document it was not investigated further due to time constraints.

For the SNN, there are many possible ways that might improve performance. Firstly, the conductivity dependency function of STDP was implemented but not thoroughly tested due to time constraints. This function drastically alters learning rates through time and can be used to stabilize learning results, an area that the current implementation has significant problems with (see Section 4.7.7). For example, the concept of synaptic bistability is easily implemented using an exponential conductivity dependency function. Similarly, time dependency functions of STDP have also been used in literature for learning result stabilization.

Furthermore, other initial conductivity distributions than the currently used uniform one can be used. A gaussian distribution would be more biologically relevant and deserves extra attention.

Appendix A

A.1 Feature Detector

A.1.1 Li QRS Detection Results Log

```
Running test "Data/100"...
Running test "Data/101"...
Running test "Data/102"...
ERROR: could not find MLII lead, skipping current test
Running test "Data/103"...
Running test "Data/104"...
ERROR: could not find MLII lead, skipping current test
Running test "Data/105"...
Running test "Data/106"...
Running test "Data/107"...
Running test "Data/108"...
Running test "Data/109"...
Running test "Data/111"...
Running test "Data/112"...
Running test "Data/113"...
Running test "Data/114"...
Running test "Data/115"...
Running test "Data/116"...
Running test "Data/117"...
Running test "Data/118"...
Running test "Data/119"...
Running test "Data/121"...
Running test "Data/122"...
Running test "Data/123"...
Running test "Data/124"...
Running test "Data/200"...
Running test "Data/201"...
Running test "Data/202"...
Running test "Data/203"...
Running test "Data/205"...
Running test "Data/207"...
Running test "Data/208"...
Running test "Data/209"...
Running test "Data/210"...
Running test "Data/212"...
Running test "Data/213"...
Running test "Data/214"...
Running test "Data/215"...
Running test "Data/217"...
Running test "Data/219"...
Running test "Data/220"...
Running test "Data/221"...
Running test "Data/222"...
Running test "Data/223"...
```

```
Running test "Data/228"...
Running test "Data/230"...
Running test "Data/231"...
Running test "Data/232"...
Running test "Data/233"...
Running test "Data/234"...

Reference       | Li
Patient   Truth |  Count     TP     FP     FN      TPR      PPV
-------------------------------------------------------------
Data/100   2273 |   2273   2273      0      0  100.00%  100.00%
Data/101   1869 |   1864   1863      1      6   99.68%   99.95%
Data/103   2084 |   2084   2084      0      0  100.00%  100.00%
Data/105   2602 |   2548   2529     19     73   97.19%   99.25%
Data/106   2027 |   1994   1993      1     34   98.32%   99.95%
Data/107   2137 |   2108   2108      0     29   98.64%  100.00%
Data/108   1771 |   1749   1707     42     64   96.39%   97.60%
Data/109   2532 |   2523   2523      0      9   99.64%  100.00%
Data/111   2124 |   2121   2118      3      6   99.72%   99.86%
Data/112   2539 |   2539   2539      0      0  100.00%  100.00%
Data/113   1795 |   1795   1795      0      0  100.00%  100.00%
Data/114   1880 |   1878   1878      0      2   99.89%  100.00%
Data/115   1959 |   1953   1953      0      6   99.69%  100.00%
Data/116   2412 |   2395   2392      3     20   99.17%   99.87%
Data/117   1535 |   1535   1535      0      0  100.00%  100.00%
Data/118   2278 |   2278   2277      1      1   99.96%   99.96%
Data/119   1987 |   1987   1987      0      0  100.00%  100.00%
Data/121   1863 |   1860   1859      1      4   99.79%   99.95%
Data/122   2478 |   2476   2476      0      2   99.92%  100.00%
Data/123   1518 |   1518   1518      0      0  100.00%  100.00%
Data/124   1619 |   1619   1617      2      2   99.88%   99.88%
Data/200   2601 |   2591   2589      2     12   99.54%   99.92%
Data/201   1963 |   1970   1959     11      4   99.80%   99.44%
Data/202   2138 |   2132   2132      0      6   99.72%  100.00%
Data/203   3006 |   2818   2799     19    207   93.11%   99.33%
Data/205   2657 |   2635   2635      0     22   99.17%  100.00%
Data/207   1862 |   1947   1820    127     42   97.74%   93.48%
Data/208   2963 |   2891   2889      2     74   97.50%   99.93%
Data/209   3012 |   3005   3005      0      7   99.77%  100.00%
Data/210   2651 |   2597   2595      2     56   97.89%   99.92%
Data/212   2749 |   2748   2748      0      1   99.96%  100.00%
Data/213   3251 |   3245   3214     31     37   98.86%   99.04%
Data/214   2267 |   2256   2250      6     17   99.25%   99.73%
Data/215   3363 |   3354   3353      1     10   99.70%   99.97%
Data/217   2209 |   2191   2190      1     19   99.14%   99.95%
Data/219   2154 |   2154   2154      0      0  100.00%  100.00%
Data/220   2048 |   2048   2048      0      0  100.00%  100.00%
Data/221   2427 |   2414   2397     17     30   98.76%   99.30%
Data/222   2483 |   2467   2466      1     17   99.32%   99.96%
Data/223   2605 |   2587   2587      0     18   99.31%  100.00%
Data/228   2077 |   1974   1939     35    138   93.36%   98.23%
Data/230   2257 |   2256   2256      0      1   99.96%  100.00%
Data/231   1571 |   1571   1571      0      0  100.00%  100.00%
Data/232   1780 |   1850   1777     73      3   99.83%   96.05%
Data/233   3081 |   3055   3055      0     26   99.16%  100.00%
Data/234   2753 |   2753   2753      0      0  100.00%  100.00%
-------------------------------------------------------------
```

```
Mean        2287 |   2274   2265      9     22   99.10%   99.58%
Sum       105210 | 104606 104205    401   1005 4558.73% 4580.52%
```

