Monte Carlo Optimization of Neuromorphic Cricket Auditory Feature Detection Circuits in the Dynap-SE Processor

Mattias Nilsson

Engineering Physics and Electrical Engineering, master's level
2018

Luleå University of Technology
Department of Computer Science, Electrical and Space Engineering
Abstract

Neuromorphic information processing systems mimic the dynamics of neurons and synapses, and the architecture of biological nervous systems. By using a combination of sub-threshold analog circuits, and fast programmable digital circuits, spiking neural networks with co-localized memory and computation can be implemented, enabling more energy-efficient information processing than conventional von Neumann digital computers. When configuring such a spiking neural network, the variability caused by device mismatch of the analog electronic circuits must be managed and exploited. While pre-trained spiking neural networks have been approximated in neuromorphic processors in previous work, configuration methods and tools need to be developed that make efficient use of the high number of inhomogeneous analog neuron and synapse circuits in a systematic manner.

The aim of the work presented here is to investigate such automatic configuration methods, focusing in particular on Monte Carlo methods, and to develop software for training and configuration of the Dynap-SE neuromorphic processor, which is based on the Dynamic Neuromorphic Asynchronous Processor (DYNAP) architecture. A Monte Carlo optimization method enabling configuration of spiking neural networks on the Dynap-SE is developed and tested with the Metropolis-Hastings algorithm in the low-temperature limit. The method is based on a hardware-in-the-loop setup where a PC performs online optimization of a Dynap-SE, and the resulting system is tested by reproducing properties of small neural networks in the auditory system of field crickets. It is shown that the system successfully configures two different auditory neural networks, consisting of three and four neurons respectively. However, appropriate bias parameter values defining the dynamic properties of the analog neuron and synapse circuits must be manually defined prior to optimization, which is time consuming and should be included in the optimization protocol in future work.
# Contents

<table>
<thead>
<tr>
<th>Preface</th>
<th>vii</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1 Introduction</strong></td>
<td>1</td>
</tr>
<tr>
<td>1.1 Background</td>
<td>1</td>
</tr>
<tr>
<td>1.1.1 Artificial Neural Networks</td>
<td>2</td>
</tr>
<tr>
<td>1.1.2 Neuromorphic Hardware</td>
<td>3</td>
</tr>
<tr>
<td>1.1.3 Related Work</td>
<td>4</td>
</tr>
<tr>
<td>1.2 Motivation and Applications</td>
<td>5</td>
</tr>
<tr>
<td>1.3 Problem Description</td>
<td>6</td>
</tr>
<tr>
<td>1.3.1 Delimitations</td>
<td>6</td>
</tr>
<tr>
<td><strong>2 Theory</strong></td>
<td>9</td>
</tr>
<tr>
<td>2.1 Neuron and Synapse Models</td>
<td>9</td>
</tr>
<tr>
<td>2.1.1 Spiking Neuron Models</td>
<td>9</td>
</tr>
<tr>
<td>2.1.2 Synapse Model</td>
<td>12</td>
</tr>
<tr>
<td>2.1.3 Synaptic Plasticity and Learning</td>
<td>12</td>
</tr>
<tr>
<td>2.2 The Dynap-SE Neuromorphic Processor</td>
<td>13</td>
</tr>
<tr>
<td>2.2.1 Neuromorphic Processor Overview</td>
<td>13</td>
</tr>
<tr>
<td>2.2.2 Spike Routing Architecture</td>
<td>14</td>
</tr>
<tr>
<td>2.2.3 Neural Processing Nodes</td>
<td>14</td>
</tr>
<tr>
<td>2.2.4 Software</td>
<td>17</td>
</tr>
<tr>
<td>2.3 Acoustic Communication in Crickets</td>
<td>18</td>
</tr>
<tr>
<td>2.3.1 Phonotaxis and Sound Localization</td>
<td>18</td>
</tr>
<tr>
<td>2.3.2 The Central Auditory Pathway</td>
<td>19</td>
</tr>
<tr>
<td><strong>3 Methods</strong></td>
<td>23</td>
</tr>
<tr>
<td>3.1 Optimization Approach</td>
<td>23</td>
</tr>
<tr>
<td>3.1.1 Training Protocol</td>
<td>23</td>
</tr>
<tr>
<td>3.2 Monte Carlo Method</td>
<td>24</td>
</tr>
<tr>
<td>3.2.1 Metropolis–Hastings Algorithm</td>
<td>25</td>
</tr>
<tr>
<td>3.3 Implementations</td>
<td>26</td>
</tr>
</tbody>
</table>
 CONTENTS

3.3.1 Software ......................................................... 26
3.3.2 Sound Localization Stimuli ................................. 27
3.3.3 Feature Detection Circuit and Stimuli .................... 28

4 Results ............................................................. 31
  4.1 Sound Localization .......................................... 31
  4.2 Feature Detection ........................................... 32

5 Discussion .......................................................... 37
  5.1 Conclusions .................................................. 38
  5.2 Future Work .................................................. 39

A Bias Parameter Values ............................................ 41
Preface

The work presented in this thesis was conducted during the spring semester and the summer of 2018 at the division Embedded Internet Systems Lab (EISLAB) at Luleå University of Technology (LTU), under the supervision of Assoc. Prof. Fredrik Sandin. This thesis is submitted for the degree of Master of Science in Engineering Physics and Electrical Engineering at LTU.

I would like to thank my supervisor Assoc. Prof. Fredrik Sandin for his excellent guidance and support during this project, and also for introducing me to the fascinating field of neuromorphic engineering.

I would also like to thank Prof. Giacomo Indiveri of the Institute of Neuroinformatics (INI) at UZH and ETH Zurich for his advice and support, and Nicoletta Risi for providing example bias parameter settings for the Dynap-SE neuromorphic processor. Further, I would like to thank Federico Corradi, at the time at iniLabs, for demonstrating how to set up and configure the Dynap-SE, as well as Carsten Nielsen at aiCTX for his advice and support concerning the Dynap-SE.

I acknowledge financial support from the strategic research and innovation area Intelligent Industrial Processes at LTU in the form of a stipend for my participation in the 2018 CapoCaccia Cognitive Neuromorphic Engineering Workshop.

Finally, I would like to thank my family for all of their love and support.

Mattias Nilsson
Luleå, Sweden
September, 2018
Acronyms

AdEx  Adaptive Exponential Integrate-and-Fire. 9, 10, 14, 15
ANN  Artificial Neural Network. 8, 11
BO  Bayesian Optimization. 3
CAM  Content-Addressable Memory. 12, 14, 24, 25
CMOS  Complementary Metal-Oxide-Semiconductor. 2, 12
CNN  Convolutional Neural Network. 3
DE  Differential Evolution. 3
DNN  Deep Neural Network. 3
DPI  Differential Pair Integrator. 10, 14, 15
DVS  Dynamic Vision Sensor. 3
DYNAP  Dynamic Neuromorphic Asynchronous Processor. 2, 3
EPSC  Excitatory Post-Synaptic Current. 10, 14, 15
FPGA  Field-Programmable Gate Array. 25, 26
HIL  Hardware-in-the-Loop. 4, 22
IPSC  Inhibitory Post-Synaptic Current. 10, 14, 15
ISI  Inter-Spike Interval. 25, 27
LGMD  Lobula Giant Movement Detector. 3
Acronyms

**LTD** Long-Term Depression. 11

**LTP** Long-Term Potentiation. 11

**MCMC** Markov Chain Monte Carlo. 21–24

**MH** Metropolis-Hastings. 22–24

**PIR** Post-Inhibitory Rebound. 19, 26, 31, 33, 34

**ROLLS** Reconfigurable On-Line Learning Spiking neuromorphic processor. 5, 11, 12

**SADE** Self-Adaptive Differential Evolution. 3

**SNN** Spiking Neural Network. 4, 8

**SRAM** Static Random-Access Memory. 12, 14, 24, 25

**UAV** Unmanned Aerial Vehicle. 3

**VLSI** Very-Large-Scale Integration. 2, 8, 11
Chapter 1

Introduction

1.1 Background

Animals interact with complex and changing environments in the world in an effortless manner, which requires general problem solving and adaptation capabilities that are challenging to reproduce with technology. Even after vast developmental progress in hardware, software, and system concepts for digital computers, these are still far outperformed by biological brains in a wide range of tasks. This is true in particular when the required power consumption is considered. One example is the intelligence observed in the honeybee as it is foraging for nectar, with regards to tasks like navigation and social dynamics. This behaviour is achieved with a power consumption below a milli watt, using less than a million neurons along with ionic device physics with a bulk mobility that is about 10 million times lower than that of electronic circuits [1]. In comparison with current autonomous robots and neural simulations, both the capacity to solve tasks and the power-efficiency displayed by the foraging honeybee is considered higher by many orders of magnitude.

