EXAMENSARBETE

Pattern recognition with spiking neural networks and the ROLLS low-power online learning neuromorphic processor

Andreas Ternstedt

Civilingenjör, Teknisk fysik och elektroteknik
2017

Luleå tekniska universitet
Institutionen för system- och rymdteknik

LULEÅ TEKNISKA UNIVERSITET

Abstract

Online monitoring applications requiring advanced pattern recognition capabilities implemented in resource-constrained wireless sensor systems are challenging to construct using standard digital computers. An interesting alternative solution is to use a low-power neuromorphic processor like the ROLLS, with subthreshold mixed analog/digital circuits and online learning capabilities that approximate the behavior of real neurons and synapses. This requires that the monitoring algorithm is implemented with spiking neural networks, which in principle are efficient computational models for tasks such as pattern recognition. In this work, I investigate how spiking neural networks can be used as a pre-processing and feature learning system in a condition monitoring application where the vibration of a machine with healthy and faulty rolling-element bearings is considered. Pattern recognition with spiking neural networks is investigated using simulations with *Brian* – a Python-based open source toolbox – and an implementation is developed for the ROLLS neuromorphic processor. I analyze the learned feature-response properties of individual neurons. When pre-processing the input signals with a neuromorphic cochlea known as the AER-EAR system, the ROLLS chip learns to classify the resulting spike patterns with a training error of less than 1 %, at a combined power consumption of approximately $30\,mW$. Thus, the neuromorphic hardware system can potentially be realized in a resource-constrained wireless sensor for online monitoring applications. However, further work is needed for testing and cross validation of the feature learning and pattern recognition networks.

Preface

My first and foremost thanks go to my supervisors Fredrik Sandin and Giacomo Indiveri for the opportunity, support, and provision of resources, enabling me to work on this project.

I acknowledge support from the International Cooperation in Research and Higher Education (STINT), grant number IG2011-2025, in the form of a scholarship and the possibility to attend the CapoCaccia Neuromorphic Engineering Workshop - 2016.

I would like to thank the people at the Institute of Neuroinformatics (INI), UZH & ETH Zürich, for their steady guidance, thought-provoking advice, and friendship. Special thanks to Ole Richter, Raphaela Kreiser, Dora Sumislawska, Timoleon Moraitis, Federico Corradi and Shih-Chii Liu.

Contents

CHAPTER 1

Introduction

Artificial sensors systems, such as microphones or cameras, are measuring and observing large amounts of high-dimensional information every second. It is typical for these sort of sensor systems to densely sample information, thus generating data sets of multiple correlated versions of the same physical world [1]. The relevant information about the entities recorded is often of much lower dimensionality compared to such data sets. Instead of lowering the sampling rates it is common to pre-process the sensor signals in monitoring applications to remove some redundant information. Applications often require computational methods for advanced pattern recognition and signal processing, which has gained considerable attention from researchers across the globe.

Implementing monitoring applications in resource constrained wireless sensor systems are challenging, especially for sensor signals requiring high sampling rates and when monitoring over extended time periods are necessary. For example, condition monitoring of complex machines and infrastructure require online analysis of signals sampled at several kHz. Standard digital computers used for such monitoring applications rely on iterative algorithms, which require more energy than what is typically available in wireless sensor systems. Wireless sensors are highly interesting in many applications where cables are error prone or cannot be used, or where wired sensors are too costly. Thus, enabling online pattern recognition at high sampling rates of several kHz and low power in the milliwatt range would be a significant enabler for new applications of wireless sensors and the development of Internet of Things (IoT).

Research in neuroscience has shown that information processing in biology, such as in the human brain, is specialized to discover and characterize patterns in the environment with highly energy efficient processes [2]. Biological sensor systems, such as ears and eyes, are very efficient in recording information and use different methods compared to microphones or cameras. Even though conventional computers have fast hardware, the nervous system outperform such system on relatively routine functions such as pattern recognition and sensory processing [3]. This is due to the fact that information processes in biology use fundamentally different computational operations; with a massively paral-

lel architecture of analog, low-powered and *spiking* elements of neurons, connected to a network by synapses with intrinsic plasticity. Studying these behaviors of natural sensors and information processing in biology could prove useful for the development of artificial systems, in order to make them as reliable, power-efficient, robust, and easily adaptable as the living creatures in nature. With a new generation of neuro-computing hardware emerging, the potential for solving machine learning tasks and develop monitoring systems using such hardware is high [4].

1.1 Background

Automated systems for monitoring applications including advanced pattern recognition embedded in wireless sensor systems are challenging to realize using conventional technology. Even though machine learning and standard digital processor design methods are improving; with transistors getting smaller, processing capacity increasing and the energy efficiency improving. According to Koomey's law [5], the number of computations that can be performed with one Joule of energy has doubled for each 1.6 years in standard digital computers. However, this is currently not sufficient for building wireless sensor systems that can solve advanced signal processing and pattern recognition problems, for example in an application like online condition monitoring. Online pattern recognition and learning are challenging due to the resource constraints associated with a wireless operation. In particular, the power available from cost-effective batteries and energy harvesting solutions is limited, and the communication of information is relatively costly in terms of power compared to computation.

In today's modern world is it common for mechanical or electrical systems being integrated with small embedded sensor platforms. These platforms are often added to systems in order to make them more "aware" of its surroundings, by either sensing or actuate with physical objects. They are relatively inexpensive computers and can be found for example in infrastructure and consumer electronics, in vehicles, within the industry, army and aviation, and even in space. However the sensor systems can be difficult for programmers to work with; As they often operate in real-time, with small memory to fit programs in, the need to conserve energy whenever possible, and high requirements on the robustness to failures. Furthermore, embedded systems with low power consumption many lack abstractions such as virtualization and cache hierarchies, and wireless communication is a common challenge in embedded applications [6]. Sending data wirelessly, especially from sensors that operate at high frequencies, use a significant amount of power in small embedded systems. Therefore having a low-power sensor system that analyze or reduce the redundant information before transmitting the data, similar to biological sensor systems, could reduce the power consumption significantly, thus making it possible for wireless operation while only relying on a constrained power source. With wireless communication, sensor systems can be integrated with the IoT, which could result in improved efficiency, accuracy and have economical benefits [7]. For example, low-power

sensor systems could prove useful in monitoring applications such as condition monitoring, security systems, drones, wildlife, traffic and in urban environments. One way to bridge this gap is to consider neuromorphic chips, which mimics the function of the nervous system using electronic nanodevices and standard semiconductor manufacturing technologies. A handful of these neuromorphic computers developed today are the SpiNNaker (Human Brain Project), the BrainScaleS (Human Brain Project), the SyNAPSE (IBM/Darpa) and the ROLLS (INI).

The new computational engineering approach of neuromorphic hardware are typically composed of very large scale integration (VLSI) technology with communication schemes optimized for spiking neural network architectures [8, 9]. Comparing to standard digital computers that use sequential and synchronous clocks, are not optimally suited for implementing massive parallel neural network architectures, as neurons and synapses in these systems have to operate one at a time. The neuromorphic processors are designed to emulate the organization and function of nervous systems and with mixed analog-digital subthreshold electronic circuits that are fabricated in the complementary metal-oxide-semiconductor (CMOS) medium, to reproduce faithfully neural and synaptic dynamics [10]. The significance of neuromorphic systems is that the CMOS circuits mimic the physics of biological neural systems with the physics of transistors biased in the subthreshold regime [11], which enable low power consumption and use of as little silicon real-estate as possible. This approach accomplishes systems that not only recreates physical behaviors that are analogous to the nervous systems but also operates in real-time irrespective of the network size.

1.2 Problem description

The main focus of this project is to investigate spiking neural network models and implementations for a low-power sensor system that can function as a condition monitoring system on the reconfigurable on-line learning spiking (ROLLS) neuromorphic processor. This includes the following specific tasks that define the problem and goals of the project:

- Simulate and evaluate basic spiking neural network models for pattern recognition that are compatible with the low-power reconfigurable neuromorphic processor, ROLLS.

- Study different signal pre-processing methods and the possibility to classify machines with healthy and faulty rolling-element bearings using the pattern recognition methods and recorded vibration data.

- Based on the simulation results, adapt and implement a spiking neural network model for pattern recognition and classification in the ROLLS and study the on-line learning and pattern recognition features of the chip.

- Study the receptive fields of the trained neurons and investigate qualitative differences between features corresponding to healthy and faulty bearings.

CHAPTER 2

Spiking neural network

Spiking neural networks (SNNs) are the third generation of artificial neural networks (ANNs) that have been developed to describe the spiking behavior of biological neurons [12]. Today, they are used in a wide range of application domains and provide state of the art results in areas such as speech recognition and image classification [13, 14]. The SNN considered in this project is based on the same neuron and synapses models as the neuromorphic processor ROLLS, which approximates the physics of real neurons and synapses with subthreshold CMOS circuits to faithfully reproduce their adaptive and dynamic behaviors [4].

This chapter introduces the theory behind the adaptive and dynamical behavior of such neurons and synapses, and the learning mechanism. The neuron circuits implemented in the ROLLS chip are based on the adaptive exponential integrate-and-fire model (AdEx) [4]. Before describing the AdEx model it is helpful to introduce a basic model of spiking neurons, as they are believed to be the primary computational elements of the brain. A basic SNN model is the leaky integrate-and-fire model (I&F) [12]. As for other SNN, the primary function of the I&F model is to describe the voltage across the cell membrane of biological neurons. Neurons receive and transmit information in the form of electrical pulses called action potentials, or *spikes*. By the precise timing of sequences of spikes, neurons can encode and processes information. Neurons have active conductance mechanisms that control the flow of ionic currents through the cell membrane and generates a voltage across the cell membrane [15]. In the leaky I&F model illustrated in Fig. 2.1, the electrical potential across the cell membrane (membrane potential), $u(t)$, can be conceived as an integration processes including the effective ionic currents, $I(t)$;

$$I(t) = C\frac{du}{dt} + \frac{u(t)}{R}, \tag{2.1}$$

combined with a nonlinear mechanism that results in two different dynamical regimes:

1. Below a specific threshold, ϑ, the neuron's membrane potential, $u(t)$, acts as a leaky

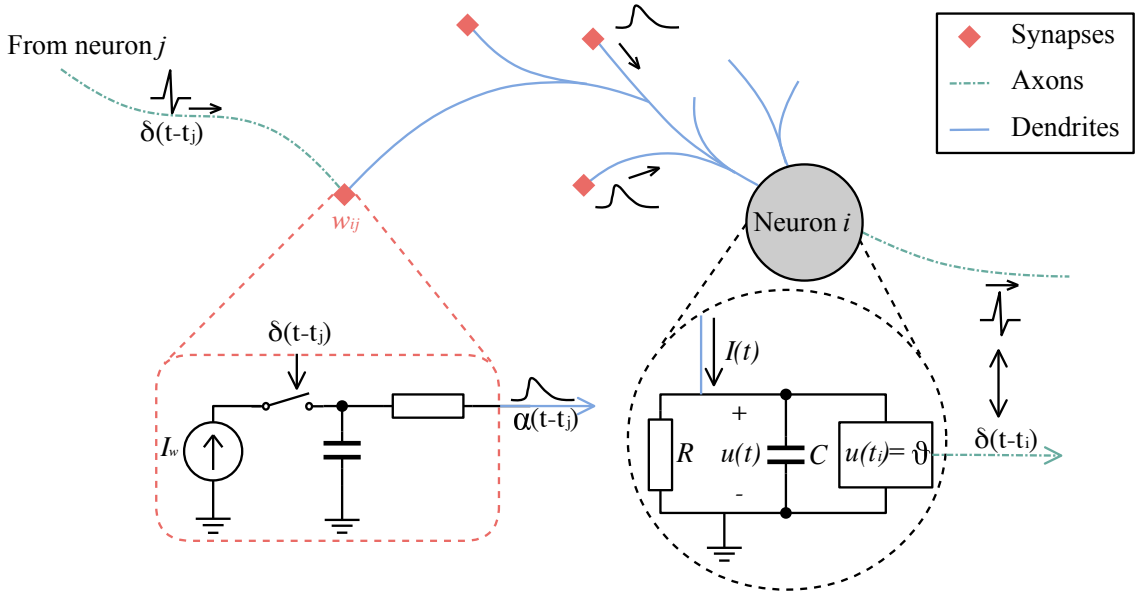


Figure 2.1: Schematic diagram of a spiking neural network described with the integrate-and-fire model. The basic circuit of a spiking neuron is the module inside the dashed circle on the right-hand side. A current $I(t)$ charge the RC circuit. If $u(t)$ of neuron i reaches ϑ a spike, $\delta(t-t_i)$, will be generated at time $t=t_i$. The dashed box illustrates a low-pass filter at the synapse that transforms the pre-synaptic spike, $\delta(t-t_j)$, to an input current pulse, $\alpha(t-t_j)$, with an amplitude determined by the synpatic efficacy, I_w.

capacitor where the voltage decays with a time constant, $\tau = RC$, to the resting potential, $u_{resting}$:

$$\tau\frac{du}{dt} = -u(t) + RI(t). \tag{2.2}$$

2. If $u(t)$ reaches the threshold level, $u(t_i) = \vartheta$, the I&F neuron will generate a transient electrical impulse at that point in time, $\delta(t-t_i)$, which defines a spike. Spikes are generated in a particular region in the biological neurons, called the axon hillock. As soon as this transient in conductance has stopped, the membrane potential returns to its ground state:

$$u(t_i^+) = u_{reseting}, \tag{2.3}$$

where t_i is the spike time and $u_{resting} < \vartheta$.

Spikes generated in a neuron will propagate along the axon of the neuron (nerve/white matter of brains) and thereby stimulate target synapses that charge other neurons.

SNN's have been used to model the observed spiking behavior of neurons in the primate and avian brain and to model basic circuit functions. Even though one spiking neuron contributes to the information processing in the central nervous system (CNS) [16], computational methods such as pattern recognition come from connecting many neurons in a network. Neural interconnections are defined by synapses, which are also observed in biological neural networks. Depending on the type of network, the synapses can couple neurons into various configurations (pools or layers) for different functionality and exert dynamics of their own. For example in the I&F model, illustrated in Fig. 2.1, a spike from neuron j stimulates neuron i over time $\alpha(t - t_j)$ [12]. The total synaptic current to neuron i is the sum over all pre-synaptic current pulses,

$$I_i(t) = \sum_j w_{ij}\alpha(t - t_j). \tag{2.4}$$

Here w_{ij} is the synaptic strength, also called the *weight*, and it is a measure of the efficacy of the synapse between neuron i and j. The weights are important as they are used to model plasticity in most ANNs, see Sec. 2.2.

2.1 Adaptive exponential integrate-and-fire model

The neuromorphic silicon neurons (SiNs) implemented in the ROLLS mimic the AdEx model, which is a generalization of the I&F model [17]. The AdEx model is designed to exhibit a wide range of neural behaviors, including a refractory period mechanism, leak conductance, and spike-frequency adaptation mechanism while allowing for a compact low-power circuit. When the AdEx circuit is tuned correctly it can produce action potentials comparable to those measured in real cortical neurons [18].