A.1.2 Pan-Tompkins QRS Detection Results Log

```
Running test "Data/100"...
Running test "Data/101"...
Running test "Data/102"...
ERROR: could not find MLII lead, skipping current test
Running test "Data/103"...
Running test "Data/104"...
ERROR: could not find MLII lead, skipping current test
Running test "Data/105"...
Running test "Data/106"...
Running test "Data/107"...
Running test "Data/108"...
Running test "Data/109"...
Running test "Data/111"...
Running test "Data/112"...
Running test "Data/113"...
Running test "Data/114"...
Running test "Data/115"...
Running test "Data/116"...
Running test "Data/117"...
Running test "Data/118"...
Running test "Data/119"...
Running test "Data/121"...
Running test "Data/122"...
Running test "Data/123"...
Running test "Data/124"...
Running test "Data/200"...
Running test "Data/201"...
Running test "Data/202"...
Running test "Data/203"...
Running test "Data/205"...
Running test "Data/207"...
Running test "Data/208"...
Running test "Data/209"...
Running test "Data/210"...
Running test "Data/212"...
Running test "Data/213"...
Running test "Data/214"...
Running test "Data/215"...
Running test "Data/217"...
Running test "Data/219"...
Running test "Data/220"...
Running test "Data/221"...
Running test "Data/222"...
Running test "Data/223"...
Running test "Data/228"...
Running test "Data/230"...
Running test "Data/231"...
Running test "Data/232"...
Running test "Data/233"...
Running test "Data/234"...

Reference       | Pan Tompkins
Patient   Truth |  Count     TP     FP     FN      TPR      PPV
-------------------------------------------------------------
Data/100   2273 |   2273   2272      1      1   99.96%   99.96%
```