The principle by which intelligent behavior is generated at such low cost in nature is not yet understood. Substantial progress has, however, been made within neuroscience toward describing the components, architectures of connection, and computational processes that constitute the brain, all of which differ fundamentally from current technology. In the brain, processing is distributed across hundreds of thousands to billions of elementary units, the neurons. Each neuron is, in turn, connected to thousands of others, receiving input through specialized modifiable connective structures, called synapses. The input is collected, and transformed, by the neurons via branched extensions called dendrites. The resulting output is distributed via tree-like
projections of the neurons, called axons. Through the spatial arrangements of the neurons, and the electrochemical interactions on the neuronal dendritic trees, memory is instantiated through the synaptic connections between neurons, and is thereby co-localized with processing [1]. Synaptic plasticity mechanisms allow animals to withhold important memories over the span of a lifetime, as well as to learn during fractions of a second. The output axons transmit, via their complex branching structures, asynchronous spike events reaching synapses on the dendrites of target neurons.

Neuromorphic engineering, a concept first developed by Carver Mead [2], focuses on the use of Very-Large-Scale Integration (VLSI) systems containing electronic analog circuits to imitate neuro-biological structures present in the nervous system, and thereby, ideally, realizing a technology for information processing that combines the organizing principles of the nervous system with the greater charge carrier mobility of electronic circuits [1].

1.1.1 Artificial Neural Networks

The investigation and development of artificial neural network models can roughly be categorized in three successive generations [3].

The first generation of neural network models is based on the use of McCulloch-Pitts neurons as computational units. Multilayer perceptrons, Hopfield nets, and Boltzmann machines are examples of neural network models that stem from the first generation. A characteristic feature of neural network models in the first generation is that the neurons are only able to produce digital outputs. Such networks are universal for computation with digital input and output. Any boolean function can be computed by a multi-layer perceptron with a single hidden layer.

The second generation of neural networks is based on computational units that apply an activation function with a continuous set of possible output values to a weighted sum, or polynomial, of the inputs. A sigmoid function or a linear saturated function is commonly used as the activation function. Typical examples of network models from the second generation are feed-forward and recurrent sigmoidal neural networks, and networks with radial basis function units. These neural networks are also able to compute arbitrary boolean functions, and can, furthermore, perform computations with analog input and output.

When interpreting neural networks of the second generation in a biological perspective, the output of a sigmoidal unit represents the rate at which a biological neuron fires. These networks are, in that sense, more biologically realistic than models from the first generation, considering that biological neurons are known to fire at different frequencies between a minimum and a
1.1. BACKGROUND

Experimental evidence accumulated within neurobiology indicates that the timing of single action potentials, so-called spikes, is used for the encoding of information in some neural systems [3]. Motivated by these results, the third generation of neural network models are the Spiking Neural Networks (SNNs), which, in contrast to neural network models of the earlier generations, employ spiking neuron models as computational units. As SNNs incorporate time in the model, they increase the level of biological realism in simulations of neural systems. SNNs are also modeled by analog VLSI systems, by the means of neuromorphic engineering.

SNNs have been shown to hold at least the same computational power as neural networks of similar sizes from the first two generations; that is, multilayer perceptrons and sigmoidal neural networks. Furthermore, some specific functions which require significantly fewer neurons to be computed by an SNN have been demonstrated [3].

1.1.2 Neuromorphic Hardware

Typically, the circuits that make up neuromorphic information processing systems are designed using mixed-mode analog/digital Complementary Metal-Oxide-Semiconductor (CMOS) transistors, and fabricated with standard VLSI processes. Information processing in neuromorphic systems is realized with energy-efficient, asynchronous, event-driven methods, in similarity to the biological systems that are being modeled. Neuromorphic systems are often adaptive and fault-tolerant, and can in principle be configured to display complex behaviors by the combination of multiple instances of simpler elements.

Like the brain, artificial neural processing systems, including neuromorphic systems, are characterized by co-localized memory and computation. Whereas computing systems based on the classical von Neumann architecture are equipped with one or more central processing units physically separated from the main memory, the synapses of a neural network implement both memory storage and complex nonlinear operators, for collective, distributed computation, at the same time [4]. This use of memory structures is the most substantial difference between neuromorphic information processing systems and conventional ones. Significant performance bottlenecks are imposed upon conventional computing architectures by constraints on memory, such as size, access latency, and throughput. Considering these limitations, alongside the ability of biological nervous systems to perform robust computation, by the means of using slow, stochastic, inhomogeneous, and faulty memory and computing elements, neuromorphic computing paradigms, us-
CHAPTER 1. INTRODUCTION

ing advanced and emerging technologies, are considered a solution for the implementation of alternatives to the von Neumann architecture [4].

One platform for neuromorphic hardware, aimed at providing reconfigurable, general-purpose, real-time networks of spiking neurons, is the Dynamic Neuromorphic Asynchronous Processor (DYNAP) [5, 6], which is developed by aiCTX, a spin-off of the Institute of Neuroinformatics (INI) at the University of Zurich and the ETH Zurich. The DYNAP uses a combination of slow, low-power, inhomogeneous sub-threshold analog circuits, and fast programmable digital circuits, allowing for the implementation of architectures for spike-based neural processing with co-localized memory and computation. The work presented here focuses on the use of the Dynap-SE neuromorphic processor, which is based on the DYNAP platform, and which has a scalable architecture for “tiling an arbitrary number of chips for massive neural networks” [7].

A straightforward implementation of an existing SNN is in general not possible because the low-power analog circuits that constitute the neurons and synapses in the chips of the Dynap-SE neuromorphic processor are inhomogeneous, due to device mismatch. Instead, SNNs that are to be implemented on the Dynap-SE need to be adapted by selecting appropriate neuron and synapse circuits. Presently, no general configuration methods and tools exist, that make efficient use of the inhomogeneous analog circuits. Such tools need to be developed before Dynap-SE and similar neuromorphic processors can be used to solve engineering problems with reasonable effort. The work presented in this thesis is an initial investigation preparing for the development and implementation of such automatic SNN configuration methods.

1.1.3 Related Work

In recent work, pre-trained Convolutional Neural Networks (CNNs) and Deep Neural Networks (DNNs) have been mapped into spike-based hardware successfully, attaining remarkable accuracy while minimizing power consumption [8, 9, 10, 11]. Furthermore, Differential Evolution (DE), and Self-Adaptive Differential Evolution (SADE) have been used for hyper-parameter selection in a simulated neuromorphic spiking neural network model of the locust Lobula Giant Movement Detector (LGMD) driven by the output of a neuromorphic Dynamic Vision Sensor (DVS). DE and SADE were applied to find optimal parameters for an obstacle avoidance system on a Unmanned Aerial Vehicle (UAV), and was shown to outperform state-of-the-art Bayesian Optimization (BO) used for comparison [12].
1.2 Motivation and Applications

There is an increasing need for automation when addressing various engineering problems of the present day. The growing complexity of the systems being developed, such as the Internet of Things (IoT), large automation systems in cities, and increasingly complex factories, along with the fact that the number of problems that emerge in these systems are too great to all be attended to manually by engineers, give rise to a need for machines and processes that are able to learn independently to a greater extent.