The complete AdEx equation that describes the neuron's subthreshold behavior is presented in [17]. Here I introduce a simplified model of SiNs, where the adaptation current is ignored:

$$\tau_{mem}\frac{d}{dt}I_{mem} = \frac{I_{th}}{I_\tau}I_{in} - I_{mem} + f(I_{mem}). \tag{2.5}$$

Here $f(I_{mem}) \approx \frac{I_a}{I_\tau}I_{mem}$, and I_{mem} is the current through the neuron cell membrane, which integrates I_{in} until the membrane potential V_{mem} reaches the threshold, V_{thr}. As I_{mem} gets closer to the threshold the positive feedback current, I_a, will grow exponentially and inject a current to I_{mem}. This effect models the activation and inactivation of sodium channels for producing the voltage spikes [4]. When the threshold is reached, the refractory leakage I_r is initialized to reset the membrane potential, $I_{mem} = I_{reset}$, for a time period, T_{refrac}. The reset and refractory time period emulates an electrophysiological property seen in potassium conductance ion channels, where channels are blocked by a chemical inhibitor, which prevents the neuron from being potentiated directly after an action potential [19, 4]. As in the leaky I&F model, if the input, I_{in}, is zero the I_{mem} current will decay exponentially with the time constant τ_{mem}. I_{th} and I_τ are constant bias currents, used

to set the threshold and time constant. I_{in} corresponds to the stimuli to each neuron, which is the sum of all pre-synaptic currents, i.e., similar to Eq. 2.4. In this model, all currents defined here are positive.

An example of relation between I_a and I_{mem} in the AdEx model is illustrated in Fig. 2.2, at a constant injection current, which makes I_{mem} increase rapidly from zero (its grounds state) until it converges to the size of the injection current, $10\,nA$. If I_{mem} is high enough I_a will increase to a point where it becomes 'unstable', resulting in exponential growth. The current of I_a is fed back to I_{mem} through $f(I_{mem})$, which will affect I_{mem} to grow in a similar manner, until I_{mem} reaches the threshold and resets to I_{reset} $(0\,nA)$. This process defines the spiking mechanism of the neuron.

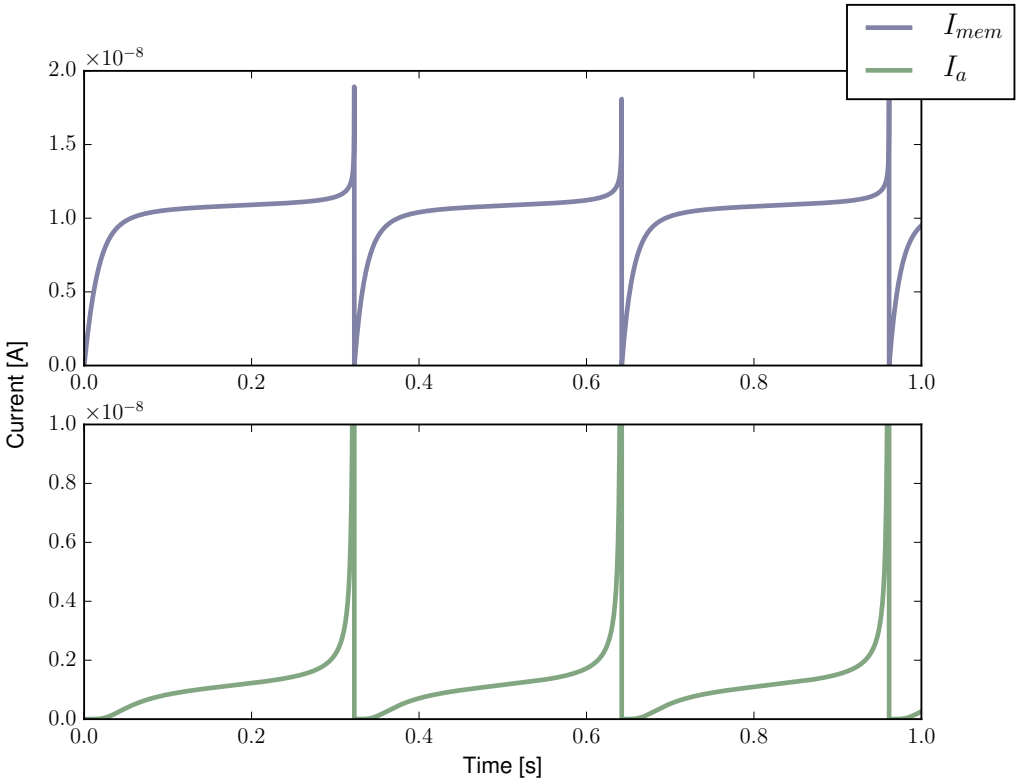


Figure 2.2: Simulation of the dynamical behavior of the AdEx model with a constant injection current $I_{in} = 10\,nA$. I_{mem} is the neurons membrane current and I_a is the positive feedback current.

Adaptation variable

Another neural behavior implemented in the SiNs is the adaptive leakage current, I_{ahp}, which is not included in the simplified version of the AdEx model (2.5). I_{ahp} is an adaptive current that is capable of short-term modulation, depending on a neuron's spike rate as a function of its input. This behavior is observed in biological neurons, where neurons become desensitized for a constant input [20].

The neuron circuit in the ROLLS chip have a differential pair integrator (DPI) implemented in negative feedback mode; The DPI can accumulate the current in a capacitor for each spike that a neuron produce and by a hyper-polarizing leakage current adapt the spike-frequency of the neuron [4]. The dynamics of the adaptive leakage current is modeled with the differential equation,

$$\frac{d}{dt}I_{ahp} = \frac{I_a^* - I_{ahp}}{\tau_{ahp}}. \tag{2.6}$$

Each spike of a neuron is integrated as a constant value, I_{wa}, in I_{ahp} and will decay exponentially with the time constant τ_{ahp}. In the complete equation of the AdEx model there is an additional term $I_{mem}(1+\frac{I_{ahp}}{I_\tau})$, which initiates the leakage current proportional to the value of I_{ahp} [17]. The dynamics of the adaptive leakage current reduce redundant spiking activity, hence reducing power consumption and bandwidth usage for the SiNs [17].

Long-term memory

Long-term memory and learning mechanisms observed in real neurons are regulated by the intracellular calcium concentration. However, the mechanisms that control the calcium ions in neurons is no trivial task to model; In contrast to sodium and potassium concentrations with large in- or efflux during action potentials, the calcium concentration depend on both diffusion and chemical calcium buffers [19]. The main effects of the calcium is to adjust neurons electrical activity (by modifying sodium and potassium concentrations) and to stimulate the release of transmitter substance. Another effect calcium hosts is the long-term modification of synaptic efficiency, which is controlled by a specific receptor, known as the N-Methyl-D-Aspartate (NMDA) channel [21]. The NMDA channels implement a Hebbian learning process, called Spike-Timing Dependent Plasticity (STDP), that depend on the relative timing between the pre- and post-synaptic spikes [22]. The Hebbian theory is often summarized in the saying, "Neurons that fire together, wire together" [23], and the STDP relate to this; By strengthen the synaptic efficacy between connected neurons with correlated spiking patterns.

The learning mechanisms in the ROLLS chip have similar behavior to the NMDA channels and also use dedicated circuits to model some of the dynamics of calcium ions, which is implemented in hardware as the differential equation [24],

$$\frac{d}{dt}I_{Ca} = \frac{I_a^* - I_{Ca}}{\tau_{Ca}}, \tag{2.7}$$

with the same dynamical behavior as I_{ahp} but with different cumulative effect, I_{wCa}, and time constant τ_{Ca}. The current, I_{Ca}, is stored in a capacitor that represent a neurons post-synaptic activity. This information is used in the learning mechanisms described in Sect. 2.2.1.

2.2 Synapse model

Synapses are the connections where neurons exchange information, creating either long-range or local connections between groups of neurons to generate different functionality. Real synapses are complex molecular machines that play an important computational role in biological neural networks [25]. They hold temporal dynamics of their own that influence the information they carry, in particular by modifying the synaptic strength (weight) that determines the efficacy of the synapse to transport ions to the post-synaptic neuron. The synapse dynamics can operate at short timescales, such as in short-term depression and facilitation, and also at long timescale in the form of learning mechanisms that modify the weights [26]. The connections can either be of inhibitory, which reduce, or excitatory which potentiate the neuron's membrane potential.

Silicon synapses implemented on the chip are capable of both short-term and long-term plasticity, and produce biologically plausible Excitatory-Post-Synaptic-Currents (EPSCs) [17, 20], in a compact analog/digital circuit [4]. The EPSCs are temporary depolarization of the post-synaptic membrane potential after a pre-synaptic spike and will stimulate the post-synaptic neuron over a time course, i.e., with similar effect as $\alpha(t - t_j)$ illustrated in Fig. 2.1. The dynamical model can be described in terms of the synaptic current I_{syn},

$$\tau_{syn}\frac{d}{dt}I_{syn} + I_{syn} = \frac{I_w I_G}{I_\tau}. \tag{2.8}$$

I_{syn} contributes to the neuron input I_{in} and therefore stimulates the neuron on a timescale defined by the synaptic time constant τ_{syn}. I_w is the synaptic efficacy and I_τ controls the time constant.

The synaptic weight, w_{pq}, have a crucial role in most ANNs, as it defines the learning state of the synapses in a single parameter. According to Hebb's postulate [27], synapses are capable of long-term neuronal plasticity in which pre- and post-synaptic contributions are strengthened or weakened by biochemical modifications. In an artificial neural network, these modifications are governed by the learning algorithm. The weight w_{pq} is the variable that quantifies the effect of a spike from a pre-synaptic neuron p on the membrane potential of the post-synaptic neuron q [28].

The synaptic efficacies I_w are normalized with the following function

$$I_q = \sum_{i=1}^{p} w_{iq} I_w. \tag{2.9}$$

Here w_{iq} are the synaptic weights and I_q is the sum of all the post-synaptic neurons i onto the pre-synaptic neuron q, as illustrated in Fig. 2.3.

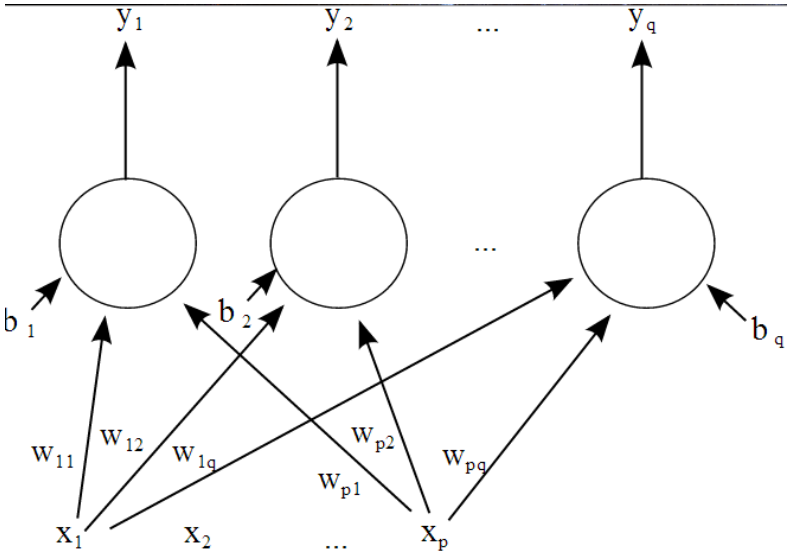


Figure 2.3: Artificial neural network schematic [29], with p pre-synaptic neurons (x), and q post-synaptic neurons (b). x_p is a pre-synaptic neuron connected to the post-synaptic neuron b_q, which synapse is weighted by w_{pq}, that results in an output y_q.

2.2.1 Learning rule

In biological neural networks synaptic plasticity is an important process involved in learning and memory. ANNs are often implemented with synaptic plasticity (or 'learning rule') which modifies the weight between neurons according to the input patterns. The fact that biological neural networks learn from examples; a child learns to recognize dogs from examples of dogs, and in a sense, it is the same for the ANNs. However, learning mechanisms based on STDP alone cannot account for all the phenomenology observed in neurophysiological experiments [30]. In conjunction with poor memory retention performance and requirements of additional mechanisms needed to learn both spike-timing correlations and mean firing rates from the input patterns [4]. Therefore, the learning circuits implemented in the ROLLS are chosen to have a plasticity mechanism that does not rely on spike-timing alone. The rule chosen is proposed in [31] and have shown to reproduce many of the synaptic plasticity behaviors observed in biological neurons and it has also been used in networks with characteristics comparable to state-of-the-art machine learning algorithms.

The learning method works as follows: The synaptic weights, w_{pq}, are only updated if a pre-synaptic spike occurs. The post-synaptic neuron's membrane potential, V_{mem}, also need to fulfill a certain condition and the recent spiking activity need to be within a specific interval, i.e., the calcium concentration need to fulfill $I_{Ca} \in [\theta_1, \theta_{k=2,3}]$. It is assumed that the synaptic weights drift to either a high or a low state, which results

in bi-stable synapses thereby avoiding the need for storing precise analog variables in long-term memory [4].

In general, the weight of synapse i is governed by the following equations:

$$\begin{cases} w_i = w_i + \Delta w^+ & \text{if } V_{mem}(t_{pre}) > \theta_{mem} \text{ and } \theta_1 < Ca(t_{pre}) < \theta_3, \\ w_i = w_i - \Delta w^- & \text{if } V_{mem}(t_{pre}) < \theta_{mem} \text{ and } \theta_1 < Ca(t_{pre}) < \theta_2, \end{cases} \tag{2.10}$$

where w_i represents the bi-stable synaptic weight. If either of the requirements is fulfilled the synaptic weight will increase or decrease by the amount defined by the terms w^+ and w^-. The condition with $V_{mem}(t_{pre})$ checks the post-synaptic neuron's membrane potential at the time of a pre-synaptic spike t_{pre}, and compares it with the threshold θ_{mem}. The term $Ca(t_{pre})$ denotes the post-synaptic calcium concentration, which is proportional to the neuron's I_{Ca} level in Eq. 2.7 at the time of the pre-synaptic spike. The three threshold levels θ_1, θ_2 and θ_3 determines if the synaptic weight is allowed to be increased, decreased, or should not be updated.

In combination with the mechanism that makes the synaptic weights step increase or decrease, the internal variable w_i is consistently being driven towards one of two stable states, depending if it is above or below a certain threshold θ_w. This makes the synaptic weights bi-stable in the following way:

$$\begin{cases} \frac{d}{dt} w_i = +C_{drift} & \text{if } w_i > \theta_w \text{ and } w_i < w_{max}, \\ \frac{d}{dt} w_i = -C_{drift} & \text{if } w_i < \theta_w \text{ and } w_i > w_{min}. \end{cases} \tag{2.11}$$

C_{drift} determines at which rate w_i is driven towards either the high state w_{max} or the low state w_{min}. When the synaptic weight is updated from w_{max} to w_{min} the synapse is said to be long-term depressed (LTD), and a change from w_{min} to w_{max} is called long-term potentiation (LTP). Finally, the binary synaptic weight of the internal variable w_i are defined by a threshold function, J_i, in order to produce an EPSC upon arrival of the pre-synaptic spike,

$$J_i = J_{max} f(w_i, \theta_J). \tag{2.12}$$

Here $f(x, \theta_J)$ is a threshold function with threshold θ_J, and J_{max} the maximum synaptic efficacy. The J_i is the actual synaptic weight, as the w_i can be any analog value between $[w_{min}, w_{max}]$. $f(x, \theta_J)$ can be either a sigmoidal or hard-threshold function, and if w_i is greater than θ_J the synaptic weight, J_i, is applied as J_{max}.

CHAPTER 3

Methods and hardware

This chapter describes how the SNN models are used as a pattern recognition and feature extraction system for processing of a condition-monitoring signal from a rotating machine. The tools and methods that made this possible, such as signal encoding systems, network models, software platform and neuromorphic processor are introduced in this chapter.

The data used in the project are obtained from the Ball Bearing Data Center at Case Western Reserve University and are recorded from a vibration sensor mounted on a rotating machine. With a sampling frequency of $12\,kHz$ and a rotational speed of about 1800 rpm. The machine has two possible states, were one state represents a healthy bearing and the other a faulty bearing. In the faulty state, there is a 0.007" indent in the outer race of the bearing. Twelve different data files are considered, where four of these are measurements with a healthy bearing at different loads on the machine, and the eight remaining files correspond to the faulty bearing with different loads on the machine.