Data/101	1869	1870	1864	6	5	99.73%	99.68%
Data/103	2084	2083	2083	0	1	99.95%	100.00%
Data/105	2602	2596	2530	66	72	97.23%	97.46%
Data/106	2027	2003	2002	1	25	98.77%	99.95%
Data/107	2137	2473	2122	351	15	99.30%	85.81%
Data/108	1771	1731	1707	24	64	96.39%	98.61%
Data/109	2532	2527	2518	9	14	99.45%	99.64%
Data/111	2124	2124	2121	3	3	99.86%	99.86%
Data/112	2539	2541	2539	2	0	100.00%	99.92%
Data/113	1795	1794	1794	0	1	99.94%	100.00%
Data/114	1880	1878	1877	1	3	99.84%	99.95%
Data/115	1959	1953	1953	0	6	99.69%	100.00%
Data/116	2412	2392	2386	6	26	98.92%	99.75%
Data/117	1535	1536	1529	7	6	99.61%	99.54%
Data/118	2278	2278	2278	0	0	100.00%	100.00%
Data/119	1987	1988	1987	1	0	100.00%	99.95%
Data/121	1863	1866	1860	6	3	99.84%	99.68%
Data/122	2478	2477	2477	0	1	99.96%	100.00%
Data/123	1518	1515	1515	0	3	99.80%	100.00%
Data/124	1619	1618	1615	3	4	99.75%	99.81%
Data/200	2601	2602	2596	6	5	99.81%	99.77%
Data/201	1963	1939	1907	32	56	97.15%	98.35%
Data/202	2138	2122	2115	7	23	98.92%	99.67%
Data/203	3006	2710	2678	32	328	89.09%	98.82%
Data/205	2657	2651	2651	0	6	99.77%	100.00%
Data/207	1862	2222	1837	385	25	98.66%	82.67%
Data/208	2963	2939	2617	322	346	88.32%	89.04%
Data/209	3012	3009	3008	1	4	99.87%	99.97%
Data/210	2651	2509	2502	7	149	94.38%	99.72%
Data/212	2749	2748	2748	0	1	99.96%	100.00%
Data/213	3251	3243	3243	0	8	99.75%	100.00%
Data/214	2267	2261	2256	5	11	99.51%	99.78%
Data/215	3363	3351	3348	3	15	99.55%	99.91%
Data/217	2209	2215	2200	15	9	99.59%	99.32%
Data/219	2154	2154	2151	3	3	99.86%	99.86%
Data/220	2048	2048	2047	1	1	99.95%	99.95%
Data/221	2427	2420	2409	11	18	99.26%	99.55%
Data/222	2483	2471	2466	5	17	99.32%	99.80%
Data/223	2605	2558	2552	6	53	97.97%	99.77%
Data/228	2077	1697	1686	11	391	81.17%	99.35%
Data/230	2257	2256	2256	0	1	99.96%	100.00%
Data/231	1571	1571	1571	0	0	100.00%	100.00%
Data/232	1780	1785	1779	6	1	99.94%	99.66%
Data/233	3081	3074	3074	0	7	99.77%	100.00%
Data/234	2753	2751	2751	0	2	99.93%	100.00%
---	---	---	---	---	---	---	---
Mean	2287	2279	2250	29	38	98.47%	98.79%
Sum	105210	104822	103477	1345	1733	4529.47%	4544.53%

A.1.3 Tekeste QRS Detection Results Log

```
Running test "Data/100"...
Running test "Data/101"...
Running test "Data/102"...
ERROR: could not find MLII lead, skipping current test
Running test "Data/103"...
Running test "Data/104"...
ERROR: could not find MLII lead, skipping current test
Running test "Data/105"...
Running test "Data/106"...
Running test "Data/107"...
Running test "Data/108"...
Running test "Data/109"...
Running test "Data/111"...
Running test "Data/112"...
Running test "Data/113"...
Running test "Data/114"...
Running test "Data/115"...
Running test "Data/116"...
Running test "Data/117"...
Running test "Data/118"...
Running test "Data/119"...
Running test "Data/121"...
Running test "Data/122"...
Running test "Data/123"...
Running test "Data/124"...
Running test "Data/200"...
Running test "Data/201"...
Running test "Data/202"...
Running test "Data/203"...
Running test "Data/205"...
Running test "Data/207"...
Running test "Data/208"...
Running test "Data/209"...
Running test "Data/210"...
Running test "Data/212"...
Running test "Data/213"...
Running test "Data/214"...
Running test "Data/215"...
Running test "Data/217"...
Running test "Data/219"...
Running test "Data/220"...
Running test "Data/221"...
Running test "Data/222"...
Running test "Data/223"...
Running test "Data/228"...
Running test "Data/230"...
Running test "Data/231"...
Running test "Data/232"...
Running test "Data/233"...
Running test "Data/234"...

Reference       | Tekeste
Patient   Truth |  Count     TP     FP     FN      TPR      PPV
-------------------------------------------------------------
Data/100   2273 |   2273   2273      0      0  100.00%  100.00%
```