Machine learning methods for pattern recognition and control are for example used to make industrial sensor systems increasingly autonomous. One important application area for such methods is in small, resource-constrained, wireless sensor systems. As the energy typically available in such systems, by feasible means of harvesting, is of the order of magnitude of 1 mW, see for example [13], the possibility for applications of machine learning methods is limited. Previous research at LTU, in co-operation with SKF, has shown that machine learning methods can outperform conventional methods in the detection of, for example, faulty wind-turbine components, by the analysis of vibration data [14]. However, when implementing these methods in conventional digital computers, several orders of magnitude more power is required. A significant increase in energy efficiency is required for the implementation of integrated machine learning methods for advanced pattern recognition tasks in wireless sensors with complex streaming data.

One possible approach to address the problem of integrating advanced machine learning methods in resource-constrained devices is the use of low-power neuromorphic hardware, such as the Dynap-SE neuromorphic processor. The introduction of this kind of energy-efficient processors can also serve to reduce the need for centralized computation and communication of vast amounts of digital sensor data. Thus, the use of neuromorphic hardware can counter-act the unsustainable development of digital computing and communication systems, where the demand for energy increases rapidly. On a related note, the production of data in the world is increasing rapidly and far exceeds the storage capacity. This trend motivates improvement of the techniques for processing and storage of data, including neuromorphic hardware and other efficient biomimetic techniques, such as DNA-based storage of digital data.

As the behavior of neural systems is mainly determined by the kinds of nerve cells involved, as well as the manner in which the cells are connected, the methods needed to configure neuromorphic systems are radically different to the methods used when programming conventional computer systems. Rather, the use of neuromorphic systems calls for knowledge about dynamical
systems and neuroscience. Furthermore, neural networks to be implemented in neuromorphic systems need to be designed and trained with the specific properties and limitations of the hardware under consideration, in particular the variations caused by device mismatch and the constraints of the architecture, along with the demands of the specific application being considered.

1.3 Problem Description

The project presented in this report aims at the investigation of a method for automatic configuration, and the development of software for training and configuration, of neural networks in the Dynap-SE neuromorphic processor, focusing in particular on the application of Monte Carlo optimization for the selection of neurons when implementing small neural networks in the Dynap-SE. The following tasks are considered in the work done within this project:

- Building on the former work in [15], and my experience of Monte Carlo simulations in the second-cycle course F7035T Statistical Physics and Thermodynamics, develop a Hardware-in-the-Loop (HIL) based software implementation of a Monte Carlo optimization algorithm for automatic configuration of SNNs in the Dynap-SE neuromorphic processor.

- Assess the performance of the developed solution by applying it to the mapping of a small neural network in the early auditory system of crickets to the Dynap-SE, and test the resulting system on a one-dimensional sound localization task.

- If the former test is successful, investigate the possibility to map a more complex auditory feature detection circuit in crickets to the Dynap-SE using the developed optimization method.

1.3.1 Delimitations

The work presented in this thesis is limited to an investigation of Monte Carlo optimization of the mapping of small neural networks to the Dynap-SE neuromorphic processor using random sampling, as an initial step of investigating methods and developing software for training and configuration of neural networks in the Dynap-SE. Methods for estimating the properties of individual neurons and synapses in the Dynap-SE are not considered, neither
is the optimization of bias parameter values, as is done for the Reconfigurable On-Line Learning Spiking neuromorphic processor (ROLLS) in [12].

Since the work presented here is limited to an initial investigation of the concepts discussed above, ethical aspects of neuromorphic technology development are not taken into any particular consideration. However, in works treating the realization of neuromorphic computation with a closer relationship to, or larger potential for, practical applications, considering ethical aspects may be more relevant.
CHAPTER 1. INTRODUCTION
Chapter 2

Theory

2.1 Neuron and Synapse Models

2.1.1 Spiking Neuron Models

Inter-neuron signals consist of short electrical pulses, so-called action potentials or spikes, with amplitudes of about 100 mV and a typical duration of 1-2 ms, see Fig. 2.1. All spikes emitted by the same neuron look alike, and the pulses do not change form as they propagate along the axon, from one neuron to another. It is rather in the number and timing of the spikes, in a so-called spike train, that information is carried between neurons. The action potential is, by this principle, the elementary unit of signal transmission in nervous systems [16].

Electrophysiological measurements of neuronal dynamics can be reproduced to a high degree of accuracy with conductance-based neuron models. One such model is the Hodgkin-Huxley model [17], which uses differential equations to describe the dynamics of action potentials on the level of ion channels and ion current-flow across the cell membrane. Owing to the complexity of such models, however, the analysis and application to modelling SNNs is complicated. Instead, simple, phenomenological spiking neuron models are typically used for the study of neural coding, memory, and network dynamics [18]. Formal threshold models of neuronal firing are commonly referred to as integrate-and-fire models. In these models, spikes are generated when the neuron membrane potential exceeds a threshold from below, and thereby defining the firing time of the neuron.

Following the notion that, in neural systems, information is carried by the number and timing of spikes, integrate-and-fire models make no attempt at describing the actual shape of action potentials. Rather, incidences of action potentials are reduced to discrete events, occurring at precise moments in
Integrate-and-fire models are, consequently, made up of two separate components necessary to account for the considered neuronal dynamics [18]. The first component is an equation describing the evolution of the membrane potential of the neuron, and the second component is a nonlinear mechanism for generating spikes.

**Adaptive Exponential Integrate-and-Fire Model**

The spiking neuron model that is emulated by the analog neuron circuits integrated in the Dynap-SE neuromorphic processor is the Adaptive Exponential Integrate-and-Fire (AdEx) model. This is a spiking neuron model with two variables, introduced by Brette and Gerstner in 2005 [19]. The AdEx model describes the dynamics of the neuron membrane potential, \( V(t) \), with a nonlinear first-order differential equation, which includes an activation term exponentially dependent on the membrane potential. The membrane potential is coupled to a second equation, describing adaptation of the neuron. By combining the exponential spiking mechanism of the exponential integrate-and-fire model [20] with the adaptation equations of the 2-variable model of Izhikevich [21], the AdEx model is able to accurately predict the spike trains generated by detailed spiking neuron models driven by realistic, realistic...
2.1. NEURON AND SYNAPSE MODELS

<table>
<thead>
<tr>
<th>Name</th>
<th>Symbol</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Membrane Potential</td>
<td>( V )</td>
<td>V</td>
</tr>
<tr>
<td>Membrane Capacitance</td>
<td>( C )</td>
<td>F</td>
</tr>
<tr>
<td>Leak Conductance</td>
<td>( g_L )</td>
<td>S</td>
</tr>
<tr>
<td>Leak Reversal Potential</td>
<td>( E_L )</td>
<td>V</td>
</tr>
<tr>
<td>Spike Threshold</td>
<td>( V_T )</td>
<td>V</td>
</tr>
<tr>
<td>Slope Factor</td>
<td>( \Delta_T )</td>
<td>V</td>
</tr>
<tr>
<td>Input Current</td>
<td>( I )</td>
<td>A</td>
</tr>
<tr>
<td>Membrane Resting Potential</td>
<td>( V_r )</td>
<td>V</td>
</tr>
<tr>
<td>Adaptation Variable</td>
<td>( w )</td>
<td>A</td>
</tr>
<tr>
<td>Adaptation Time Constant</td>
<td>( \tau_w )</td>
<td>s</td>
</tr>
<tr>
<td>Sub-threshold Adaptation</td>
<td>( a )</td>
<td>S</td>
</tr>
<tr>
<td>Spike-triggered Adaptation</td>
<td>( b )</td>
<td>A</td>
</tr>
</tbody>
</table>

Table 2.1: List of AdEx neuron model parameters.