The first step is to create an SNN that is capable of classifying the healthy and faulty states of the machine, thereby making it possible to detect when the machine transitions from the healthy to the faulty state. By supervised learning, the SNN should be able of learning to classify the two different states, and will show this by activating different populations in the layer of neurons which act as an output. In the output layer will half of the neurons activate for the healthy state, and the other half for the faulty state.

A major challenge is that the measured vibration signals have to be pre-processed in order to be used as input to the SNN. Furthermore, the ROLLS chip uses peripheral I/O logic circuits for receiving inputs and transmitting outputs using the Address-Event Representation (AER) communication protocol. Therefore, the vibration signal somehow needs to be transformed to a spike-based code that is compatible with an SNN and the AER protocol. The pre-processing has to be done in parallel to the processing of the neural network, while being power efficient enough to allow the system to function in a wireless sensor, but without losing the information needed for the classification to work reliably.

3.1 Pre-processing of input signal

Presented in this section are the pre-processing methods used to represent the vibration signal as spikes in the form of asynchronous AER events. Three types of pre-processing methods are used in this project. The methods are based on a Learned dictionary, a Spike rate coder and a Neuromorphic cochlea, respectively.

3.1.1 Learned dictionary

The dictionary learning pre-processing model is based on a machine learning algorithm known as sparse coding with adaptive matching pursuit. The objective of the system is to learn a finite set of statistically independent features from a signal and store them in a dictionary. Each extracted feature in the dictionary, called an *atom*, is initialised as Gaussian noise and are subsequently learned to maximize the expectation of the log data probability [32]. The column named 'Atoms' in Fig. 3.1 represent the features after learning a dictionary from the vibration signals considered here. The shape of the atoms reminds of superposed sinusoids or a wavelet function, and some of the atoms are impulse-like waveforms. The impulse-like atoms are learned from the faulty signals [32], where the defect in the bearing creates a metal to metal contact that excites brief transient vibrations in the system. The learned atoms can also be used to encode the signal into asynchronous digital events. For each event shown in the atom channel vs. time plot in Fig. 3.1, an atom have been subtracted from the input signal. It is possible to reconstruct the input signal by knowing the event times for each channel and the shapes of the atoms.

The dictionary learning method is useful for classification tasks involving vibration signals from bearings with different faults [32]. However, the algorithm is iterative and require a high-end computers or FPGAs. Therefore, using this approach it is challenging to meet the criteria of being sufficiently power efficient for online processing in a wireless module powered by a battery.

In this project, I will interpret the sparse events encoded using the learned atoms as a spike based input pattern, which approximates the temporal structure of the signal. The Dictionary learning is used to test on how well an SNN can learn to classify the patterns in the condition monitoring signals.

3.1.2 Spike rate coder

The second pre-processing method considered here is a neural recording system known as an A/D asynchronous delta modulator [33]. This module converts recorded membrane potentials to an asynchronous stream of spikes, which are encoded using AER. In the original work spikes are transmitted to a low-power spiking neural network chip with learning capabilities for decoding and classification online [33]. In contrast to the dictionary learning algorithm and neuromorphic cochlea, the delta modulator/spike rate

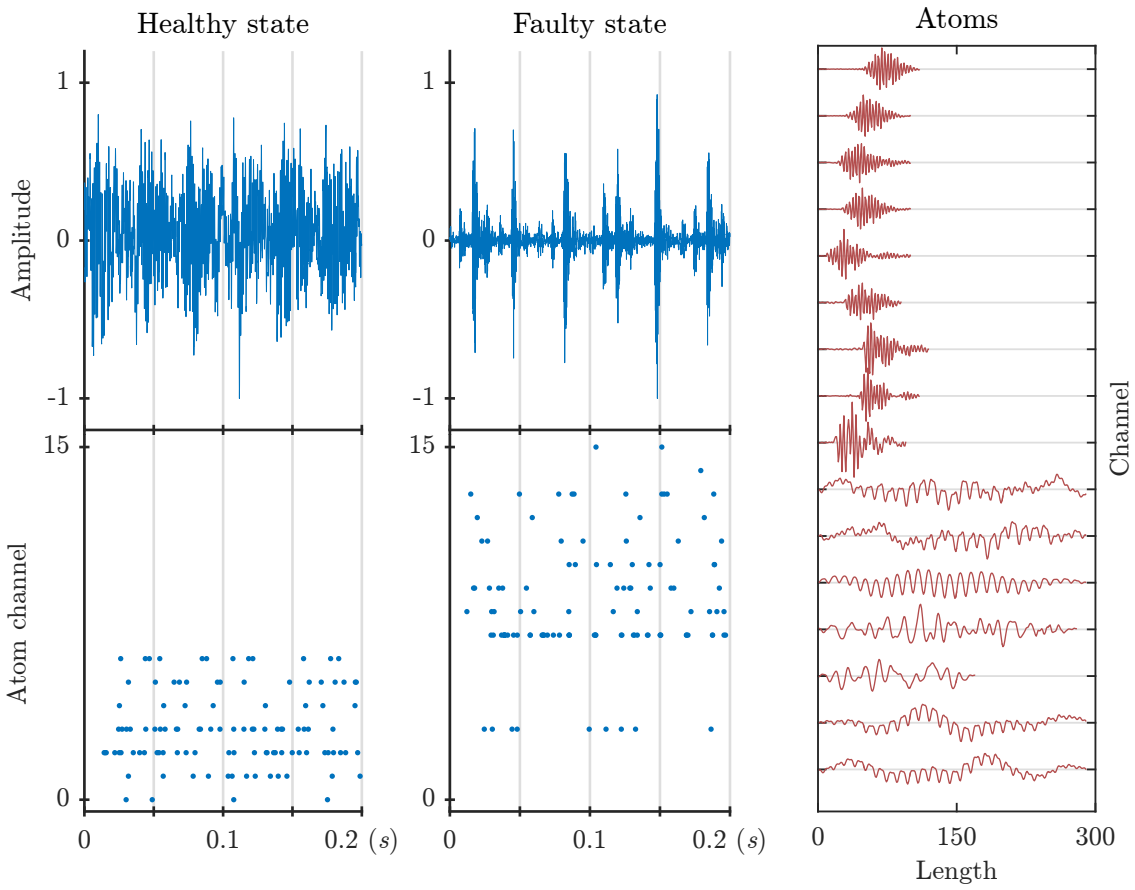


Figure 3.1: Samples of the vibrational signal amplitudes (top) and resulting spike-based code with determined dictionary learning (bottom). In the healthy state of the machine atoms 0-7 are typically activated, while for the faulty state atoms 8-15 are activated with the exception of the 3rd atom. The panel on the right-hand side shows the features extracted with the dictionary learning algorithm, the so-called atoms.

encoder is less complex and comparable to a frequency modulator.

The principle of the Spike rate coder is shown in algorithm, 1. It takes two adjacent samples from the input signal and encodes them in two spike-lists called UP and DN. The spike rates of the lists are proportional to the parameter δ divided by the slope of the line between the two data samples. The δ parameter is fitted to each input file, to emulate homeostasis (invariant spike rate) between the healthy and faulty states of the vibrational signal. Since N, the number of spikes is a finite number the *rest* variable cancel the time difference between the last spike and the data sample *x(i)*, otherwise the reconstructed signal would be biased. By knowing the δ parameter and spike-timings for the UP and DN lists, it is possible to reconstruct the input signal by adding δ for each

spike in the UP list and subtract δ for each spike in the DN list.

Algorithm 1: Delta modulater algorithm that generates UP and DN spikes.

Data: samples, $x(i)$, from vibrational signal
while *more samples to process* **do**
  $slope = \frac{x(i)-x(i-1)}{t_{samp}}$;
  $rate = \delta/slope$;
  $N = floor(\frac{x(i)-rest}{\delta})$;
  if $slope > 0$ **then**
    generate spike list UP(length=N, dt =$rate$);
  if $slope < 0$ **then**
    generate spike list DN(length=N, dt=$rate$);
  $rest = \delta \cdot N - \frac{x(i-1)-rest}{slope}$

The spike rate coder have potential for being implemented in a real system, where the encoding of the input signal is done in a low-power chip or FPGA because it is computationally lightweight. However, a signal with high dynamic range would require an adaptive δ parameter in order to maintain a specific level of spiking activity.

3.1.3 Neuromorphic cochlea

The third kind of pre-processing system used here is the AER-EAR system, an implementation of a spiked-based VLSI (very-large-scale integration) processor. The AER-EAR is an auditory system with two independent microphones, or "ears", that can record data in real-time to encode asynchronous spikes in a similar form to that of the spiral ganglion cells, or the cochlea [34].

The neuromorphic cochlea has 64 cascaded channels for each ear, and each channel responds to a different frequency band, from low to high pitches. The channels mimic the responses of cells in the human cochlea, which also respond to different frequencies and preforms transduction between the acoustic input and the neural signals that transmit the encoded information to the brain [35]. Within the neuromorphic cochlea, each 64 channels consist of a second-order section filter, where each differential readout drives the half-wave rectifiers which in turn drives 4 pulse-frequency modulators, shown in Fig. 3.2. The modulators are implemented as integrate-and-fire models with individual spiking thresholds (VT1-VT4), this allows for volume coding, transmission of events asynchronously and a temporal resolution of order microseconds.

The power consumption of the cochlea system is from 18.4 mW to 26 mW (DVdd) [34]. Therefore it is an interesting basis for further development of wireless sensor systems, for

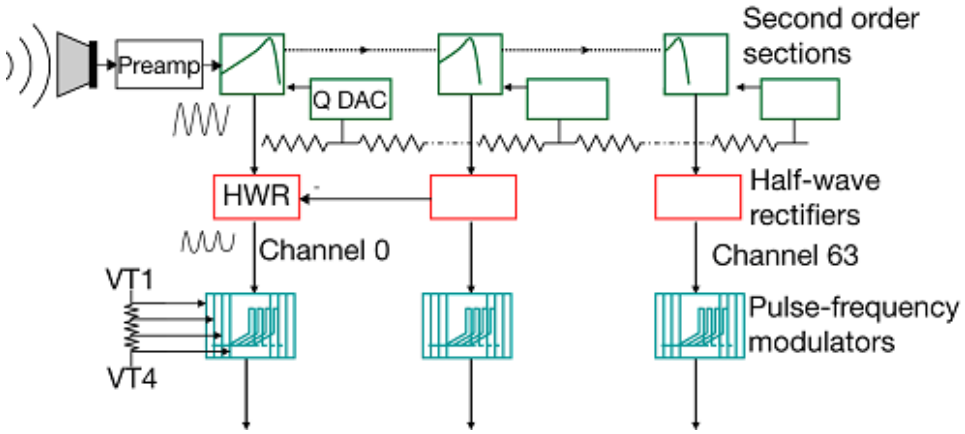


Figure 3.2: Schematics of the AER-EAR system [34].

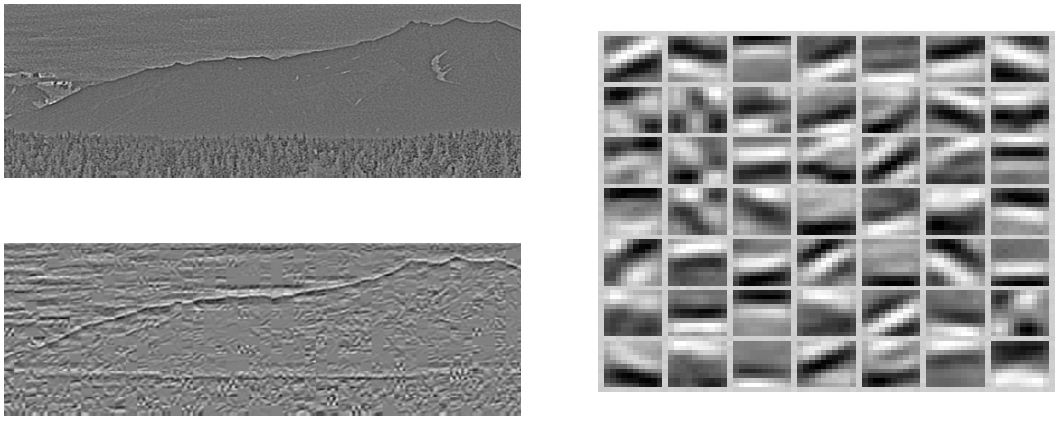
example for condition monitoring and feature learning.

3.2 Analysis of learned features

I aim to explore possibilities to learn features from acoustic/vibration signals using spiking neural networks. In previous work, a method known as spike-triggered average (STA) is used in order to determine response properties of neurons that are stimulated, for example by whitened images of natural scenes [36, 37]. By recording the spike activity of the output neurons and mapping the spike times to the time-varying spatial stimuli, STA's are formed by averaging over all spatial stimuli coinciding with a neurons spiking activity. The STA of a neuron gives an estimation of the linear receptive field, i.e., what stimuli trigger the neuron.

The E-I Net described in [36], have a network model of 100 input neurons that are connected to an output layer of 400 neurons and a hidden layer of 49 neurons. The weights between the input and output layer (400×100 weights) are learning according to the Oja's rule, which is a variant of the Hebbian learning rule and aims to make the output neurons compute for the principle components of the input stimuli. At equilibrium, i.e., the weights are fully learned, will the weights between the input layer and each output neuron (1×100 weights) be proportional to its own STA. Fig. 3.3a shows an illustration of the E-I Net's capability of reconstructing the input stimuli. The reconstruction is done for every time step during the simulation and is the sum of the active output neurons corresponding weights to the input layer.

The STA's learned from natural images shows great resemblance to Gabor filters, which are linear filters used for edge detection. Fig. 3.3b illustrates the STA's of the E-I Net. The fact that Gabor filters have also been found in the primary visual cortex (V1) of macaque [38] indicates that the model captures some important properties of the visual system. Gabor filters can be represented with orientation and frequency, so with a set



(a) (b)

Figure 3.3: (a) Top panel shows a whitened image of a natural scene of size $500x200$ pixels. The image is divided into $10x10$ pixel image patches and each patch acts as a stimulus to the network for a brief moment of time. A reconstructed image composed of reconstructed image patches can be seen in the lower panel. (b) Examples of STA's represented as $10x10$ pixel image patches.

of Gabor filters with different frequencies and orientations useful features can be derived from an image. Assuming that the cortical networks that encode auditory signals use a similar mechanism to extract features, STA's could be a useful tool for feature analysis and are expected to give shapes similar to the *atoms* obtained with dictionary learning.

3.2.1 Spike-triggered average

I investigate if the spike-triggered average is a useful tool for analysing features learned from vibration data. The calculations are made after the neural network has learned to classify the input signals, which results in STA's that depend on the two different states of the machine. The principle of how this works is shown in Fig. 3.4, in terms of a general network model and encoding system. First, the vibration signal is fed as the stimuli to the encoder, which sends AER's to the network model. The AERs are interpreted as spikes by the input layer and will propagate further through the SNN. The method that I use is to let stimuli activate the network for a given amount of time and then match the spike timings in the output layer to the closest samples in the processed stimuli. Because of the time delay of spikes that propagate through neural and encoder systems, and to capture the relevant dynamical behavior of the machine, the STA's are calculated from a window of 800 *samples*, which correspond to about 2 revolutions of the machine.