Data/101	1869	1868	1865	3	4	99.79%	99.84%
Data/103	2084	2081	2081	0	3	99.86%	100.00%
Data/105	2602	2667	2556	111	46	98.23%	95.84%
Data/106	2027	1996	1988	8	39	98.08%	99.60%
Data/107	2137	2130	2127	3	10	99.53%	99.86%
Data/108	1771	1965	1614	351	157	91.13%	82.14%
Data/109	2532	2522	2522	0	10	99.61%	100.00%
Data/111	2124	2123	2123	0	1	99.95%	100.00%
Data/112	2539	2540	2539	1	0	100.00%	99.96%
Data/113	1795	1795	1795	0	0	100.00%	100.00%
Data/114	1880	1877	1877	0	3	99.84%	100.00%
Data/115	1959	1953	1953	0	6	99.69%	100.00%
Data/116	2412	2402	2391	11	21	99.13%	99.54%
Data/117	1535	1536	1535	1	0	100.00%	99.93%
Data/118	2278	2280	2278	2	0	100.00%	99.91%
Data/119	1987	1988	1987	1	0	100.00%	99.95%
Data/121	1863	1864	1860	4	3	99.84%	99.79%
Data/122	2478	2476	2476	0	2	99.92%	100.00%
Data/123	1518	1515	1515	0	3	99.80%	100.00%
Data/124	1619	1611	1610	1	9	99.44%	99.94%
Data/200	2601	2606	2598	8	3	99.88%	99.69%
Data/201	1963	1899	1895	4	68	96.54%	99.79%
Data/202	2138	2119	2117	2	21	99.02%	99.91%
Data/203	3006	2969	2886	83	120	96.01%	97.20%
Data/205	2657	2650	2650	0	7	99.74%	100.00%
Data/207	2334	2242	2104	138	230	90.15%	93.84%
Data/208	2963	2946	2918	28	45	98.48%	99.05%
Data/209	3012	3007	3006	1	6	99.80%	99.97%
Data/210	2651	2582	2576	6	75	97.17%	99.77%
Data/212	2749	2747	2747	0	2	99.93%	100.00%
Data/213	3251	3240	3240	0	11	99.66%	100.00%
Data/214	2267	2257	2254	3	13	99.43%	99.87%
Data/215	3363	3358	3356	2	7	99.79%	99.94%
Data/217	2209	2201	2200	1	9	99.59%	99.95%
Data/219	2154	2152	2150	2	4	99.81%	99.91%
Data/220	2048	2048	2047	1	1	99.95%	99.95%
Data/221	2427	2410	2396	14	31	98.72%	99.42%
Data/222	2483	2474	2470	4	13	99.48%	99.84%
Data/223	2605	2556	2555	1	50	98.08%	99.96%
Data/228	2077	1965	1864	101	213	89.74%	94.86%
Data/230	2257	2256	2256	0	1	99.96%	100.00%
Data/231	1571	1569	1569	0	2	99.87%	100.00%
Data/232	1780	1847	1723	124	57	96.80%	93.29%
Data/233	3081	3070	3069	1	12	99.61%	99.97%
Data/234	2753	2748	2748	0	5	99.82%	100.00%
---	---	---	---	---	---	---	---
Mean	2297	2291	2269	22	29	98.71%	98.97%
Sum	105682	105380	104359	1021	1323	4540.87%	4552.47%

A.1.4 Zong QRS Detection Results Log

```
Running test "Data/100"...
Running test "Data/101"...
Running test "Data/102"...
ERROR: could not find MLII lead, skipping current test
Running test "Data/103"...
Running test "Data/104"...
ERROR: could not find MLII lead, skipping current test
Running test "Data/105"...
Running test "Data/106"...
Running test "Data/107"...
Running test "Data/108"...
Running test "Data/109"...
Running test "Data/111"...
Running test "Data/112"...
Running test "Data/113"...
Running test "Data/114"...
Running test "Data/115"...
Running test "Data/116"...
Running test "Data/117"...
Running test "Data/118"...
Running test "Data/119"...
Running test "Data/121"...
Running test "Data/122"...
Running test "Data/123"...
Running test "Data/124"...
Running test "Data/200"...
Running test "Data/201"...
Running test "Data/202"...
Running test "Data/203"...
Running test "Data/205"...
Running test "Data/207"...
Running test "Data/208"...
Running test "Data/209"...
Running test "Data/210"...
Running test "Data/212"...
Running test "Data/213"...
Running test "Data/214"...
Running test "Data/215"...
Running test "Data/217"...
Running test "Data/219"...
Running test "Data/220"...
Running test "Data/221"...
Running test "Data/222"...
Running test "Data/223"...
Running test "Data/228"...
Running test "Data/230"...
Running test "Data/231"...
Running test "Data/232"...
Running test "Data/233"...
Running test "Data/234"...

Reference       | Zong
Patient   Truth |  Count     TP     FP     FN       TPR       PPV
-------------------------------------------------------------------
Data/100   2273 |   2273   2273      0      0   100.00%   100.00%
```