The two coupled nonlinear differential equations that define the AdEx model are

\[
C \frac{dV}{dt} = -g_L(V - E_L) + g_L \Delta_T e^{(V - V_T)/\Delta_T} - w + I, \quad (2.1a)
\]

\[
\tau_w \frac{dw}{dt} = a(V - E_L) - w, \quad (2.1b)
\]

with the parameters defined in Table 2.1. The nonlinear exponential term in Eq. (2.1a) models the spike generation process with an upswing of the action potential. The model is defined such that a spike occurs at time \( t_{\text{spike}} \) when the membrane potential \( V \) diverges towards infinity. The decrease of the action potential that follows the occurrence of a spike is not described by this model. Instead, the voltage is reset to a membrane resting potential \( V_r \):

\[
\text{at } t = t_{\text{spike}}, \quad V \rightarrow V_r, \quad (2.2a)
\]

and, at the same time, the adaptation variable \( w \) is increased by the spike-triggered adaptation \( b \):

\[
\text{at } t = t_{\text{spike}}, \quad w \rightarrow w + b. \quad (2.2b)
\]
2.1.2 Synapse Model

Synapses are essential elements for computation and transfer of information in neural networks. Being highly specialized structures, synapses are involved in the transmission of signals between different neurons, by the means of chemical reactions and ion diffusion. When an action potential generated in a neuron reaches a synapse, it triggers the release of neurotransmitters that, in turn, give rise to a flow of ionic currents moving either into or out of the membrane of the post-synaptic neuron. These ionic current flows can either be Excitatory Post-Synaptic Currents (EPSCs), which increase the membrane potential of the post-synaptic neuron, or Inhibitory Post-Synaptic Currents (IPSCs), which decrease that membrane potential. Post-synaptic currents can span up to several hundreds of milliseconds [22].

In the Dynap-SE neuromorphic processor, synapses are implemented in the form of Differential Pair Integrator (DPI) circuits. The dynamic behavior of these circuits can be characterized by a first-order linear differential equation in terms of the synaptic input and output currents, $I_{in}$ and $I_{out}$, as

$$\tau \frac{d}{dt} I_{out} + I_{out} = I_{th} I_{in},$$

(2.3)

where $\tau$ is the synaptic time-constant, and $I_{th}$ and $I_{r}$ are circuit parameters [23].

2.1.3 Synaptic Plasticity and Learning

In models of neural networks, the strength of a connection from neuron $j$ to neuron $i$ is characterized by a single parameter $w_{ij}$ called the synaptic weight, synaptic strength, or synaptic efficacy. If $w_{ij}$ was constant, it would be expected that the amplitude of the response of a postsynaptic neuron $i$, upon receiving an action potential from a presynaptic neuron $j$, would always be the same. However, experiments in electrophysiology show that the response amplitude can indeed vary over time [18]. Changes in synaptic strengths are referred to as synaptic plasticity.

Changes of the postsynaptic response, lasting for hours, or even days, can be induced by appropriate stimulation [18]. Long-lasting changes that constitute an increase of the synaptic efficacy are referred to as Long-Term Potentiation (LTP), while, if the synaptic efficacy instead decreases, the change is called Long-Term Depression (LTD). These persistent changes of synaptic efficacy are regarded as the neuronal mechanisms of memory and learning. The long-term changes of synaptic efficacy are considered the basis of long-lasting memories. In contrast, phenomena that are referred to as short-term
synaptic plasticity, such as synaptic facilitation or depression, only affect the strength of a synapse during a few seconds [18].

In the theory of Artificial Neural Networks (ANNs), the weight $w_{ij}$ of a connection between the neurons $j$ and $i$ is, for a given task, considered a parameter that is subject to change when optimizing the behavior of the network. Here, the process of adapting the weight parameters is referred to as learning, and the procedure according to which this is done is called a learning rule. Learning is, in this regard, meant in a broad sense, covering the phenomenon of synaptic changes during development, as well as specific changes necessary, for instance, for the recognition of a particular pattern of stimulation.

In attempts to implement compact, low-power, artificial neural processing systems with real-time on-line learning abilities, VLSI devices that comprise neuromorphic circuits implementing spike-based plasticity mechanisms have been designed. One such device is the ROLLS neuromorphic processor, presented in [5]. Another master’s thesis project investigating the possibilities of making use of the ROLLS processor as a pre-processing and feature learning system in a condition monitoring application is presented in [24]. That work demonstrates the potential for a hardware system based on the ROLLS neuromorphic processor to be used for resource-constrained on-line monitoring applications. However, also in that case, it is necessary to carefully tune bias values and to exploit the variations due to device mismatch, which is the focus of the work presented here.

2.2 The Dynap-SE Neuromorphic Processor

2.2.1 Neuromorphic Processor Overview

The Dynap-SE neuromorphic system comprises four chips, each having four cores [6]. Each core comprises 256 neurons, and each neuron has a maximum fan-out of 4k. The Dynap-SE has hierarchical asynchronous routers in three levels, and embedded asynchronous Static Random-Access Memory (SRAM) and Content-Addressable Memory (CAM) memory cells that are distributed across the cores and routers. The chips of the Dynap-SE are fabricated using a standard 0.18 µm 1P6M CMOS technology.

The Dynap-SE is designed following a mixed-signal approach, with implementations of neuron and synapse functions and state-holding properties using parallel analog circuits, and time-multiplexed digital spike-routers. The analog circuits are operated in the sub-threshold domain, thus minimizing the dynamic power consumption, and enabling implementations of neural and
synaptic behaviors that are biophysically realistic, with biologically plausible temporal dynamics [6].

2.2.2 Spike Routing Architecture

The mixed-mode routing architecture of the Dynap-SE adopts a hierarchical routing scheme with three distinct levels of routers. At the lowest level, one "R1" router in each core handles local traffic by either sending back events to its local core, or sending them to the next hierarchical level. Each R1 router has four SRAM cells for every neuron of the core paired to the router. Each of these SRAM cells contains a 20-bit word that consists of a 10-bit tag address, a 6-bit header, and a 4-bit destination core ID. The 10-bit tag address contains the identity of the neuron, local to its chip. The source of events being transmitted from neurons holding the same core and neuron ID, but that are situated on different chips, are thus indistinguishable from each other on the receiving end of the event. The 6-bit header encodes the destination chip of transmitted events. In this header, 2-bits are used to encode the $\Delta X$ number of hops, and 2-bits for the $\Delta Y$ hops; one sign bit is used for the X direction, east or west, and one sign bit for the Y direction, north or south. Thus, sixteen chips in total can be addressed.

Spikes that are sent to the R1 router in the local core are broadcast to all nodes within that core. Incoming events are then, at each neuron, presented to the CAM block, and are considered as inputs by the neuron if there is a match of the tag addresses. Events targeting a different core are sent to the second level router "R2", which communicates with both local cores and the next level router through bidirectional channels, see Fig. 2.2. The highest level router "R3" is employed to transmit data packages over longer distances, such as between different connected Dynap-SE units, and uses the relative 2D-mesh routing strategy mentioned above, which is called XY routing.

2.2.3 Neural Processing Nodes

The memory and computing operations of the architecture are implemented in nodes, see Fig. 2.3, each comprising one AdEx neuron circuit, four synaptic dynamics circuits, 64 10-bit CAM words, and 64 2-bit SRAM cells. The asynchronous CAMs are used to store the tags of the source addresses that are connected to the neuron, while the SRAM cells are used to program what type of synaptic dynamic circuit that is to be used for each connection. The four available types of synaptic behavior are: fast excitatory, slow excitatory, subtractive inhibitory, and shunting inhibitory, respectively. Each synapse type is modeled by a dedicated DPI circuit, with globally shared bias values
Figure 2.2: Spike routing architecture of one multi-core chip in the Dynap-SE neuromorphic processor, comprising 256 neurons per core. The first three levels of spike routing are illustrated, which support communication within and between different cores and chips. More comprehensive illustrations, along with the original description, are available in [6].
that determine synaptic weights and time constants. When the address of an event transmitted to a core match the one stored in the CAM cell of a synapse, the synapse accepts the address-event, i.e. spike, and, in turn, a local pulse-generator circuit is triggered, driving a pulse-extender circuit to produce a square wave. The square waves, having tunable widths ranging from fractions of $\mu$s to tens of ms, are then fed into the synaptic DPI circuit which, upon integrating the square waves over time, produces an EPSC or IPSC with corresponding weight and temporal dynamics. When the synaptic currents integrated by a neuron eventually reaches its spiking threshold, the neuron produces an output spike event that it sends to its handshaking block.