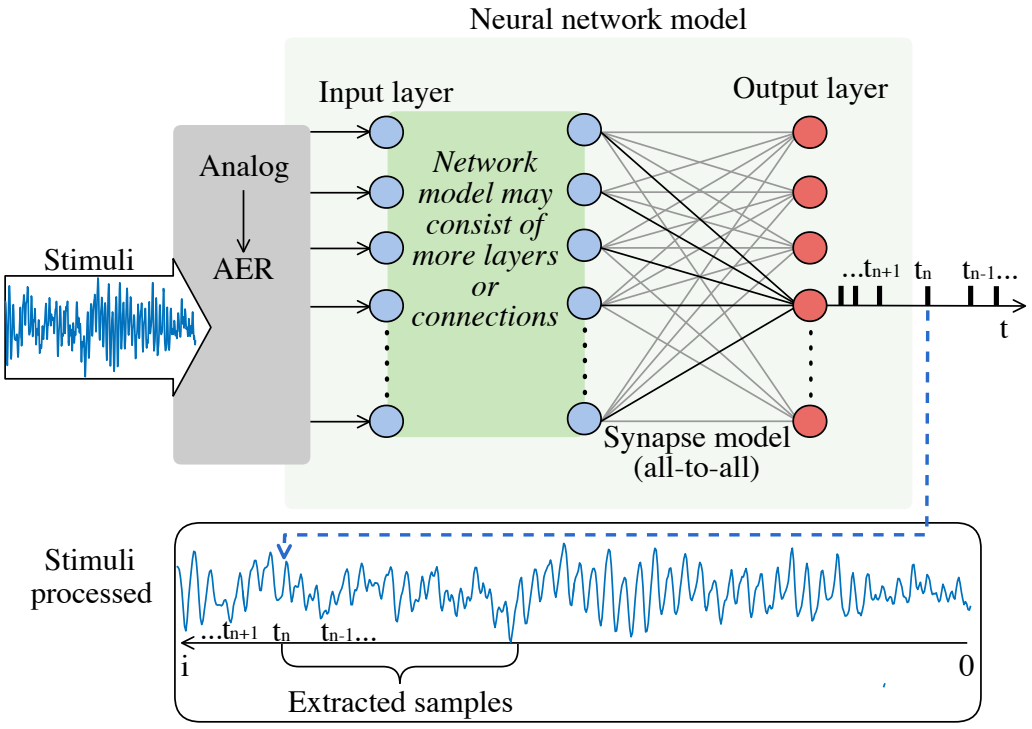


Figure 3.4: The stimuli is processed by the encoding system and sends the signal as AER to the neural network model. Events are interpreted as spikes by the input layer and send them to whichever network model used. The spikes are classified by the synapse model and output layer, where spike timings of the neurons are recorded. STA's are made by matching the recorded spikes with closest time step in the processed stimuli, $i(t_n)$, and by taking an average over all the extracted samples.

3.3 Network model

The computational power of artificial neural networks comes from connecting numerous neurons in a network. Even though a single neuron can preform basic information processing tasks [39], the power to detect patterns and resolve non-linearities in data structures comes from the network. The network properties depend on the neurons transfer function, the network architecture, and the self-organising behavior, i.e., learning rule and teacher signal. With the help of sufficiently many interconnections and learning, neurons can adapt to a given problem and approximate the solution to some level of accuracy. In the last few decades many different designs for neural network models have been proposed.

In this project, I adapt a model that can learn to recognize two states given vibration signals as input. The model should be simple, the network should not be more complex

than necessary and it should be possible to implement in the 256-neuron ROLLS chip [4]. The classification problem is solved by a fully connected synapses model and supervised learning, that teaches the output layer, as illustrated in Fig. 3.4. In supervised learning, the goal is to predict one or more target values from one or more input variables, by guiding the output layer with teacher signals. This means that the teacher signals have to adapt the output neurons spike timings and spike frequencies so the conditions of the learning rule are satisfied.

The complexity in the models represented are dependent on the structure of the encoded signal and a network model for each encoding system had to be made. In Sec. 3.3.1 and 3.3.2 I show some of the more common network models that were tested during the project.

3.3.1 Hidden layer

The hidden layer(s) are common to use in network models used for pattern recognition. In general terms by setting one or more layers in between the input and output layer, which are the hidden ones, the model create an internal pattern that determines a solution to the problem. In theory, it is said that most functions could be approximated by using only one hidden layer [40].

For an initial experiment, a hidden layer model were interpret to classify the events from the dictionary learning as an encoding system, but not used to extract any features. That being said, interpretation of the algorithm being an encoding system is not entirely true, a correct term would be a filter bank; where the timing of each event triggered tells only the onset for a filter being matched to the input signal. Therefore, with the use of a hidden layer I create a more abstract level while maintaining temporal structure of the original input signal.

By randomly connect and weight non-plastic synapses, the 16 atom channels are spanned over a higher dimension of 128 neurons, illustrated in Fig. 3.5, where the hidden layer represents a level of unknown features. Spike timings within the layer are randomly superimposed features and could be seen as an arbitrary encoding system.

3.3.2 Reservoir

Another network model used is the echo state network, which is similar to a hidden layer but instead of having one perceptron layer create a sparsely recurrent pool of neurons, also refereed to as a reservoir. The main idea is to drive a random, large, fixed recurrent neural network with the encoder system, and let it exploit the analog dynamics of neurons and synapses to establish a nonlinear response to the input signal. The connectivity and weights within this network are fixed and randomly assigned at the start, and synapses between the reservoir and output layer will learn by supervision temporal and spatial correlations to the input patterns.

The network model was implemented in a previous work [33], where recorded bio-signals

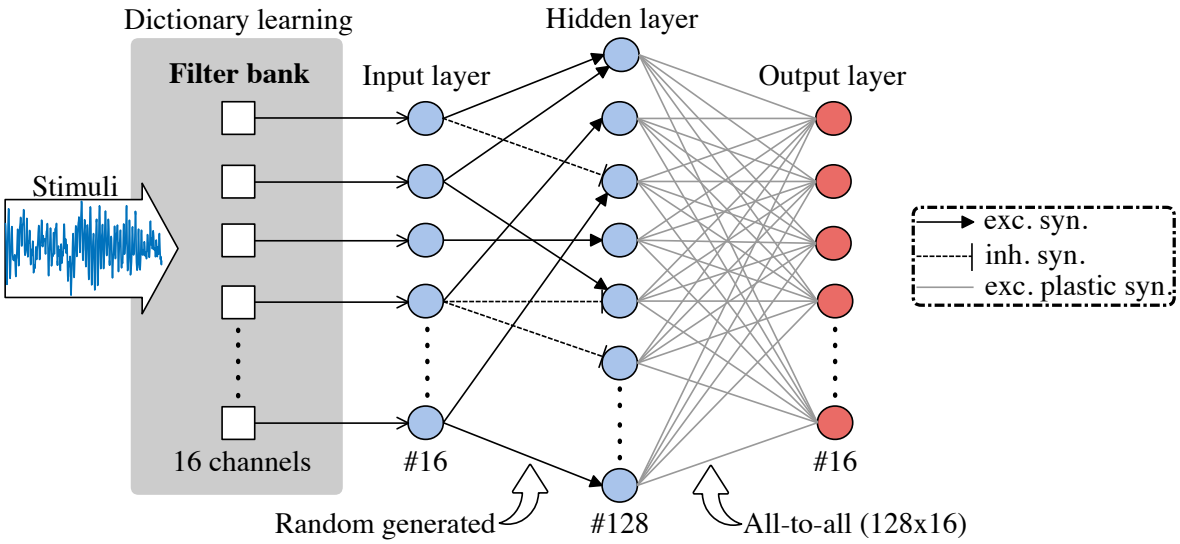


Figure 3.5: First experiment, stimuli encoded by the dictionary learning algorithm and feds events to the 16 input layer where they are interpreted as spikes. Random weighted and non-plastic synapses propagate spikes to the 128 neurons in the hidden layer. The plastic synapses are trained to make the 16 output neurons spike as desired.

are encoded by the A/D Asynchronous Delta Modulator that drives a recurrent layer to enhance temporal properties of the input patterns. For this project, I want to implement a 'simple' encoding system such as the spike rate coder to a reservoir of neurons, in order to generate a nonlinear internal pattern for the classifier to solve.

3.4 Training protocol

This section describes the procedure for the supervised learning. To ensure synapses learn the correct temporal properties from the input signal, the synaptic weight ω_i most be adequately trained, i.e supervised. Synapses will depend on the learning rules described in section 2.2.1 and external training signals are applied to the post-synaptic neurons (output layer) so their spiking patterns correlates to the correct pre-synaptic stimuli. The training signals are generated as two version, a teacher-true signal which drives neurons to fire at a high rate and a teacher-false signal for a low firing rate. The teacher signals are generated so that neurons internal calcium value will lie within the threshold intervals of θ_1, θ_2 or θ_3, as seen equation 2.7. In perspective of training signals, the output layer is divided into two populations, where the neurons labeled $1-8$ are supervised to correlate to the healthy state of the machine (the healthy group), and neurons $9-16$ learns to correlate to the faulty state of the machine (the faulty group). When a healthy state is present as an input, the true-teacher signal is applied to the healthy group, and

the faulty group receives the false-teacher signal. As synapses connecting to the healthy group will tend to potentiate, if driven by the true teacher signal, making ω_i transits to the high binary state, while synapses connecting to the faulty group will tend to make transitions to the low binary state, hence driven by the false teacher signal, and vice versa. In this way, the two subgroups can learn to associate their spiking patterns to unique input patterns of the machine.

The training session start with initializing all synaptic weights, ω_i, with random values between $[\omega_{min},\ \omega_{max}]$. Proceeding with creating input stimuli, generated as a subset of 1 second intervals where each set is extracted from a random vibration file and with a random offset. In parallel the two training signals are generated as subsets of 1 second intervals of Poisson-distributed spike trains, with a mean spiking frequency corresponding to $600Hz$ if true-teacher signal and $300Hz$ if false-teacher signal. This is done for the entire set of training stimuli, which runs for a specified amount of time. It is possible to test the classification during learning by remove the influence of training signals and increase the neurons calcium threshold θ_1, so the synaptic weights do not transit.

3.5 Software simulation

All spiking neural networks defined in this project were first constructed and tested in software simulations. Implementations are done with the programming language Python, using an open-source library called *BrianSimulator*, intended for simulating spiking neural networks [41, 42]. With the Brian toolbox, users can define neuron and synapse models in differential equations as standard mathematical notations and create network architectures by connecting neurons via synapses. The intent is to minimize developers time with a process that is easy to understand, reproducible, expressive and flexible. Designing it around equations also lets researchers independent of specific neuron or synapse models. This section gives a short description for Brian classes used and their corresponding parameters for creating the neural dynamics given in Ch. 2. All of the built-in Brian functions are italicized.

The input layer and training signals are generated with the *SpikeGeneratorGroup*, in order to create populations of cells emitting spikes at given times. For example, spike timings in the input layer are generated proportional to the registered events from the encoding system. With built-in parameters specified as the *number* of neurons within the group, a list of *indices* for the spiking cells with corresponding spike timing are needed.

The reservoir and perceptron layers are defined by the *NeuronGroup*. This class contains built-in parameters, such as the *model*, where implementation of the differential Eq. 2.5 - 2.7 are given and will govern the dynamical behavior of neurons under simulation. Other parameters set within this class are; *threshold*, which sets the minimum membrane magnitude, V_{thr}, for a neuron to produce a spike; *reset* defines neural variables that will adapt immediately after a neuron spike, such as setting the membrane potential of a neuron to I_{reset} and accumulate I_{wCa} to the internal calcium level I_{Ca}; *refractor*, tells the

minimum time period, T_{refrac}, a neuron is prevented from depolarizing following a spike. Both *threshold* and *reset* parameters are relative to the state variable used to describe the membrane intensity, in this case, they are compared to I_{mem}.

Creating the network architecture is done by the *Synapses* class. Here users define pre- and post-synaptic *NeuronGroup*'s and specify which neurons are to be connected. Synaptic dynamics are given in the *model* parameter, where conditions for the internal variables that are affected of pre- and post-synaptic spikes are defined. The plastic synapses use the internal variable ω_i which updates for every pre-synaptic spike according to the learning algorithm of Eq. 2.10 and 2.11. The contribution to the post-synaptic neuron is given as the Eq. in 2.8. The non-plastic synapses are set with static synaptic values with both excitatory and inhibitory connections. Excitatory weights are randomly generated between integers $1-3$ and inhibitory are either -2 or -1. After a presynaptic spike, the corresponding synapses injects an exponentially decaying current with respect to its weight to the neural state variable I_{in}.

With Brian it is possible to declare monitors for recording during simulation, such as *SpikeMonitor* and *StateMonitor*, if spike timings of given neurons or specified neural state variables are needed for further analysis.

3.6 Neuromorphic processor architecture

The ROLLS chip is fabricated using an 180 nm 1P6M Complementary Metal-Oxide-Semiconductor (CMOS) process that occupies an area of $51.4mm^2$, with approximately 12.2 million transistors and a power dissipation (at $30Hz$ and $1.8V$) of $4mW$ [43, 4]. The device comprises a row of 256×1 AdEx analog neurons and three arrays of different synapses; 256×256 short-term plasticity (STP) programmable synapses, 256×256 long-term plasticity (LTP) learning synapses and 256×2 linear integrator filters denoted as "virtual synapses". It also comprises additional peripheral analog/digital input and output circuits for receiving and transmitting AERs off-chip in real-time, and a "synapse de-multiplexer" static logic circuit [4]. The AER input circuits send the input spike events and also the chip configuration through a common input interface that uses a 21-bit address space. The AER output circuits use a four-phase handshaking scheme that receives addresses from an output bus, that represents an 8-bit address of each neuron which sends instantaneously as it spikes. The synapse de-multiplexer makes it possible to choose how the synapses connect to the neuron array. By default, the de-multiplexer switch-matrix is configured so that columns of 1×256 STP and LTP (1×512) synapses connects to each row in the neuron array. For example, changing the circuit control bits the user can allocate the previous synapse column to the following neuron and thereby sacrificing every other neuron, e.i., one operating neuron connects to an array of 2×512 synapses and 128 neurons are unused.

Configuring the neuromorphic processor is supported by a Python-based software front-end, called PyNCS. With PyNCS the user can design network architectures, setting

the global tuneable parameters, send and read AERs on- and off-chip. The software makes it also possible for continuous online monitoring and interacting with the system during execution [4]. Designing the dynamical behaviors of the desired network model is made possible by 13 tuneable parameters. The parameters represent biases to specific transistors in each circuit, which are precisely set by an on-chip Bias Generator [4] that can be tuned through an interface with PyNCS. All of the circuit schematics for the neurons, synapses and learning mechanism can be found in the original article [4].

Sillicon neurons and synapses

All of the ROLLS synapses processes input spikes in real-time and the AdEx neurons transmit spikes immediately as they reach their voltage threshold. As the input data is processed instantaneously and no additional mechanisms for storing partial results in memory banks is the ROLLS chip operating with no virtualization of time. As a consequence, the time constants of neurons and synapses require being well-matched to the input patterns they are designed to processes.

The STP and LTP synapses converts input spikes into output currents with non-linear dynamics (due to their adaptation or learning features) and tuned correctly can perform the biological plausible EPSC. The STP synapses allow the user to program the synaptic weights as four possible values, by a 2-bit programmable latch and with an extra latch for telling excitatory or inhibitory connections. If the STP synapses are set in excitatory mode will additional circuits model Short-Term Depression (STD) dynamics, where the magnitude of the EPSC decrease as a function of input spikes, and will recover slowly as the input disappears. The LTP synapses use two circuits; the post-synaptic learning circuits, that evaluate the learning mechanisms for updating synaptic weights and "stop-learning" conditions, and also a combined analog/digital that implements the dynamics of the bi-stable weights and their synaptic efficacy. Additionally, the chip has an array of virtual synapses, that can model excitatory and inhibitory synapses with shared synaptic weights and time constants.