56	Data/101	1869	1867	1864	3	5	99.73%	99.84%
57	Data/103	2084	2081	2081	0	3	99.86%	100.00%
58	Data/105	2602	2670	2550	120	52	98.00%	95.51%
59	Data/106	2027	1999	1991	8	36	98.22%	99.60%
60	Data/107	2137	2131	2122	9	15	99.30%	99.58%
61	Data/108	1771	1971	1630	341	141	92.04%	82.70%
62	Data/109	2532	2522	2513	9	19	99.25%	99.64%
63	Data/111	2124	2123	2117	6	7	99.67%	99.72%
64	Data/112	2539	2540	2539	1	0	100.00%	99.96%
65	Data/113	1795	1795	1794	1	1	99.94%	99.94%
66	Data/114	1880	1877	1877	0	3	99.84%	100.00%
67	Data/115	1959	1954	1953	1	6	99.69%	99.95%
68	Data/116	2412	2400	2393	7	19	99.21%	99.71%
69	Data/117	1535	1536	1535	1	0	100.00%	99.93%
70	Data/118	2278	2279	2278	1	0	100.00%	99.96%
71	Data/119	1987	1989	1987	2	0	100.00%	99.90%
72	Data/121	1863	1863	1860	3	3	99.84%	99.84%
73	Data/122	2478	2476	2476	0	2	99.92%	100.00%
74	Data/123	1518	1515	1515	0	3	99.80%	100.00%
75	Data/124	1619	1610	1609	1	10	99.38%	99.94%
76	Data/200	2601	2603	2597	6	4	99.85%	99.77%
77	Data/201	1963	1901	1894	7	69	96.48%	99.63%
78	Data/202	2138	2120	2115	5	23	98.92%	99.76%
79	Data/203	3006	2964	2867	97	139	95.38%	96.73%
80	Data/205	2657	2650	2649	1	8	99.70%	99.96%
81	Data/207	1862	2255	1845	410	17	99.09%	81.82%
82	Data/208	2963	2948	2922	26	41	98.62%	99.12%
83	Data/209	3012	3005	3005	0	7	99.77%	100.00%
84	Data/210	2651	2583	2574	9	77	97.10%	99.65%
85	Data/212	2749	2748	2747	1	2	99.93%	99.96%
86	Data/213	3251	3240	3239	1	12	99.63%	99.97%
87	Data/214	2267	2256	2254	2	13	99.43%	99.91%
88	Data/215	3363	3359	3355	4	8	99.76%	99.88%
89	Data/217	2209	2201	2200	1	9	99.59%	99.95%
90	Data/219	2154	2155	2150	5	4	99.81%	99.77%
91	Data/220	2048	2048	2047	1	1	99.95%	99.95%
92	Data/221	2427	2411	2402	9	25	98.97%	99.63%
93	Data/222	2483	2473	2470	3	13	99.48%	99.88%
94	Data/223	2605	2559	2555	4	50	98.08%	99.84%
95	Data/228	2077	1941	1852	89	225	89.17%	95.41%
96	Data/230	2257	2256	2256	0	1	99.96%	100.00%
97	Data/231	1571	1570	1568	2	3	99.81%	99.87%
98	Data/232	1780	1852	1711	141	69	96.12%	92.39%
99	Data/233	3081	3070	3068	2	13	99.58%	99.93%
100	Data/234	2753	2748	2748	0	5	99.82%	100.00%
101	---	---	---	---	---	---	---	---
102	Mean	2287	2291	2262	29	25	98.86%	98.66%
103	Sum	105210	105387	104047	1340	1163	4547.68%	4538.51%

A.2 Full DSE Sweep Results

A.2.1 Manual Feature Selection Log

(omitted)

A.2.2 Full Encoder Sweep Log

(omitted)

A.2.3 Full UV Weight Sweep Log

(omitted)

A.2.4 Full VV Weight Sweep Log

(omitted)

A.2.5 STDP Window Sweep Log

(omitted)

A.2.6 Training Sweep Log

(omitted)

Bibliography

[1] E. K. Jacob, "Classification and Categorization: A Difference that Makes a Difference," *Library Trends*, 2004.