The neurons of the Dynap-SE are implemented using AdEx neuron circuits of the kind described in detail in [5], with schematics presented in Fig. 2.4. Synapses, employing biophysically realistic synapse dynamics, are implemented using sub-threshold DPI circuits of the kind described in [23]. These circuits produce EPSCs and IPSCs, with time constants ranging from a few $\mu$s to hundreds of ms. In order to minimize the circuit footprint, relatively small capacitors, in sizes of about 1 pF, are used for the DPIs [6]. The behavior and dynamics of the neurons and synapses are governed by analog circuit parameters that are set by programmable on-chip temperature compensated bias-generators. For example, the constants in Eqs. (2.1) and (2.2) are set with the bias generators, as well as the other parameters governing the dynamics of neurons and synapses. In order to provide independent sets of biases for core pairs, two independent bias-generator-blocks exist, allowing
for the modeling of different types of neural populations [6].

Owing to the analog nature of the circuits in the Dynap-SE, variations exist in the dynamic behavior of its silicon neurons and synapses. These variations are analogous to differences in values of the parameters that govern the differential equations modeling the neural and synaptic dynamics, and are due to, for instance, differences in the physical dimensions of the transistors and capacitors on the silicons surfaces of the chips. This phenomenon is referred to as device mismatch in semiconductor technology.

### 2.2.4 Software

The Dynap-SE neuromorphic processor is supported by cAER, an open-source event-based framework for neuromorphic devices, written in C and C++, that targets embedded and desktop systems [25]. For interaction with the Dynap-SE, this framework provides a system of modules containing several tools for configuration and monitoring. cAER uses another C library, libcAER, to access, configure and retrieve data from the Dynap-SE. This library provides the basic functions for accessing and programming the memory cells of the Dynap-SE.
CHAPTER 2. THEORY

2.3 Acoustic Communication in Crickets

This section is based on the description of acoustic communication in crickets in [26], which focuses on the neural mechanisms for the production, detection, and recognition of songs.

Male crickets sing in order to attract mates, to promote copulation, and whilst engaged in aggressive interactions with rivals. When a sexually receptive female hears the long-distance mate-attraction signal, a so-called "calling song", of a male belonging to the same species, the female cricket moves towards the source of the sound by walking or flying. This behavior is called phonotaxis. The male cricket also has a distinct "courtship song", which is part of a multi-modal display for the enticement of females to mate, as well as a "rivalry song" that is used during aggressive encounters with rivals, in particular following a victory.

2.3.1 Phonotaxis and Sound Localization

Female crickets are able to approach singing males using nothing but the song as a cue. Researchers have, by using artificial stimuli generated with computers in place of the songs of actual males, learned what it is that females actually find attractive about a song. It is found that females respond most reliably to artificial stimuli that match both the sound frequency and the temporal pattern of the songs of their own species.

The ears of crickets are most sensitive to the sound frequency dominant of calling songs, why other frequencies will be perceived as less intense. However, there is no evidence that crickets are able to qualitatively discriminate between sound frequencies that are similar. Rather, the auditory perception of crickets is, in what is called categorical perception, divided into two broad frequency ranges; one being centered around the sound frequency dominant of calling song, and the other one including a wide range of sound frequencies. Sensitivity to sound of high frequencies is important for responsiveness to courtship song, as well as for the detection of the ultrasonic echolocation calls of hunting bats that prey on crickets and other insects.

Phonotaxis requires, apart from the female cricket recognizing the temporal pattern that is specific for the songs of its own species, the localization of the source of the sound. Sound localization in crickets is based on comparison of acoustic cues in the two ears. Since cricket songs consist of nearly pure tones, the only possible cues for sound localization are differences in timing and intensity of the sound reaching the two ears.

As the distance separating the ears of the cricket is probably too small for the difference in time of the arrival of sound to be resolved by their relatively
2.3. ACOUSTIC COMMUNICATION IN CRICKETS

simple nervous systems, crickets instead make out the difference in intensity of sound using a pressure-difference auditory system, allowing sound to reach both the external and internal surfaces of its eardrum. The two surfaces have path lengths that differ in a manner dependent on direction, giving rise to a direction-dependent interference between sound acting externally and internally, which in turn generates changes, dependent on direction, in the effective amplitude of the stimulus at the “eardrum” of up to 15 dB. Behavioral tests are stated to show that crickets are able to resolve bin-aural differences in sound intensity below 1 dB.

2.3.2 The Central Auditory Pathway

In the early auditory circuitry of crickets, receptor neurons terminate in the prothoracic ganglion, that is, the cluster of neurons, ganglion, located in the foremost of the three segments in the midsection of the insect body, prothorax. There the receptor neurons provide several auditory inter-neurons with synaptic input. While about a dozen of these neurons have been identified, based on their anatomy and physiology, only the three most well characterized of them have known behavioral roles. Having been named variously by different researchers, these inter-neurons are most commonly referred to as ascending neurons AN1 and AN2, and omega neuron ON1. The designations of AN1 and AN2 as ascending neurons are due to their axons terminating in the brain, while the processes of ON1 are confined to the prothoracic ganglion, which makes ON1 a local inter-neuron. All three of these neurons occur in bilateral pairs, with each member of a pair receiving excitatory input mainly from the receptor neurons of one of the ears.

Being sharply tuned to sound frequencies near that of the calling song of the species, AN1 constitutes the main route by which information about calling song reaches the brain. The temporal structure of the song is encoded in the timing of action potentials in AN1.

AN2 provides a robust response to a broad range of high frequencies, including a strong response to elements of high frequency in the courtship songs of some species. The role of AN2 in communication within species is, however, so far remaining unclear. In contrast, a well established function of AN2 is the detection of ultrasonic echolocation calls of hunting bats, in response to which, AN2 triggers evasive steering during flight.

The ON1 neuron acts to increase bilateral contrast and facilitate sound localization. It does this by, upon receiving excitatory input from one ear, providing inhibitory output onto the AN1, AN2, and ON1 neurons that receive excitatory input from the other ear. ON1, being mainly sensitive to the dominant frequency of calling songs, does also, as opposed to AN1, respond
Figure 2.5: Bilateral early auditory circuitry of crickets, for processing of low-frequency stimuli. LF denotes auditory receptor neurons tuned for low frequencies. Open circles and solid disks indicate excitatory and inhibitory synapses, respectively.

robustly to a broad range of high frequencies, including ultrasound. This allows ON1 to increase bilateral contrast for not only cricket songs, but also for the hunting calls of bats. An illustration of the part of the early auditory circuitry discussed here devoted to the processing of low-frequency stimuli is presented in Fig. 2.5.

An Auditory Feature Detection Circuit

In crickets, decisions about whether or not to respond to acoustic signals take place in the brain, upon which they are relayed to the neural circuits in control of motor response, by descending neurons. Even though the network of brain neurons responsible for temporal filtering is not yet fully understood, selectivity for species-specific temporal features is evident in neurons separated by only two or three synaptic connections from the auditory receptor neurons. The axons of these neurons branch out in the same restricted area of the nervous system as the axon terminals of AN1 do. The selectivity for temporal features, then, appears to rely on the interplay of excitatory and inhibitory synaptic inputs, as stimuli with species-specific inter-pulse intervals give rise to excitation that exceeds the corresponding inhibition.

This selectivity arises from a circuit, presented in [27] and illustrated in Fig. 2.6, in which a coincidence detecting neuron, the local neuron LN3,
receives excitation along two different pathways; one running directly from AN1, and the other one going via an inhibitory neuron, LN2, and a non-spiking delay neuron, LN5. This circuitry results in the excitation being delayed by a long inhibition and then realized as a Post-Inhibitory Rebound (PIR). The duration of the inhibition, and, accordingly, that of the delay of excitation, matches the specific inter-pulse interval of the feature that the circuit is specialized for detecting. Thus, the delayed excitation arrives simultaneously with the direct excitation of the subsequent sound pulse at the coincidence detecting neuron LN5, which in turn is able to excite the feature detecting neuron LN4.
Chapter 3

Methods

Since the synapses of the Dynap-SE neuromorphic processor do not include mechanisms for plasticity, neural networks need to be trained by some other means. The idea in the work presented here is to systematically exploit the intrinsic differences between the neuronal and synaptic circuits in the Dynap-SE due to device mismatch, by identifying suitable neuronal circuits for the implementation of a given network architecture.