As an example I choose to show the neuron schematics for tutoring purpose; seen in Fig. 3.6 the transistors marked with an exclamation mark represent the tuneable parameters, that can be tuned to create a desired behavior of the AdEx model. The silicon neurons schematic is divided into five analog circuit blocks, each emulating the ion diffusion and cell membrane in real neurons, as described in Sec. 2.1. It is comprised of a NMDA block ($M_{N1,N2}$), which implements a voltage gating mechanisms of the NMDA channel. The LEAK DPI circuit (M_{L1-L7}) implements the leak conductance of the cell membrane. The AHP DPI circuit (M_{A1-A7}) is modeling the adaptation variable. The Na^+ ($M_{Na1-Na5}$) produce the voltage spike and the K^+ (M_{K1-K8}) to reset the neuron's cell membrane, C_M, and model the refractory period mechanism.

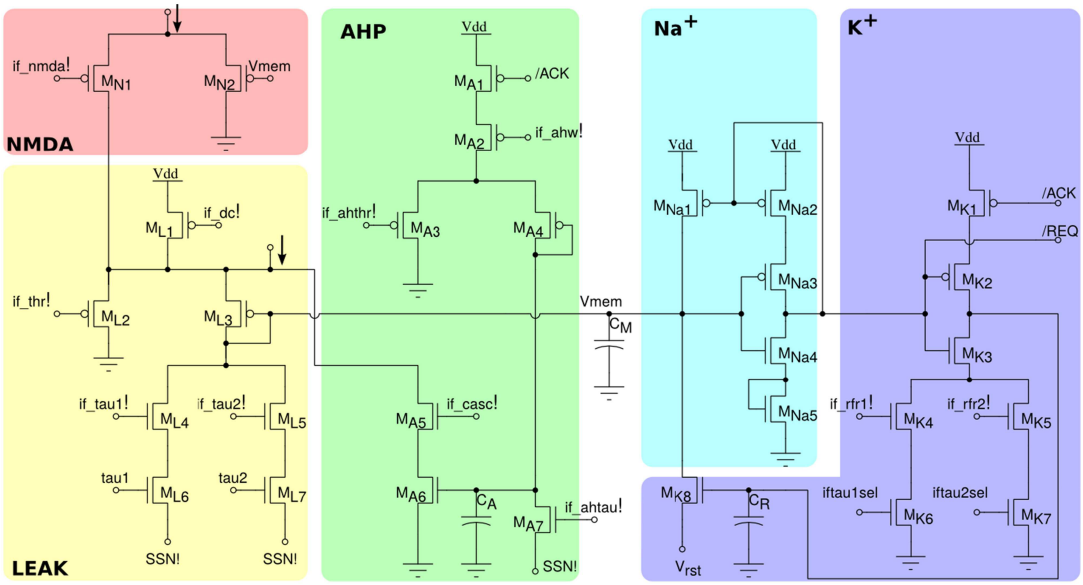


Figure 3.6: ROLLS sillicon neuron schematics [4] implementing a model of the AdEx neuron.

CHAPTER 4

Results

Results from the Brian simulations and classification experiments with the ROLLS chip are presented in the following sections. All models used the same learning procedure; where one-second intervals of stimuli are randomly selected from the vibration files and offset, over the total simulation time. The whole set of stimulus is then encoded and registered as AER's that the SNN input layer will transmit as spikes.

4.1 Numerical experiments

Parameters used in the Brian simulations are based on the biologically plausible values from [20]. However, the encoding systems require some parameters set exclusive to stabilize the training sessions, for the most part regarding the learning and teacher parameters, which is caused by each encoding system produce different spike rates for the input layer.

Most of the parameters shaping the neural and synaptic behaviors are listed in Tab. 4.1 and 4.2. All numerical results are relative to the fitting parameter, α. As the variable, α is used to weight each spike so that the output layer would spike in an appropriate range. During the learning α is set to a low value, 0.2, so that most of the output neurons spike according to the training signal. In the testing phase, the α is set to its high value and have been fitted to give a good classification.

Learning synapses are always initialized as random for the numerical experiments and are selected between $w_i \in [w_{min}, w_{max}]$ as seen in Fig. 4.2.

4.1.1 Dictionary based pre-processing

As explained in section 3.1.1, this model is a first experiment to evaluate the classification capability and to help understand the dynamics of the spiking neural network. The simulation started with transforming events from the dictionary learning to spikes to their corresponding neuron in the input layer.

Neuron		Adaptation		Calcium	
θ_{spk}	$20\,nA$	I_{wa}	$0\,pA$	I_{wCa}	$1\,pA$
I_{reset}	$0\,nA$	$I_{\tau a}$	$1pA$	τ_{Ca}	$200\,ms$
T_{refrac}	$10\,\mu s$	τ_{ahp}	$17.7ms$	θ_{max}	$15\,pA$
θ_{mem}	$9\,nA$			θ_3	$0.9 \cdot \theta_{max}$
I_τ	$1\,pA$			θ_2	$0.6 \cdot \theta_{max}$
I_{th}	$1\,pA$			θ_1	$0.1 \cdot \theta_{max}$

Table 4.1: Neuron parameters used in the software simulations.

Synapses		Plasticity		Teacher	
I_w	$\alpha \frac{32}{N_{pre}}\,nA$	w_{min}	$0\,V$	I_{wtea}	$30\,nA$
τ_{syn}	$20\,ms$	w_{max}	$1.8\,V$	τ_{tea}	$1\,ms$
		θ_J	$0.5\,X_w + w_{min}$		
		Δw^+	$0.02\,X_w$		
		Δw^-	$-0.02\,X_w$		
		C_{drift}	$0.3\,w_{max}\,V/s$		

Table 4.2: Synaptic parameters used in the software simulations. The synaptic efficacy, I_w, is normalized to the number of pre-synaptic neurons, N_{pre}, and a fitting parameter α is used to stabilize each encoding system. X_w is the relative difference between weight boundaries $(w_{max} - w_{min})$.

The network connections between the hidden and input layers are randomly generated of non-plastic synapses with random weights, in order to create the 'arbitrary' encoder with the spatial and temporal spiking patterns correlating to the machines two states. Connections are generated in an algorithm, that takes the probability of how many synapses should be excitatory, p_E, and inhibitory, p_I. The total amount of connected synapses are equal to $(p_E + p_I) \cdot (N_{in} \cdot N_{hidden})$. For this system, multiple varieties of p_E and p_I have been tested.

The following results are based on the following conditions; $p_E = 0.4\%$ and random weights of $w \in [1, 3]$, with a standard deviation of 0.2 and $p_I = 0.2\%$ with $w = -1$, also a standard deviation of 0.2. The system learned during a session of 600 batches of $1\,s$ stimuli and after each $20\,s$ the states of the synaptic weights are to be tested. While testing, the parametric value of θ_1 is set much higher so that no transition in w_i will occur and therefore the system will not learn during this time. The testing stimuli are a set of $40\,s$, divided in $20\,s$ intervals of the different states of the machine. Spike timings of the output layer are recorded during each testing phase.

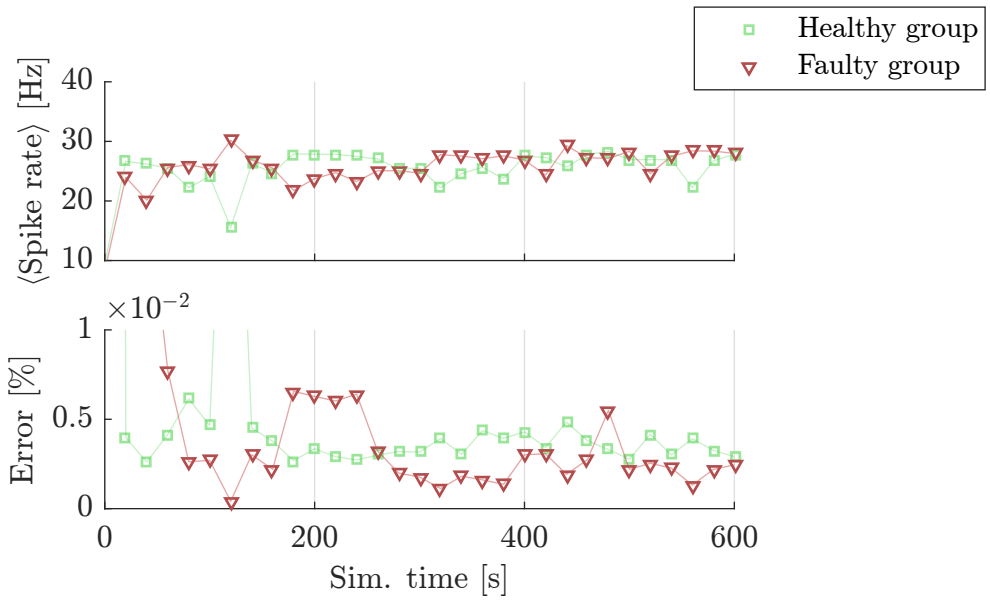


Figure 4.1: Convergence of the output layer during 600 s learning. Calculated every 20 s (square and triangle) with a separated batch of 40 s stimuli, with no learning. Top panel shows the mean spike frequency of the 8 neurons in the healthy group (square) and the faulty group (triangle) during the test phase. The lower panel shows the classification error during the test phase.

The results of the classification can be seen in Fig. 4.1. Given as the mean spike frequency of the 8 neurons in each group (top panel) and with their corresponding error (bottom panel), which is calculated as the proportion of neurons activity during the wrong state is present. An illustration on how the synaptic weight matrix evolves during learning is shown in Fig. 4.2, where clusters are forming according to the spatial patterns in the hidden layer, seen in Fig. 4.3.

The system shows to learn the classification with an error $< 1 \cdot 10^{-2}$ under the first 100 s. However, this is heavily depending on the spatial and temporal patterns generated in the hidden layer. For example, changing the connection probabilities to $p_E = 0.5\%$ and $p_I = 0.1\%$ would make the learning oscillate as illustrated in Fig. 4.4, and seemingly would not converge as good as the pervious system.

4.1.2 Spike rate coder

In this section are some results for the Spike rate coder presented. The spike rate coder showed to be a potential pre-processing system to the vibrational signal. However with limited amount of time were the spiking reservoir model never stabilzed, and instead I will only show the results of the Spike rate coder.

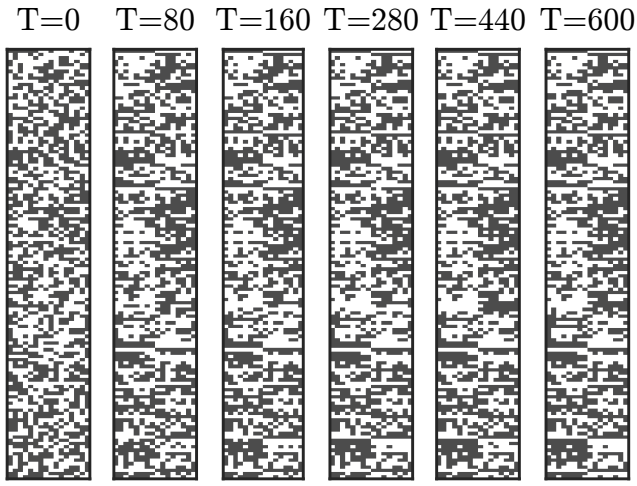


Figure 4.2: States of the synapse weight matrix between 128 hidden neurons (row's) and 16 output neurons (column's) during different time periods while the system learns. Black boxes indicates w_{max} and white boxes w_{min}.

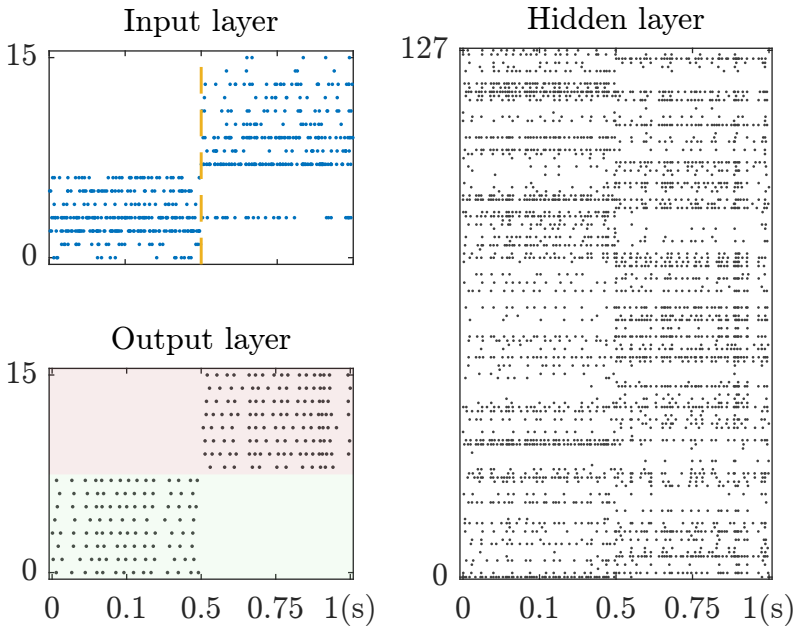


Figure 4.3: Raster plots of layers in the dictionary based model. Spiking patterns between 0-0.5 s are from a healthy input signal and 0.5-1 s from a faulty input signal. Recordings are done after 600 s of learning.

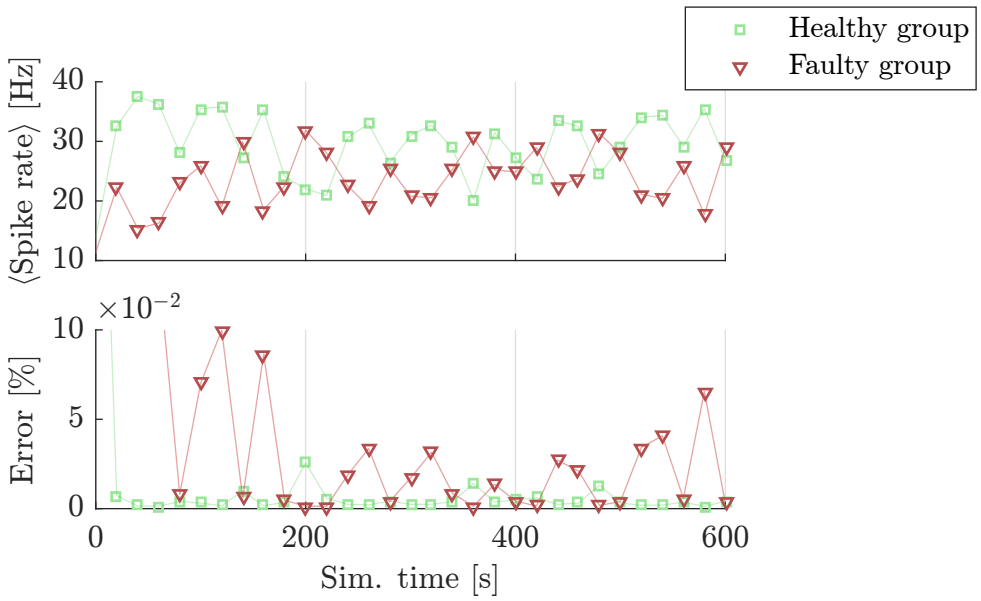


Figure 4.4: Convergence of the output layer during 600 s learning. Calculated every 20 s (square and triangle) with a separated batch of 40 s stimuli, with no learning. Top panel shows the mean spike frequency of the 8 neurons in the healthy group (square) and the faulty group (triangle) during the test phase. The lower panel shows the classification error during the test phase.

The δ, in algorithm 1, is manually chosen for each vibrational file so that the spike frequency of the UP and DN lists are consistent for every input signal. In this experiment the δ's are also picked rather small, in order to present as much information as possible. The difference in what information are lost due to the Spike rate coder can be seen between the reconstructed and original signals in Fig. 4.5. Both of the UP and DN spike lists lies at mean firing frequency of $6000\,Hz$ and this is consistent for all vibrational files.