[2] BruceBlaus, "Multipolar Neuron," 2013. [Online]. Available: https://blausen.com/?Topic=9495

[3] en:User:Chris 73 and en:User:Diberri, "Schematic of an action potential," 2007. [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/4/4a/Action_potential.svg

[4] A.L. Hodgkin, and A.F. and 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.

[5] National Institute on Aging, "Drawing illustrating the process of synaptic transmission in neurons," 2013. [Online]. Available: http://www.nia.nih.gov/alzheimers/publication/alzheimers-disease-unraveling-mystery/preface

[6] G. Bi and M. Poo, "Synaptic Modifications in Cultured Hippocampal Neurons: Dependence on Spike Timing, Synaptic Strength, and Postsynaptic Cell Type," *The Journal of Neuroscience*, 1998.

[7] J.P. Pfister and W. Gerstner, "Triplets of Spikes in a Model of Spike Timing-Dependent Plasticity," *The Journal of Neuroscience*, 2006.

[8] PhysioNet, "MIT-BIH Arrhythmia Database," 2017. [Online]. Available: https://www.physionet.org/physiobank/database/mitdb/

[9] Npatchett, "Graphical representation of Einthoven's triangle," 2015. [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/1/19/Limb_leads_of_EKG.png

[10] S.M. Bohte, H. La Poutre and J.N. Kok, "Unsupervised clustering with spiking neurons by sparse temporal coding and multilayer RBF networks," *IEEE Transactions on Neural Networks*, [2002.

[11] M.R. Azghadi and others, "Spike-Based Synaptic Plasticity in Silicon: Design, Implementation, Application, and Challenges," *Proceedings of the IEEE*, 2013.

[12] ECGPedia, "ECG Reference Card," 2010. [Online]. Available: https://www.ecgpedia.org/A4/ECGpedia_on_1_A4En.pdf

[13] C.A. Mead, *Analog VLSI and Neural systems*. Addison-Wesley, 1989.

[14] D.E. Rumelhart, R. Durbin, R. Golden and Y. Chauvin, *Backpropagation*. Hillsdale, NJ, USA: L. Erlbaum Associates Inc., 1995.

[15] D. Nelson and M. Cox, *Lehninger Principles of Biochemistry.* W.H. Freeman, 2013.

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

[17] R.B. Stein, "Some Models of Neuronal Variability," *Biophysical Journal*, 1967.

[18] W. Gerstner, and W.M. Kistler, *Spiking Neuron Models.* Cambridge University Press, 2002.

[19] G. Indiveri and others, "Neuromorphic silicon neuron circuits," *Frontiers in Neuroscience*, 2011.

[20] G. Indiveri and S. Liu, "Memory and information processing in neuromorphic systems," *Proceedings of the IEEE*, 2015.

[21] J.H.B. Wijekoon and P. Dudek, "Compact silicon neuron circuit with spiking and bursting behaviour," *Neural Networks*, 2008.

[22] R. Wang and others, "A compact aVLSI conductance-based silicon neuron," *IEEE*, 2015.

[23] D.O. Hebb, *The Organization of Behavior: A Neuropsychological Theory.* Wiley, 1949.

[24] E.L. Bienenstock, L.N. Cooper and P.W. Munro, "Theory for the development of neuron selectivity: orientation specificity and binocular interaction in visual cortex," *Journal of Neuroscience*, 1982.

[25] S. Fusi and others, "Spike-driven synaptic plasticity: theory, simulation, VLSI implementation," *Neural Computation*, 2000.

[26] J.M. Brader, W. Senn and S. Fusi, "Learning real-world stimuli in a neural network with spike-driven synaptic dynamics," *Neural Computation*, 2007.

[27] M. Minsky and Seymour Papert, *Perceptrons: an introduction to computational geometry.* MIT Press, 1969.

[28] Cables And Sensors, "12-Lead ECG Placement Guide with Illustrations," 2017. [Online]. Available: https://www.cablesandsensors.eu/pages/12-lead-ecg-placement-guide-with-illustrations

[29] W. Einthoven, *Galvanometrische registratie van het menschelijk electrocardiogram.* Leiden, 1902.