3.1 Optimization Approach

The optimization of neural network configurations in the Dynap-SE is to be done using a Markov Chain Monte Carlo (MCMC) sampler, and is defined as the minimization of the objective function

\[ U(x) = \sum_i x_i^2, \]  

(3.1)

where, for each output neuron \( i \), \( x_i = \rho_i - r_i \) is the difference between the observed spike rate \( r_i \) and a desired target rate \( \rho_i \).

In order to investigate the potential for training the Dynap-SE according to this concept, a simple neural network architecture in the early auditory circuitry of crickets, presented in Fig. 2.5, is applied to a one-dimensional sound localization task. If the MCMC training protocol is successful on this task, the training algorithm is to be applied to a more complex auditory feature detection network, as illustrated in Fig. 2.6.

3.1.1 Training Protocol

The premise of the training protocol is to, following each MCMC sample of the system, stimulate the network with artificial signals representing the
Figure 3.1: Hardware-in-the-Loop (HIL) simulator, with a computer running cAER that interacts with the Dynap-SE neuromorphic processor [15].

response of early auditory neurons, alternatively the auditory receptor cells of the cricket, to auditory stimuli belonging to some number of predefined classes. The resulting state of the system is then to be assessed by evaluating the objective function in Eq. (3.1) for a stimulus from each class for each MCMC sample.

In order to be able to continuously reconfigure the Dynap-SE and evaluate the objective function, using the MCMC sampler, the training protocol was based on a HIL simulator developed using cAER for interaction with the Dynap-SE. The HIL setup, with a computer that sends commands to refigure the Dynap-SE and then receives the corresponding system response, is illustrated in Fig. 3.1.

Analog membrane voltages in the Dynap-SE are measured with an 8-bit SmartScope USB oscilloscope manufactured by LabNation.

3.2 Monte Carlo Method

The MCMC sampler employed for the optimization task was the Metropolis-Hastings (MH) algorithm [28, 29], with the target distribution

$$P(x) = \frac{1}{Z} e^{-U(x)/\tau},$$

(3.2)

where $U(x)$ is the objective function, $Z$ is a normalizing constant, and $\tau$ is a temperature dependent constant. The objective function is, in terms of physics, employed as the energy function of the system, analogously to how this is done in [30]. Each MCMC sample was defined as the exchange of one neural circuit, employed as a neuron in the network, with another neural circuit, drawn randomly from all of the available neurons not currently involved in the network. By progressing along a indexed list of the network neurons, such an exchange is performed for each iteration of the sampler.
3.2. MONTE CARLO METHOD

3.2.1 Metropolis–Hastings Algorithm

The MH algorithm is an MCMC method for obtaining sequences of random samples from a given probability distribution for which direct sampling is difficult. According to the algorithm, a Markov chain of successive states \( s_1 \to s_2 \to \ldots \) is generated; that is, a sequence of events in which the probability of each event exclusively depends on the state attained in the preceding event. Using a carefully designed transition probability \( \mathcal{P}(s \to s') \), a new state is generated from the old one, such that the change occurs with a probability given by the equilibrium Boltzmann distribution [31]

\[
\mathcal{P}_{\text{eq}}(s) = \frac{1}{Z} e^{-\mathcal{H}(s)/kT},
\]

(3.3)

where \( s \) denotes the state of the system, \( k \) is the Boltzmann constant, \( T \) is the system temperature, \( \mathcal{H} \) is the Hamiltonian, and \( Z = \sum_s e^{-\mathcal{H}(s)/kT} \) is the partition function normalizing the distribution.

At the \( n \)th time step in the Markov process, the state \( s \) occurs with the probability \( \mathcal{P}_n(s) \), described by

\[
\mathcal{P}_{n+1}(s) = \mathcal{P}_n(s) + \sum_{s'} (\mathcal{T}(s' \to s)\mathcal{P}_n(s') - \mathcal{T}(s \to s')\mathcal{P}_n(s)).
\]

(3.4)

With the sum in Eq. (3.4) being over all states \( s' \), the first term in the sum describes all processes reaching state \( s \), while all processes leaving state \( s \) are described by the second term. Eventually, for \( n \to \infty \), the goal is for the probabilities \( \mathcal{P}_n(s) \) to reach a stationary state described by the Boltzmann distribution. It is possible to design the transition probabilities \( \mathcal{T} \) such that, for \( \mathcal{P}_k(s) = \mathcal{P}_{\text{eq}}(s) \), all the terms in the sum vanish. Following this, the detailed balance condition

\[
\mathcal{T}(s' \to s)\mathcal{P}_{\text{eq}}(s') = \mathcal{T}(s \to s')\mathcal{P}_{\text{eq}}(s)
\]

(3.5)

must hold for all \( s \) and \( s' \). According to the condition in Eq. (3.5), the process has to be reversible. Also, with the system having assumed the equilibrium probabilities, the ratio of the transition probabilities is only dependent of the change in energy \( \Delta \mathcal{H}(s, s') = \mathcal{H}(s') - \mathcal{H}(s) \), such that

\[
\frac{\mathcal{T}(s \to s')}{\mathcal{T}(s' \to s)} = e^{-(\mathcal{H}(s')-\mathcal{H}(s))/kT} = e^{-\Delta \mathcal{H}(s,s')/kT}.
\]

(3.6)

Eq. (3.6) can be satisfied by different choices for the transition probabilities \( \mathcal{T} \), but it is possible to show that \( \mathcal{T} \) has to satisfy the general equation \( \mathcal{T}(x)/\mathcal{T}(1/x) = x \), \( \forall x \), where \( x = e^{-\Delta \mathcal{H}/kT} \). One convenient choice that
CHAPTER 3. METHODS

satisfies this condition is given by the MH algorithm, according to which
\( T(x) = \min(1, x) \), why
\[
T(s \rightarrow s') = \begin{cases} 
\Gamma, & \Delta \mathcal{H} \leq 0 \\
\Gamma e^{-\Delta \mathcal{H}(s, s')/kT}, & \Delta \mathcal{H} \geq 0
\end{cases}
\] (3.7)

where \( \Gamma^{-1} \) represents a Monte Carlo time [31]. In principle, the move between
the states \( s \) and \( s' \) can be arbitrary, but if the separation in energy is too
great between those states the move is not likely to be accepted.

Since the neural networks to be optimized in this work are small and
of trivial complexity, it was decided, for the sake of simplicity, that the
implementation of the MH algorithm was to be limited to low temperatures,
making the transition probability
\( T(s \rightarrow s') \approx 0, \quad \Delta \mathcal{H} \geq 0 \). (3.8)

For the minimization of the objective function \( U(x) \), this means that only the
samples that actually reduce the value of \( U(x) \) will be accepted. This limitation would, for more complex optimization problems, like larger and more
complex neural networks, be likely to cause the optimization to get caught
in a local energy minimum near the starting point. For the investigation of
networks considered in this work, however, this simplified case is sufficient.

3.3 Implementations

3.3.1 Software

The Monte Carlo optimization algorithm was implemented as a custom mod-
ule added to the \( cAER \) framework, making use of the various support func-
tions provided by \( libcAER \). Basic functionality for performing measurements
of the spiking activity of the Dynap-SE was extracted from the module
\texttt{meanRateFilter}. The measurements, set to be performed during a spe-
cific amount of time, serve as the basis for the MCMC sampling. The results
of the measurements, each of which is performed during the time that the
network is presented with one stimulus, is the number of spikes from each of
the neurons in the network. One sample comprises measurements performed
on the network as it is presented with one stimulus of each of the stimulus
classes.

The exchanges of neuronal circuits constituting the MCMC sampling pro-
cedure are implemented by rewriting the CAM and SRAM cells of the neu-
rons in the network. When the circuit selected for the realization of one
particular neuron is to be exchanged, the address of that neuron is replaced in the CAMs of all neurons in the network receiving input from the replaced neuron. The CAM and SRAM of the neuron to be replaced is cleared, so that it stops sending and receiving spikes within the network, and this data is instead written to the newly selected neuron to be used as replacement.