With very high spiking frequencies, the spikes are transferred to a hidden layer of 128 neurons. The layer would act as an input to the reservoir, with similar properties as the 4 pulse-frequency modulators in the cochlea, with neurons that have different firing thresholds and refractory period. This is to make sure that neurons couple to the reservoir spikes asynchronous and with mean firing rates at $100\,Hz$. The connections between the UP and DN cells to the hidden layer are generated randomly; with probability of $p_E = 15\%$ excitatory synapses, with non-plastic static weights ranging $(1, 3)$, and $p_I = 5\%$ inhibitory synapses with a non-plastic weight of -1. An example of spiking patterns generated within this layer can be seen in Fig. 4.6, and seemingly the layer gives temporal spiking pattern that depends on the state of the signal.

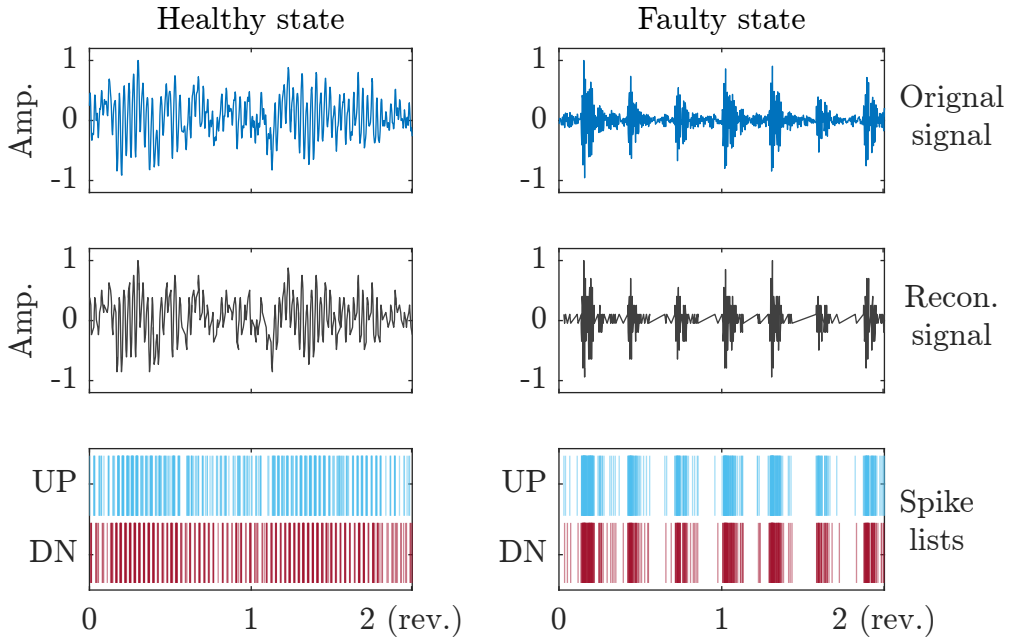


Figure 4.5: Illustration of UP and DN spike list generation from the vibrational signal (orginal). Shown as two revolutions of the machine for each state. The δ's used in this example are $2.5 \cdot 10^{-2}\,V$ and $3.6 \cdot 10^{-1}\,V$, respectively.

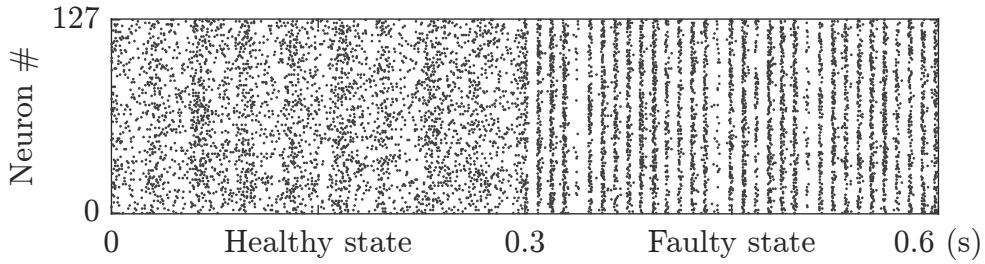


Figure 4.6: Raster plot the perceptron layer. A healthy state of the machine is presented during $0 - 0.3\,s$ and a faulty state during $0.3 - 0.6\,s$.

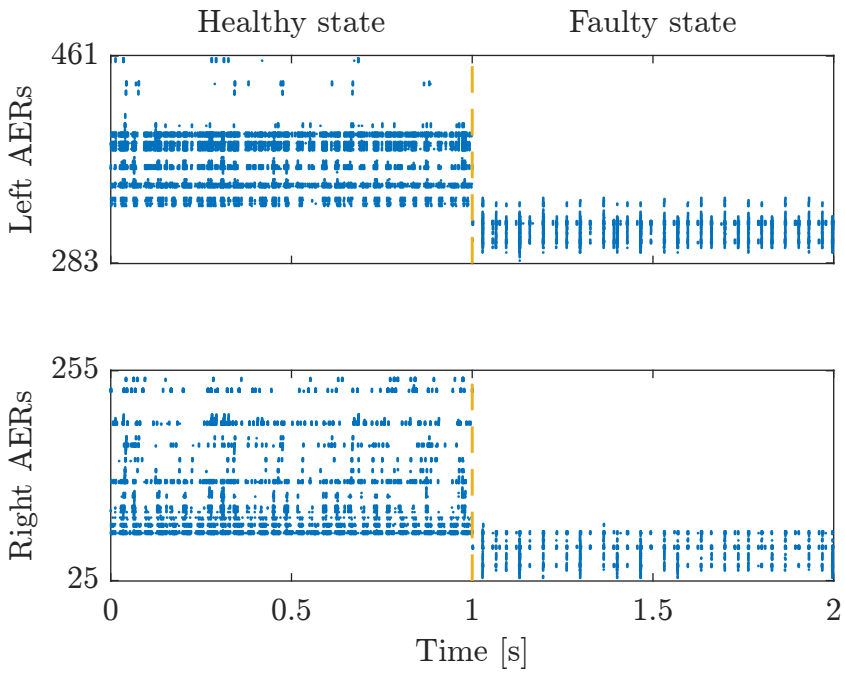


Figure 4.7: Raster plot of the neuromorphic cochlea AER's during 2 s (1 for each state). The addresses that is not shown are representing frequencies that the cochlea never encoded for.

The purpose of the reservoir were to extract an internal spatial pattern that correlates to the different states of the machine. However, the reservoir is a highly sensitive system and requires to tune the parameters with a high precision to make it work. Attempts were made to correlate spatial and temporal patterns of the UP and DN cells to the output neurons activity, but the overall complexity of the network model were the system not able to learn the classification task.

4.1.3 Cochlea based pre-processing

Results presented here are based on AER's produced by the neuromorphic cochlea, which acted as a pre-processing system for an SNN model. All coming results are based on this encoding system.

As explained in section 3.1.2, the cochlea encodes an input signal into AER's ranging from 0-511 addresses, where the first 255 addresses are from the left channels and the 256 to 511 are from the right channels. The channels could be understood as two different 'ears', where each ear encode the input signal to different frequency bands, from high to low pitches, which means that they could be taken as two separated systems.

The vibrational files were first encoded to AERs and each input signal showed on

differences in both spatial and temporal correlations to the two states of the machine. By simply observing the outcome, one can tell which file corresponded to a healthy or a faulty state, and to some extent what load on the machine were present. An example is shown in Fig. 4.7, where the spatial patterns, i.e., which neurons that are active versus stimuli, are close to being completely different. This effect is caused by the two states of the machine, which gives two very different vibration signals and the neuromorphic cochlea normalizing its input signal. As a consequent, the high amplitude peaks in the faulty signal that appears periodically will overthrow the information in between, which is vibrations of lower frequencies.

For the purpose of designing a condition monitoring system that can learn the vibration signals will the spatial patterns of the AERs depending on the states of the machine become an advantage. This was shown for the SNN model classifying the dictionary based pre-processing system; where a lower percentage of excitatory connections gave a more spatial difference between the states, which let to a better classification and stability during learning. Therefore, will the network model not rely on any hidden layers or a reservoir to extract extra features.

Learning of cochlea data

For simplicity, the SNN model was shaped so that it resembled the previous simulation, with 16 output neurons fully connected with learning synapses to an input layer of 512 neurons. The input layer will represent the output layer of the cochlea, where neurons will spike according to the AER's. As explained, the training procedure used randomly selected one-second batches from the AERs of the cochlea and with matching Poisson-distributed true- and false-teacher signals. The convergence of learning is shown in Fig. 4.8, with the neuronal and synaptic parameters from the Tab. 4.1 and 4.2, and an $\alpha = 1$.

4.2 Learned features

After SNN model have learned to differentiate the states, spiked-triggered averages are to be calculated for the 16 output neurons.

For the software simulation, STA is determined from the complete set of all input files, with a total length of ~ 235 seconds. It was generated to load all pre-processed input files in series, with a 1 second delay between each, and in parallel have their corresponding sets of vibrational data in same order. While the complete set of stimuli being processed by the SNN model, the spike timings for the output layer are recorded. After the whole set of data is classified, the recorded spike timings are backtracked to the vibrational data where the timings are matched to its closest sample. For each spike a bundle of samples are collected, with a length of 800 samples, starting at the closest sample to earlier processed ones. All STA's are initialized as zeros and averaged for every collected

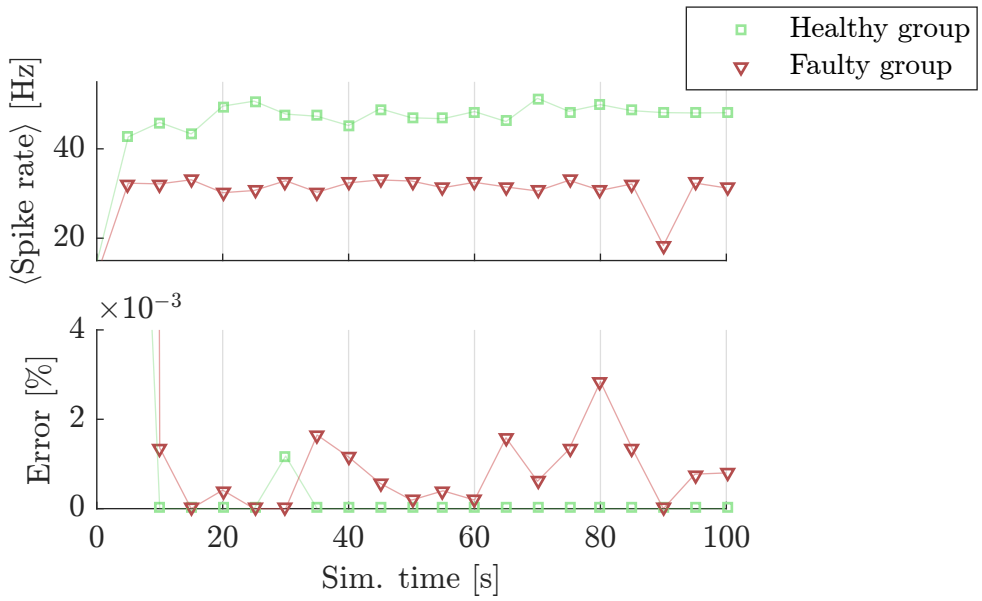


Figure 4.8: Convergence of the output layer during 100 s learning. Calculated every 5 s (square and triangle) with a separated batch of 40 s stimuli, with no learning. Top panel shows mean spike frequency of the 8 neurons in healthy and faulty group during the test phase. The lower panel illustrates the classification error during the test phase.

bundle to their corresponding neuron.

The calculated STA's are shown in Fig. 4.9, where the red boxes (R's) are formed by random spike timings from the healthy (left) or faulty (right) signals, with mean spike frequencies proportional to the healthy and faulty groups. They are generated to illustrate if neurons spike randomly would their STA's appear as a noise, but instead each STA appearers to be responding to their own specific frequencies. However, the output activity of the system is relative to the α variable. For higher α's would the output neurons generate a higher spike rate and the neurons would give different properties of the vibrational data, which can be seen in their STA's (Fig. 4.10). In Fig. 4.9, during the classification, the α variable were set to 1, which is proportional to the mean spike frequencies of 46.5 Hz for the healthy group and 36.4 Hz for the faulty group. The spiking frequencies with the $\alpha = 1$ are rather low, however, this was to match the classification done with the ROLLS chip.

As an example, if α were changed to 2, the system respond with mean frequencies of 118.5 Hz and 90.4 Hz, respectively. Interestingly the STA's formed for the α value showed on a more periodic appearance for the healthy neurons.

Another common method for study neuronal variability is to calculate the difference between spike timings, called the inter-spike interval (ISI). This was done for three dif-

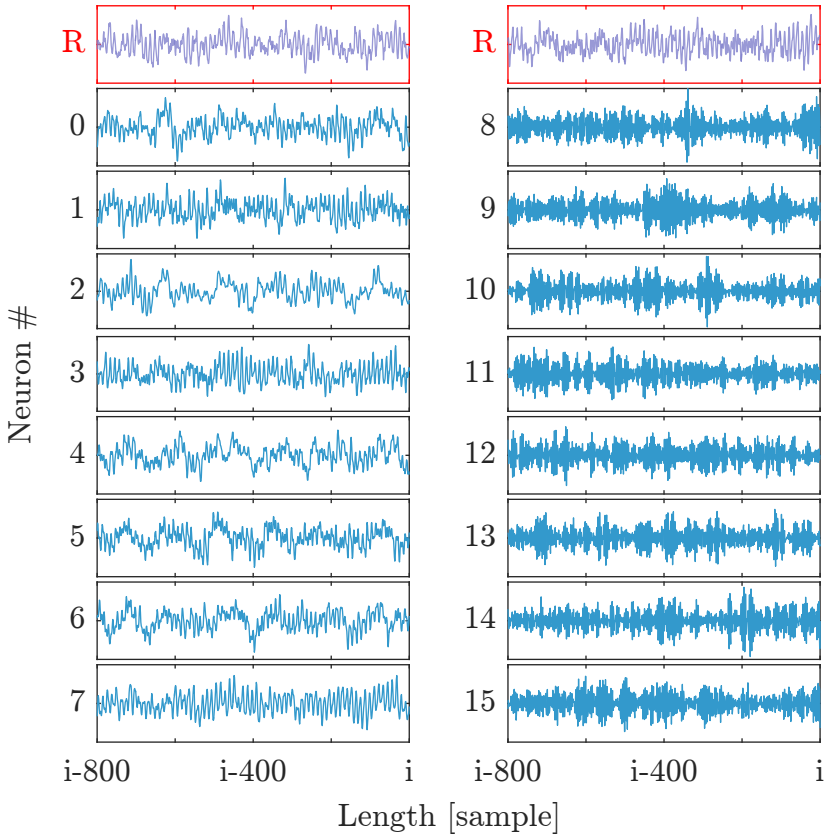


Figure 4.9: STA calculated from simulations, normalized to a standard deviation of 1. The red R's are STA's formed from randomly generated spike timings. The panels on the left-hand side are STA's formed for the healthy group and the STA's on the panels on the right-hand side are from the faulty group. The classification were done with $\alpha = 1$.

ferent simulations with the same learned synaptic state values, but while changing the α, illustrated in Fig. 4.11.

4.3 Neuromorphic hardware experiments

This section present the work done with the ROLLS chip, which is based on classifying the AER's from the neuromorphic cochlea.

In Tab. 4.3 are some of the biases that were set in the ROLLS to create the neural and synaptic behaviors to solve the classification.