[30] S. Furber, F. Galluppi, S. Temple, and L. Plana, "The SpiNNaker project," *Proceedings of the IEEE*, 2014.

[31] P. A. Merolla and others, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, 2014.

[32] B.V. Benjamin and others, "Neurogrid: A Mixed-Analog-Digital Multichip System for Large-Scale Neural Simulations," *IEEE*, 2014 .

[33] N. Qiao and others, "A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128K synapses," *Frontierts in Neuroscience*, 2015.

[34] C.K. Lin and others, "Programming Spiking Neural Networks on Intels Loihi," *IEEE*, 2018.

[35] B. Ruf and M. Schmitt, "Self-Organization of Spiking Neurons Using Action Potential Timing," *IEEE*, 1998.

[36] T. Rumbell and others, "A Spiking Self-Organizing Map Combining STDP, Oscillations, and Continuous Learning," *IEEE Transaction on Neural Networks*, 2014.

[37] A.L. Goldberger and others, "Components of a New Research Resource for Complex Physiologic Signals," *Circulation Electronic Pages*, 2000.

[38] PhysioNet, "PhysioBank Annotations," 2017. [Online]. Available: https://www.physionet.org/physiobank/annotations.shtml

[39] A.E. Kolagasioglu, "Energy Efficient Feature Extraction for Single-Lead ECG Classification Based On Spiking Neural Networks," Master's thesis, Delft University of Technology, 2018.

[40] PhysioNet, "SIGNAL(5)," 2018. [Online]. Available: https://www.physionet.org/physiotools/wag/signal-5.htm#sect9

[41] G.B. Moody, "WFDB Applications Guide," 2018. [Online]. Available: https://www.physionet.org/physiotools/wag/wag.htm

[42] P. Jiapu, W.J. Tompkins, "A Real-Time QRS Detection Algorithm," *IEEE Transactions on Biomedical Engineering*, 1985.

[43] W. Zong, G.B. Moody and D. Jiang, "A Robust Open-source Algorithm to Detect Onset and Duration of QRS Complexes," *Computers in Cardiology*, 2003.

[44] C. Li, C. Zheng and C. Tai, "Detection of ECG Characteristic Points Using Wavelet Transforms," *IEEE Transactions on Biomedical Engineering*, 1995.

[45] T. Tekeste and others, "Adaptive ECG Interval Extraction," *IEEE*, 2015.

[46] T. Tekeste and others, "A Nano-Watt ECG Feature Extraction Engine in 65nm Technology," *IEEE Transactions on Circuits and Systems*, 2016.

[47] W. Zong, G.B. Moody and D. Jiang, "WQRS(1)," 2018. [Online]. Available: https://physionet.org/physiotools/wag/wqrs-1.htm

[48] S. Mallat, "Zero-Crossings of a Wavelet Transform," *IEEE Transactions on Information Theory*, 1991.

[49] M. Holschneider, R. Kronland-Martinet, J. Morlet and P. Tchamitchian, *A real-time algorithm for signal analysis with the help of the wavelet transform.* Springer-Verlag, 1989.

[50] G.J. Goodhill and T.J. Sejnowski, "A Unifying Objective Function for Topographic Mappings," *Neural Computation*, 1997.

[51] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biological Cybernetics*, pp. 59–69, 1982.

[52] R. Keith-Magee, "Learning and development in Kohonen-style self organising maps," Ph.D. dissertation, Curtin University, 2001.

[53] Randall Munroe, "Machine Learning," 2017. [Online]. Available: https://xkcd.com/1838/

[54] P. Laguna and others, "A Database for Evaluation of Algorithms for Measurement of QT and Other Waveform Intervals in the ECG," *Computers in Cardiology*, 1997.

[55] D.A. Pomerleau, "Knowledge-based Training of Artificial Neural Networks for Autonomous Robot Driving," *Robot Learning*, 1993.

[56] G. Cybenkot, "Approximation by Superpositions of a Sigmoidal Function," *Math. Control Signals System*, 1989.

[57] K. Hornik, "Approximation Capabilities of Multilayer Feedforward Networks," *Neural Networks*, 1990.

[58] R.F. Schmidt and G. Thews, *Human Physiology.* Springer, 1989.

[59] American National Standards Institute, "IEEE Standard for Binary Floating-Point Arithmetic," *Standards Committee of the IEEE Computer Society*, 1985.