Once the network has been reconfigured for the exchange of one neuron, one stimulus of each class is presented and activity measurements are performed, allowing for the evaluation of the objective function. By comparing the result with the value of the objective function in the previous state of the network, the decision whether to keep the change of neuron, or to revert it, is made according to the optimization algorithm. If the change is to be reverted, all CAMs and SRAMs are restored to their previous states, while, if the exchange is instead to be kept, no additional programming operations are needed.

Stimuli are generated by accessing functions in libcAER for programming of the built in spike generator in the Field-Programmable Gate Array (FPGA) of the Dynap-SE. This spike generator requires information about the virtual address of each spike, that is, the address from which the events seem to be sent from when received by the synapses on the chips. For the generation of a spike train with variable Inter-Spike Intervals (ISIs), information about the ISI that is to follow each generated event is also required. This information is thus provided, in order to generate Poisson spike trains according to the procedure described in Section 3.3.2.

3.3.2 Sound Localization Stimuli

For the task related to the early auditory circuitry of crickets, illustrated in Fig. 2.5, the stimuli representing the response of the auditory receptor cells of the cricket were numerically generated using a Poisson distribution. According to this probability distribution, given an average number of events $\lambda$ during a time interval $t$, the probability of $k$ events to occur during $t$ is

$$P(k \text{ events during } t) = e^{-\lambda} \frac{\lambda^k}{k!}. \quad (3.9)$$

Consequently, assuming an underlying instantaneous firing rate $r$ that is constant over time, the probability for a spike to occur during a short time interval $\delta t$ is given by

$$P(\text{one spike during } \delta t) \approx r\delta t, \quad (3.10)$$

in what is referred to as a homogeneous Poisson process [32]. By this definition, Poisson spike trains are generated by subdividing the time of a stimulus
into intervals of duration $\delta t$. For each of these intervals then, considering a random number $x_i$ from a uniform distribution between 0 and 1, a spike is defined to occur if $x_i \leq r \delta t$. This procedure is appropriate when $\delta t$ is small, that is, when $r \delta t \ll 1$.

The direction of incidence of the sound stimuli were accounted for by setting a higher firing rate $r$ for the Poisson spike train coming from the receptor cells of the ear adjacent to the sound direction, representing the difference in intensity of the sound reaching the different ears, see Section 2.3.1 for further information.

### 3.3.3 Feature Detection Circuit and Stimuli

In order to approximate the effect of the non-spiking PIR of the delay neuron LN5 in the auditory feature detection network, see Fig. 2.6, with the Dynap-SE, a simplification had to be made. Since the Dynap-SE implements spiking neurons and transmits spike-events between neurons, the effect of the connection of LN5 to the coincidence-detection neuron LN3 is instead modeled by substituting LN5 with one excitatory and one inhibitory synapse on LN3 receiving input directly from LN2. Thus, the PIR of the non-spiking inhibitory neuron LN5 is replaced with the sum of an IPSC and an EPSC with different time constants, such that the resulting current mimics the PIR. The resulting, simplified feature detection network is presented in Fig. 3.2.

The response of AN1 to the onset of a sound pulse was modeled by having AN1 spiking with a constant rate, why the output from AN1 to the rest of the network was generated using the FPGA spike-generator.

The remaining three local neurons were mapped to three separate cores, respectively, on one chip of the Dynap-SE. By arbitrarily selecting one initial set of neurons from these cores for the network, they were then tuned with the MCMC protocol to approximately recreate the processing mechanism of the auditory feature detector network, which is presented in [27]. More specifically, the cores were tuned such that the network, in its initial arbitrary configuration, successfully solved the feature detection task. The pair of synapses substituting the connection from LN5 to LN3 were tuned to recreate the PIR acting on LN3, before adding the direct excitation from AN1 and tuning LN3 according to its appropriate behavior in the network.
Figure 3.2: Auditory feature detection network modified to be compatible with the Dynap-SE neuromorphic processor, with a substituted delay mechanism mimicking the non-spiking neuron LN5. Open circles and solid disks indicate excitatory and inhibitory synapses, respectively.
Chapter 4

Results

4.1 Sound Localization

In order to test the optimization algorithm and training protocol outlined in the former chapters, a simple sound localization task is considered using the network in Fig. 2.5. Sound stimuli are generated in the form of Poisson spike trains, with two different firing rates of 110 Hz and 100 Hz for the auditory receptor cells on the adjacent and opposite sides, respectively, in relation to the direction of the sound source. The target rates for the corresponding response of the AN1 neurons were set to 100 Hz and 50 Hz, respectively. The length of each Poisson stimulus are set to one second, thus making the length of each HIL iteration roughly equal to two seconds. The same pair of left- and right-dominant spike trains are used in each iteration of the algorithm.

The application of the algorithm is confined to one chip on the Dynap-SE. The address space of one core is reserved for virtual input neurons, and is thus configured to have no activity. The remaining three chips are used by the training protocol. The bias parameter values used on the active chips are retrieved from sections 5.7.3 and 5.7.5 in [33], followed by manual tuning of the synaptic parameters weights of the fast excitatory and inhibitory synapses, $PS\_WEIGHT\_EXC\_F\_N$ and $PS\_WEIGHT\_INH\_F\_N$, and of the threshold parameter of the fast excitatory synapses $NPDPIE\_THR\_F\_P$. All bias parameter values that are used are used in this work are presented in Appendix A.

Results from the optimization of the sound localization network are presented in Fig. 4.1, which shows how the minimum value of the objective function, $U(x)$, evolves with the Monte Carlo iterations. Furthermore, Fig. 4.2 shows the spike rates of AN1 corresponding to each minimum value of the objective function. Additionally, results of a trained sound localization net-
4.2 Feature Detection

Given the successful optimization results for the sound localization task, which confirm that the method and software functions as expected, the more complex feature detection problem outlined in Section 3.3.3 is considered next.

The two classes of stimuli are defined as a single and a pair of sound pulses, respectively, represented by AN1 firing at a constant rate under the duration of a pulse. The sound pulses are generated as ten equally spaced spikes, which are virtually transmitted from AN1 with an ISI of 2 ms. The double pulses are generated with an inter-pulse interval of 20 ms. A stimulation time of 500 ms is used for the presentation of each stimulus. The target for the objective function is defined as, for each of the three neurons, a specific number of output spikes when the network is stimulated with a single and a pair of sound pulses, respectively. The target number of spikes are 4 and 8 for LN2,
4.2. FEATURE DETECTION

Figure 4.2: The measured AN1 activity rates corresponding to the minimum values of the objective function during Monte Carlo optimization of the auditory network for sound localization, with sound incident from the left-hand side (a) and the right-hand side (b), respectively. Dashed lines connecting data points do not represent actual data, but serve to illustrate the trends. The black dashed lines mark the target spike rates, $\rho_i$.

Figure 4.3: Objective function $U(x)$ during repeated stimulation of the trained neural network for sound localization, alternating between stimuli corresponding to the left- and right-hand sound sources within each stimulation cycle.
Figure 4.4: The measured AN1 activity rates during repeated stimulation of the trained neural network for sound localization, alternating between stimuli representing sound incident from the left-hand side (a) and the right-hand side (b) within each stimulation cycle. The black dashed lines mark the target spike rates, $\rho_i$.

2 and 6 for LN3, and 0 and 1 for LN4, in line with the reported behavior of the network in crickets [27]. The optimization algorithm is initiated by randomly selecting neurons from three of the cores of the Dynap-SE for the mapping of each neuron.

Results from the optimization of the auditory feature detection network are presented in Fig. 4.5, which shows the minimum value of the objective function, $U(x)$, during the Monte Carlo iterations of five independent runs of the optimization algorithm. Furthermore, Fig. 4.6 shows the resulting PIR in LN3 of an optimized feature detection network, obtained with the pair of excitatory and inhibitory synapses. The resulting processing mechanism of the feature detection network, after optimization, is demonstrated in Fig. 4.7, by showing the measured potentials of its neurons upon stimulation with a single and a pair of sound pulses respectively.
4.2. FEATURE DETECTION

Figure 4.5: Evolution of the minimum value of the objective function $U(x)$ during five independent runs of the Monte Carlo optimization algorithm applied to the auditory feature detection network. Dashed lines connecting data points do not represent actual data, but serve to illustrate trends.