The following system is set up by the framework pyNCS, where all necessary address and functions can be called. In order to solve the classification task with the AER's

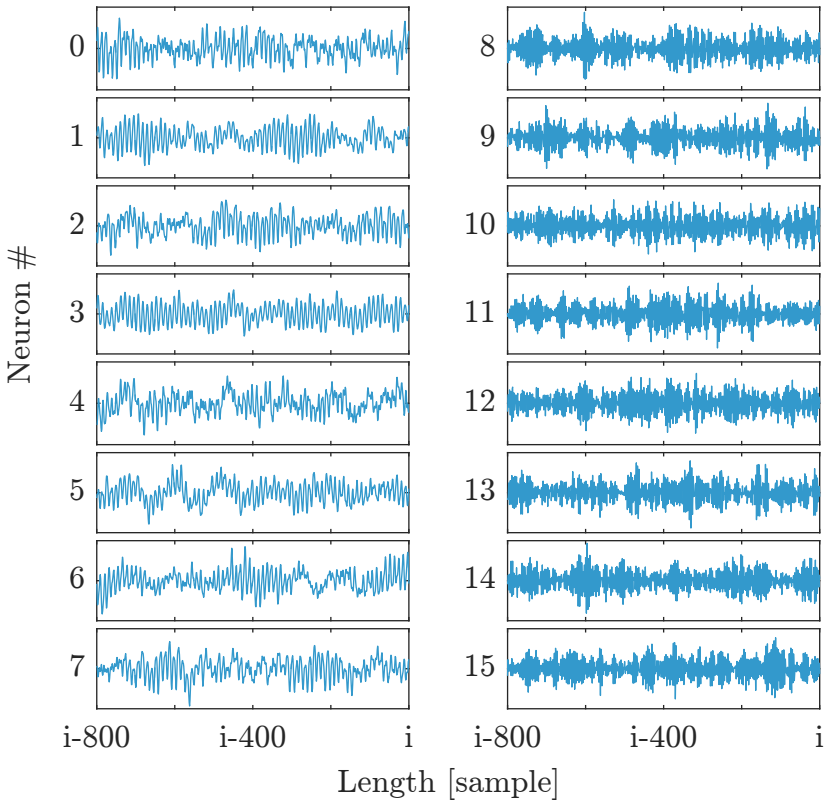


Figure 4.10: STA calculated from simulations, with $\alpha = 2$.

Neuron		Plasticity			
$rfr2$	$124\,pA$	$thdn$	$1.00\,pA$	$thmin$	$181\,pA$
$tau2$	$139\,pA$	$thup$	$9.92\,\mu A$	$whidn$	$33.6\,pA$
thr	$30.9\,nA$	$wdriftdn$	$3.96\,pA$	$deltaup$	$383\,pA$
$memthr$	$25.6\,nA$	$wdriftup$	$7.11\,pA$	$deltadn$	$105\,pA$

Table 4.3: Analog biases set to give ROLLS chip the neural and synaptic behaviours for solving the classification task.

from the neuromorphic cochlea, the 16 output neurons need 512 synaptic addresses each. Therefore, by sending appropriate bits to the chip it was possible to change the switch-matrix for the de-multiplexer, so that the configuration between neurons and synapses were set to 1×512. However, this setting made every other neuron unused and neuron addresses operating are the odd ones from 0 to 31.

Training procedure was made in two trails, this includes a training phase and a test

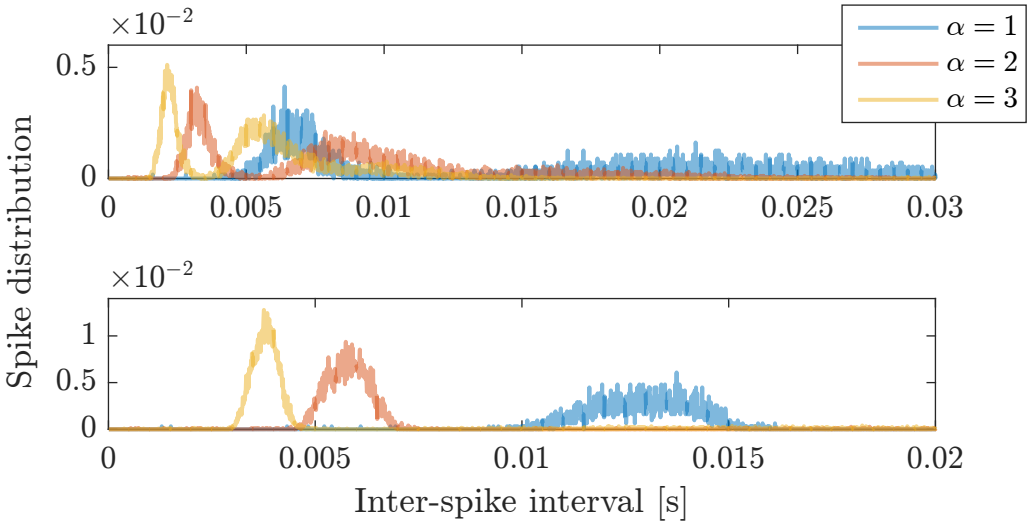


Figure 4.11: Inter-spike intervals for three values of α. The top panel show the averaged ISI for the healthy group and bottom panel are the neurons in the faulty group.

phase, where the system first is trained and afterward tested by a set of stimuli. The input stimuli of the trails are build as usual with a set of random picked $1\,s$ intervals, for a total length of $25\,s$. After stimuli are gathered the AER will be interpreted to the synaptic addresses; where events from the left channels are given to synapses that usually connects to the sacrificed neurons (even-numbered) and events from the right channels are fed to the synapse addresses for the operating neurons (odd-numbered).

Supervision was done with the training signals, which are applied to the neurons by the chips virtual synapses. The true- and false-teacher signals are generated by a function that gathers Poisson-distributed spike trains, with spike frequencies of $1000\,Hz$ if true and $100\,Hz$ if false. They are applied to the appropriate neuron groups for each set of picked stimuli and will stimulate during $900\,ms$ for each set.

Before training the biases listed in table 4.3 are set to neurons and learning synapses circuits to make the chip learn. For the testing phase the teacher signals are removed and some biases are changed, $thup = 1\,pA$, $thmin = 24.0\,\mu A$ and $whidn = 2.22\,nA$, which made sure that the bi-stable synaptic weights stay unchanged.

After both of the trails are done, the neuromorphic processor showed on learning and could classify the cochlea based pre-processed signal. The output patterns correlated to the different states of the machine, with mean spike frequencies of $68\,Hz$ and $63\,Hz$ respectively to the healthy and faulty group. As a results a 4 second window of the ROLLS chips output is shown in Fig. 4.12, with the corresponding input transmitted to the chip and some of the learned weights. The AER and learned weights are only from the synaptic addresses that connect to the neuron number one, but every neuron received

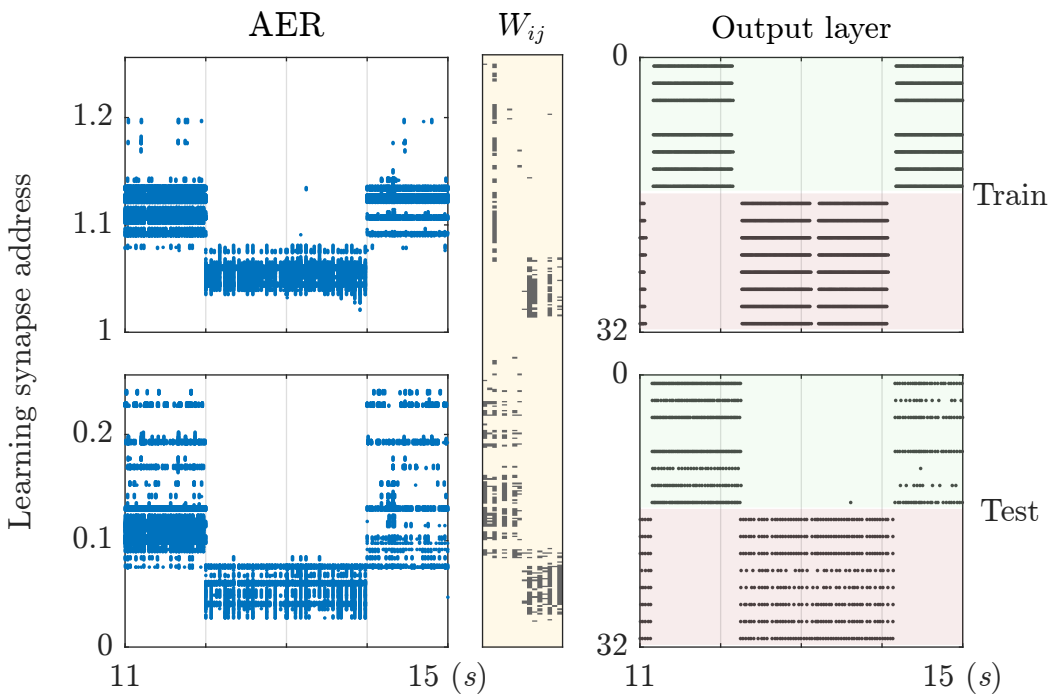


Figure 4.12: The input stimuli are given to the learning synapses and spikes according to the AER's, illustrated with synapses connecting to neuron 1. W_{ij} are the corresponding learned weights. The output layer shows the spiking patterns transmitted from the ROLLS chip. The train panel is while learning (influenced by the training signal) and the test panel is after learning with the same set of stimuli.

the same input.

As the ROLLS chip learned to classify the states of the machines, the last thing done was to calculate the STA's for the output neurons. The recorded spikes from the chip were matched to the input signal and as in the previous section a length of 800 samples are collected and averaged over all spikes. The resulting STA's created from the ROLLS, Fig. 4.13, were remarkable similar to the STA's from the simulations.

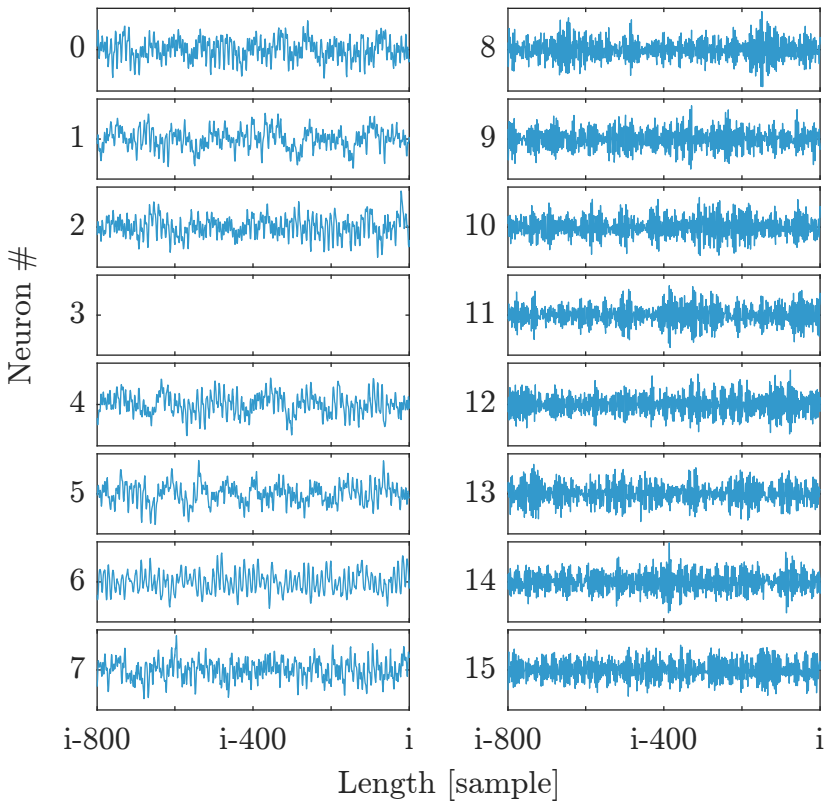


Figure 4.13: STA calculated from the output of the ROLLS chip. All STA's are normalized to a standard deviation of 1.

CHAPTER 5

Discussion

This thesis presents a study of pattern recognition with SNN models and the implementation of such networks in the neuromorphic processor ROLLS. Condition monitoring with vibration sensor data is considered as a potential use case of such technology that requires low-power wireless embedded systems that can process, learn and analyze high-frequency signals in real-time. My work is been based on previous studies investigating, for example, the network parameters and configuring the models to match properties of the nervous system. Biological neurons come in various types and display a range of variability in their behavior and adaptation depending on the functionality and the kind of signal they process [44]. Here the neural and adaptive parameters are tuned to produce plausible firing rates and time constants relative to the scale of the input signals.

Pre-processing systems and SNN models

The Learned dictionary encoder is used to test the dynamics of the combined STDP and calcium ion-dependent learning rule of the model and implemented in Brian only. The address events (AEs) calculated with the Learned dictionary are transmitted to an input layer of neurons, where each neuron is given its own address and spike once for each event. The spikes in the input layer (16 neurons) stimulate neurons in the hidden layer (128 neurons) through the non-learning synapses. Learning synapses are implemented between the hidden layer and the output layer and enable the classification. The idea with the network model is to generate a general encoding system, where the hidden layer represents features of the vibrational signal that appear in spatial and temporal patterns that the network learn to classify. Multiple sets of synapse weights, and probabilities of excitatory and inhibitory connections are tested. The results show that when generating synapses between input and hidden layer with the probability of excitatory synapses, $p_E = 40\%$, and inhibitory synapses, $p_I = 20\%$, the learning synapses reach their attractor states in less than $100\,s$ (Fig. 4.1). Changing the conditions to $p_E = 50\%$ and $p_I = 10\%$ reduced the stability of the learning process, and caused the classification error to oscillate during

the simulation (Fig. 4.4). For a higher p_E and a lower p_I would the neurons in the hidden layer increase their spike rates, this also made each neuron more likely to respond to both states of the machine. This is to be expected since there are more excitatory synapses, and the neurons are less influenced by inhibition, thus activating more neurons during each state. For this reason, the output neurons had to rely on learning the mean firing patterns of each neuron corresponding to the input stimuli, which showed to be harder for the network model to learn. Considering that the model learns from vibration files corresponding to multiple loads on the machine, where some of the loads produce different spike rates in the network model, can also be a factor for the instability during the training sessions. Even though the output neurons can learn from a range of spike rates, governed by the time constant and threshold levels of the calcium concentration (Eq. 2.10), it was hard to make the system learn all of the loads correctly. Simulating the network model is time-consuming, the parameters for the learning rule and synapses was adequately adapted each time the probabilities of synapses was changed, as it produce different firing rates in the system.

The spike rate coder produces spike patterns that correlate with the healthy and faulty states of the machine (Fig. 4.6), while offering a mathematical relatively simple algorithm. Additional mechanisms may be needed to adapt the δ variable to the signal derivative. In this project, the δ is manually fitted for each vibration file in order to simulate homeostasis independently of the states of the machine and the load. Since the spike rate coder encodes only local temporal information the idea is to use a reservoir of neurons to generate both spatial and temporal spiking patterns for further processing. Simulating an SNN reservoir model proved to be very challenging. Setting the spectral radius < 1 for the synaptic weights in the reservoir, as required for stability, sensitive to small changes in the input signal. As a consequence, the spike-frequency of the reservoir might change with several orders of magnitude while changing the synaptic efficacy, I_w, with small currents of $\pm 1\, pA$.

By mimicking the transfer function of the human ear the neuromorphic cochlea generates AERs from a vibration signal, which the SNN model implemented in the ROLLS can learn to classify. The events from the chip (512 possible addresses) encode information about the spatio-temporal structure of the input signals; encoding the vibrational signal gives spiking patterns correlating with the two different states of the machine (Fig. 4.7). As a side note, the AERs representing the faulty signal lacks information about some of the lower frequencies, due to the nature of the signal (with the high amplitude peaks) and the neuromorphic cochlea normalizing its input signal. The resulting spiking patterns have an evident spatial structure. Therefore, the network model is created without extra hidden layers of neurons as a single layer of 16 neurons, acting as an output and coupled with learning synapses to the cochlea. If the model is configured with suitable parameters the self-organizing behavior of the model enable learning of the classification task in both simulations and with the ROLLS chip (Fig. 4.8 and 4.12). Even though the network architecture is rather simple the output spikes integrate the high-dimensional spike

patterns from the cochlea through the non-linear dynamics implemented in the model. For this reason, I want to study what signal features individual neurons are activated by. I do that by calculating STAs of the input signal.

Evaluating learned features

The calculated STAs show that neurons adapt and correlate with different patterns in the input signal. Most STAs are shaped differently and are likely related to different signal sources in the system. In the Brian simulations, there was a need for initializing the synaptic weights with a standard deviation, otherwise, could the STAs converge to the exact same shapes. While for the ROLLS chip it happens by hardware mismatch and stochastic plasticity mechanism [4]. In order to show that neurons are not triggered randomly a set of STAs based on random (Poisson) spikes are also calculated (Fig. 4.9, labeled R). The random STAs are clearly different compared to the STAs of neurons corresponding to the faulty bearing ($8-15$). The shapes of STAs $8-15$ are influenced by the high-amplitude impulses in the faulty signals, which might be caused by the AERs from the neuromorphic cochlea correlating with these features. Neurons in the healthy group ($0-7$) generates STA's that are hard to distinguish from the random one (Fig. 4.9). Increasing α, i.e., the current generated by one spike results in higher spike-frequencies in the output layer and STAs with different shapes. Different values of α have been tested. Even though the neurons are stimulated with the same input signal, doubling the α generates a new set of STAs as illustrated in Fig. 4.9 and Fig. 4.10. With $\alpha = 2$, neurons encoding for the faulty signals have similar STAs as in the case of $\alpha = 1$, but in the healthy group, STAs will appear less "random" and are comparable to some of the atoms in the Learned dictionary (Fig. 3.1). For example, neuron 1 has an STA (Fig. 4.10) that is similar to a superposition of sinusoids.

The ISI may explain why the STAs with $\alpha = 1$ appears noise-like, as the spectrum of firing rates is broad, see Fig. 4.11. By increasing α the neurons are activated at one or multiple firing rates, which leads to the characteristic shapes of the STAs. The details of how such peaks in the ISI correlates with the input signal is yet to be understood. Increasing α further would eventually lead to neurons firing at their maximum rates, with ISIs equal to the refractory period.

Neuromorphic hardware

The results demonstrate that using neuromorphic hardware it is in principle possible to create a low-power self-organizing system for processing of high-resolution sensor signals sampled at $10^4\,Hz$, which for example is sufficient in many monitoring tasks involving vibration or audio sensors. The SNN are interesting mathematical models for tasks likes advanced pattern recognition, but are not as well-suited for logic and arithmetic tasks solved by conventional von Neumann processors. For example, simulations with Brian on my computer (using an Intel Core i7 at $2.5\,GHz$) for the same classification problem as

solved by the ROLLS required about 15 s for one second of signal time, while the power consumption of the PC is about 40 W. Scaling up the size of the network model would increase the number of operations and power consumption of the computer, which is not the case with the ROLLS chip. With the ROLLS the simulation occurs in real-time and the size of the network does not affect its performance in processing spikes. For comparison. Machine learning algorithms, like dictionary learning, can be implemented in low-power digital processors or FPGAs. The Xilinx Zynq-7000 or Adapteva Epiphany-IV are two examples of such computers that require about 1 W to operate, which is high if the goal is to integrate the processor in a resource-constrained system. Combining the neuromorphic cochlea with the ROLLS chip the resulting system can operate at a remarkably lower power of about 30 mW, which is more feasible when relying on a battery as power source.

To make the ROLLS chip solve the pattern recognition problem (Fig. 4.12) it was necessary to carefully configure the network parameters and biases in the chip. The chip offers biological plausible neural dynamics but setting the biases in the circuits correctly was both time-consuming and challenging for the first-time user. Given the limited amount of time I made some experiments and defined the biases by trail and error. The parameters of the silicon neurons were relatively straight forward to set and had similar time constants as the neurons in the Brian simulations. The most challenging part was to make the chip learn, in particular to set the three threshold levels required for the bi-stable weights to increase, decrease or to not be updated. Also, the drifting and stepping of weights were hard to realize with similar results as in the simulations. It was challenging to get the chip to learn the healthy and faulty stimuli at the same time, which seemed to be caused by the pulse-width of the 4-phase handshaking protocol, where one pulse-width would work fine for one set of stimuli and vice versa. The workaround I made was to initialize the bi-stable synaptic weights with their low-state (which are random in the simulations) so that the weights would only need to be potentiated during learning. I cannot say if this was the case for all of the synapses, as it is only possible to monitor one synapse at a time by an oscilloscope. The synaptic efficacy was set quite low, which might have prevented the third neuron from firing, see Fig. 4.13, and that is why no STA was calculated for that neuron. With a small synaptic efficacy, the ROLLS chip did learn to classify the two states of the machine and to do so with only two spikes miss placed in 50 s of classification.

5.1 Conclusion

I present a study of a spiking neural network model that emulate the biophysics of real spiking neurons and dynamic synapses to perform advanced pattern recognition with recorded sensor signals. The dynamic behavior of the model is defined by the ROLLS neuromorphic processor, which is a low-powered mixed-signal analog/digital VLSI device with self-organizing features, which was investigated, in order to build a low-power brain-

inspired sensor system.

In order to process information with spiking neurons, the high-frequency sensor signals need to be encoded with spikes. The AER-EAR system was used for that purpose and provided an address event representation that the ROLLS chip can learn to classify (if tuned correctly) at a combined power consumption of $\sim 30\,mW$.

I demonstrate spiking neural network architectures that successfully perform the condition monitoring problem given the recorded signals from a rotating machine with healthy and faulty bearings. I also demonstrate a method that can be used to extract the features learned by individual neurons from the input signal in both simulations and hardware.

5.2 Future work

As a continuation of the work presented here I would propose to run more tests with both simulations and the ROLLS neuromorphic processor. Further experiments with the ROLLS chip are needed to get the learning rule to work as intended with a higher synaptic efficacy. Additional analysis, in particular, cross-validation of the classification results, are required to understand how well the condition monitoring system perform with a new set of stimuli not available when the system learned. It would have been interesting to see the reservoir results, but this will probably require further fine-tuning of the parameters.

Further research of unsupervised network models are needed to investigate what kind of features that are learned and to correlate the learned features with known states of the system, like different kinds faults. If more information of the STA can be acknowledged and if it is a feasible way of illustrating individual neurons features from sensor signals in the range of kHz. For example, record a new set of vibrational data with a smaller defect in the bearing, to be pre-processed with the neuromorphic cochlea and do pattern recognition with the SNN investigated within the project.

References

[1] I. Tosic and P. Frossard, "Dictionary Learning, What is the right representation for my signal?," *Signal Processing Magazine, IEEE*, vol. 28, no. 2, pp. 27–38, 2011.

[2] R. P. W. Duin and E. Pękalska, "The science of pattern recognition. achievements and perspectives," *Studies in Computational Intelligence*, vol. 63, pp. 221–259, 2007.

[3] R. J. Douglas and K. A. C. Martin, "Recurrent neuronal circuits in the neocortex," 2007. *Current Biology*, DIO: `http://dx.doi.org/10.1016/j.cub.2007.04.024`.

[4] N. Qiao, H. Mostafa, F. Corradi, M. Osswald, F. Stefanini, D. Sumislawska, and G. Indiveri, "A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128K synapses," *Frontiers in Neuroscience*, vol. 9, no. APR, pp. 1–17, 2015.

[5] J. Koomey, S. Berard, M. Sanchez, and H. Wong, "Implications of historical trends in the electrical efficiency of computing," *IEEE Annals of the History of Computing*, vol. 33, pp. 46–54, March 2011.

[6] SING, "Embedded sensing systems." Available at `https://sing.stanford.edu/site/projects/3` (visited on Dec. 20, 2016).

[7] O. Vermesan and P. Friess, *Internet of Things: Converging Technologies for Smart Environments and Integrated Ecosystems*. River Publishers Series in Communications Series, River Publishers, 2013.

[8] R. Wang, G. Cohen, K. Stiefel, T. Hamilton, J. Tapson, and A. van Schaik, "An fpga implementation of a polychronous spiking neural network with delay adaptation," *Frontiers in Neuroscience*, vol. 7, p. 14, 2013.

[9] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana, "The spinnaker project," *Proceedings of the IEEE*, vol. 102, pp. 652–665, May 2014.

[10] R. Douglas, M. Mahowald, and C. Mead, "Neuromorphic analogue VLSI," *Annual review of neuroscience*, vol. 18, pp. 255–281, 1995.

[11] S.-C. Lui, J. Kramer, G. Indiveri, T. Delbruk, and R. Douglas, *Analog VLSI: Circuits and Principles*. Cambridge; London: MIT Press, 2002.

[12] W. Gerstner and W. M. Kistler, *Formal spiking neuron models*, pp. 93–146. Cambridge University Press, Aug 2002.

[13] A.-r. Mohamed, G. E. Dahl, and G. Hinton, "Acoustic Modeling Using Deep Belief Networks," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 14–22, 2012.

[14] D. Cireşan, U. Meier, and J. Schmidhuber, "Multi-column Deep Neural Networks for Image Classification," *International Conference of Pattern Recognition*, no. February, pp. 3642–3649, 2012.

[15] S.-C. Liu, T. Delbruck, G. Indiveri, A. Whatley, and R. Douglas, *Silicon Neurons*, pp. 153–183. John Wiley & Sons, Ltd, 2015.

[16] N. Brunel, V. Hakim, and M. J. Richardson, "Single neuron dynamics and computation," *Current Opinion in Neurobiology*, vol. 25, pp. 149 – 155, 2014. Theoretical and computational neuroscience.

[17] E. Chicca, F. Stefanini, C. Bartolozzi, and G. Indiveri, "Neuromorphic electronic circuits for building autonomous cognitive systems," *Proceedings of the IEEE*, vol. 102, no. 9, pp. 1367–1388, 2014.

[18] M. Mahowald and R. Douglas, "A silicon neuron," *Nature*, vol. 354, no. 6354, pp. 515–8, 1991.

[19] W. Gerstner and W. M. Kistler, *Detailed neuron models*, pp. 31–68. Cambridge University Press, Aug 2002.

[20] D. Ling, "Behavioral Simulations of Spike-based Learning VLSI Circuits," *Semester project, Institute of Neuroinformatics University of Zürich and ETH Zürich*, April 2015.

[21] H. Z. Shouval, G. C. Castellani, B. S. Blais, L. C. Yeung, and L. N. Cooper, "Converging evidence for a simplified biophysical model of synaptic plasticity," *Biological Cybernetics*, vol. 87, no. 5-6, pp. 383–391, 2002.

[22] K. P. Giese, N. B. Fedorov, R. K. Filipkowski, and A. J. Silva, "Autophosphorylation at Thr286 of the alpha calcium-calmodulin kinase II in LTP and learning," *Science (New York, N.Y.)*, vol. 279, no. 5352, pp. 870–873, 1998.

[23] S. Lowel and W. Singer, "Selection of intrinsic horizontal connections in the visual cortex by correlated neuronal activity," *Science*, vol. 255, no. 5041, pp. 209–212, 1992.

[24] K. M. Hynna and K. Boahen, "Thermodynamically equivalent silicon models of voltage-dependent ion channels," *Neural computation*, vol. 19, no. 2, pp. 327–350, 2007.

[25] E. Chicca, G. Indiveri, and R. Douglas, "An adaptive silicon synapse," *Proceedings of the 2003 International Symposium on Circuits and Systems (ISCAS)*, vol. 1, pp. I–82–84, 2003.

[26] S.-C. Liu, T. Delbruck, G. Indiveri, A. Whatley, and R. Douglas, *Silicon Synapses*, pp. 185–217. John Wiley & Sons, Ltd, 2015.

[27] R. G. M. Morris, "D.O. Hebb: The Organization of Behavior, Wiley: New York; 1949," 1999.

[28] A. P. Engelbrecht, *Computational Intelligence: An Introduction*. John Wiley & Sons, Ltd, 2007.

[29] McStrother, "A single-layer feedforward artificial neural network," 2010. Available at `https://en.wikipedia.org/wiki/Artificial_neural_network` (visited on Dec. 20, 2016).

[30] J. Lisman and N. Spruston, "Questions about STDP as a general model of synaptic plasticity," *Frontiers in Synaptic Neuroscience*, no. OCT, pp. 1–5, 2010.

[31] J. M. Brader, W. Senn, and S. Fusi, "Learning Real-World Stimuli in a Neural Network with Spike-Driven Synaptic Dynamics," *Neural Computation Massachusetts Institute of Technology*, vol. 19, pp. 2881–2912, 2007.

[32] S. del Campo, K. Albertsson, J. Nilsson, J. Eliasson, and F. Sandin, "Fpga prototype of machine learning analog-to-feature converter for event-based succinct representation of signals," in *Machine Learning for Signal Processing (MLSP), 2013 IEEE International Workshop on*, pp. 1–6, Sept 2013.

[33] F. Corradi and G. Indiveri, "A Neuromorphic Event-Based Neural Recording System for Smart Brain-Machine-Interfaces," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 9, no. 5, pp. 699–709, 2015.

[34] L. Shih-Chii, "Aer-ear cochlea system - asynchronous spike-based vlsi cochlea," 2010. Available at `http://aer-ear.ini.uzh.ch/dokuwiki-2010-11-07/doku.php` (visited on Oct. 20, 2016).

[35] S.-C. Liu, T. Delbruck, G. Indiveri, A. Whatley, and R. Douglas, *Silicon Cochleas*, pp. 71–90. John Wiley & Sons, Ltd, 2015.

[36] A. Ternstedt and O. Öberg, "Feature learning with a spiking neural network," *Project report, Luleå University of Technology*, 2016.

[37] P. D. King, J. Zylberberg, and M. R. DeWeese, "Inhibitory interneurons decorrelate excitatory cells to drive sparse code formation in a spiking model of V1," *The Journal of neuroscience : the official journal of the Society for Neuroscience*, vol. 33, no. 13, pp. 5475–85, 2013.

[38] D. L. Ringach, "Spatial structure and symmetry of simple-cell receptive fields in macaque primary visual cortex," *Journal of neurophysiology*, vol. 88, no. 1, pp. 455–63, 2002.

[39] S. Agatonovic-Kustrin and R. Beresford, "Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research," *Journal of Pharmaceutical and Biomedical Analysis*, vol. 22, no. 5, pp. 717–727, 2000.

[40] B. D. Ripley, "Pattern Recognition and Neural Networks," *Analysis*, no. 1995, p. 403, 1996.

[41] D. Goodman and R. Brette, "Brian: a simulator for spiking neural networks in python," *Frontiers in Neuroinformatics*, vol. 2, p. 5, 2008.

[42] C. Rossant, D. F. Goodman, B. Fontaine, J. Platkiewicz, A. Magnusson, and R. Brette, "Fitting neuron models to spike trains," *Frontiers in Neuroscience*, vol. 5, p. 9, 2011.

[43] G. Indiveri, F. Corradi, and N. Qiao, "Neuromorphic architectures for spiking deep neural networks," *Electron Devices Meeting (IEDM), 2015 IEEE International*, pp. 4.2.1–4.2.4, Dec 2015.

[44] D. Debanne, "Information processing in the axon," *Nature Reviews Neuroscience*, vol. 5, no. 4, pp. 304–316, 2004.