Figure 4.6: Resulting PIR in LN3 of the feature detection network obtained with the excitatory-inhibitory synapse pair after optimization, without the direct excitation from AN1 for clarity.
Figure 4.7: Measured potentials of neurons in the optimized feature detection network, when stimulated with a single (left column) and a pair (right column) of sound pulses, with a pulse duration and pulse interval of 20 ms.
Chapter 5
Discussion

Considering the result presented in Fig. 4.1, the optimization of the sound localization network on the Dynap-SE with respect to the objective function, $U(x)$, defined in Eq. (3.1) is successful. However, the results were highly dependent on the choice of bias parameters values. Fig. 4.2 shows that the biases had to be tuned to such an extent that the system in fact already did solve the classification problem initially, although the optimization caused substantial improvements of the objective function, which is dependent on the target spike rates $\rho_i$ for the AN1 neurons. Figs. 4.3 and 4.4 show how, once optimized, the network is able to reliably solve the sound localization task while operating in a relatively stable manner.

The sound localization network displays a high degree of dependence on the specific signals used for stimulation. The only experiments that proved successful were only one pair of Poisson spike trains were repeatedly used for training and evaluation. In cases were new, unique Poisson spike trains were generated for each stimulation, the system was unpredictable and showed no sign of convergence during training. No further consideration or fine-tuning was, however, performed to solve this problem, since the sound localization task was considered to test the validity of the Monte Carlo optimization algorithm and software, which are found to function as expected considering the minimization trend of the objective function in Fig. 4.1.

The application of the optimization algorithm to the auditory feature detection network proved successful in reaching configurations that correctly performed the feature detection task. In this case, the network required a smaller number of successful neuron exchanges before the objective function reached a minimum value of zero or one, see Fig. 4.5. However, given the limited time of twenty weeks for this project, the network is not tuned or optimized to avoid false positives. Consequently, upon stimulation with a
single pulse of double length, the feature detector was excited, indicating a
double pulse when there is none.

Considering the PIR-like behavior of LN3 in an optimized network con-
figuration, see Fig. 4.6, a probable cause of the false positive is the balance
between inhibition and excitation in the PIR. In the original processing mech-
anism of the feature detection network [27], the magnitude of the inhibition
is greater than that of the excitation. The responses of the individual neu-
rons to the sound pulses after optimization show a clear resemblance to the
neurobiological mechanisms that each type of neuron were tuned to model,
see Fig. 4.7. However, it is evident that the inhibition related to the PIR,
which should create a clear separation between the excitation by different
pulses, of both LN3 and LN4, is insufficient in magnitude relative to the
subsequent excitation. It is likely that this problem can be solved with an
extended training set, because units with stronger inhibition are observed
when tuning the bias values to mimic the PIR with a pair of synapses in the
Dynap-SE.

The network applied for sound localization required extensive tuning of
bias parameter values before it could be successfully implemented via op-
timization. The feature detection network was tuned for all three types of
neurons to fit a specific neurobiological behavior. When a network was ini-
tiated with randomly selected bias-tuned neurons, further optimization with
the MCMC-based training protocol was required before the network was able
to perform its task.

5.1 Conclusions

In the work presented in this thesis, an HIL-based software implementation of
a Monte Carlo optimization algorithm for automatic configuration of SNNs
in the Dynap-SE neuromorphic processor is developed. The optimization
algorithm is, with respect to the objective function $U(x)$ defined in Eq. (3.1),
successfully applied to the mapping of a small network in the early auditory
system of crickets assigned with a one-dimensional sound localization task.

Following the successful optimization results for the sound localization
circuit, a more complex auditory feature detection circuit in field crickets
is implemented on the Dynap-SE by substituting a non-spiking delay neu-
ron with a pair of excitatory and inhibitory synapses and tuning the bias
parameter values to fit the neurobiological behavior of the remaining three
specialized neurons in the network. The optimization method is successfully
applied to the auditory feature detection network, in terms of a pattern recog-
nition problem, specified with training data and the objective function $U(x)$. 
The network is, however, not tuned or optimized to avoid false positives. This would require a larger training data-set, as well as further adjustments of the objective function and of the bias parameter values, for the network to more closely resemble the behavior of the biological neural network in a quantitative sense.

Extensive tuning of bias parameter values is required before the optimization experiments are successful. This is time consuming, and should thus be included in the optimization protocol.

### 5.2 Future Work

Neural networks become more complex as they increase in size. Hence, the energy landscape of the neural network objective function increase in complexity as well. The Monte Carlo optimization algorithm presented here is, thus, likely to be too simple to manage the hardware mapping of larger networks in reasonable time. One possible approach to improve the optimizer for larger networks is to investigate improved sampling techniques for rough energy landscapes [31].

The selection of appropriate bias parameter values is fundamental to successfully implementing neural network designs in the Dynap-SE. Thus, any further development of a general method for hardware mapping, alternatively for training networks directly on the Dynap-SE, is likely to benefit from taking methods for selection of bias parameter values into account. Optimization of bias parameter values, as is done in [12], is, at least for some neural network architectures, likely to be a necessary part of making the network mapping to the Dynap-SE automatic to some extent.

The feature detection circuit developed in this work, for pattern recognition tasks using the Dynap-SE neuromorphic processor, is a basis for further development and applications. A successful implementation of this neural network is likely to require further tuning of the bias parameter values, for the processing mechanism to more closely resemble that of the biological neural network. In particular, false positives need to be considered in the tuning of the bias parameter values and in the optimization of the network. Furthermore, the optimizer and software developed in this project can be used to implement other biological neural networks, and to identify efficient ways to realize task-specific computations in the Dynap-SE in general.
Appendix A

Bias Parameter Values

All sets of bias parameter values used for the Dynap-SE neuromorphic processor in the experiments presented in this work, see Sections 4.1 and 4.2, are presented in this appendix. For a brief account of the bias parameters of the Dynap-SE, see Section 2.2.3. Information about the conversion of bias values to the corresponding physical quantities is provided in [33].
Table A.1: Bias parameter values used for the neurons and synapses in the implementation of the sound localization network, see Fig. 2.5, in the Dynap-SE neuromorphic processor.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Coarse Value</th>
<th>Fine Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>IF_AHTAU_N</td>
<td>7</td>
<td>35</td>
</tr>
<tr>
<td>IF_AHTHR_N</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>IF_AHW_P</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>IF_BUF_P</td>
<td>3</td>
<td>80</td>
</tr>
<tr>
<td>IF_CASC_N</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>IF_DC_P</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>IF_NMDA_N</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>IF_RFR_N</td>
<td>4</td>
<td>208</td>
</tr>
<tr>
<td>IF_TAU1_N</td>
<td>5</td>
<td>120</td>
</tr>
<tr>
<td>IF_TAU2_N</td>
<td>5</td>
<td>15</td>
</tr>
<tr>
<td>IF_THR_N</td>
<td>3</td>
<td>20</td>
</tr>
<tr>
<td>NPDPIE_TAU_F_P</td>
<td>5</td>
<td>130</td>
</tr>
<tr>
<td>NPDPIE_TAU_S_P</td>
<td>7</td>
<td>40</td>
</tr>
<tr>
<td>NPDPIE_THR_F_P</td>
<td>1</td>
<td>120</td>
</tr>
<tr>
<td>NPDPIE_THR_S_P</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>NPDPII_TAU_F_P</td>
<td>7</td>
<td>40</td>
</tr>
<tr>
<td>NPDPII_TAU_S_P</td>
<td>7</td>
<td>40</td>
</tr>
<tr>
<td>NPDPII_THR_F_P</td>
<td>7</td>
<td>40</td>
</tr>
<tr>
<td>NPDPII_THR_S_P</td>
<td>7</td>
<td>40</td>
</tr>
<tr>
<td>PS_WEIGHT_EXC_F_N</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>PS_WEIGHT_EXC_S_N</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>PS_WEIGHT_INH_F_N</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>PS_WEIGHT_INH_S_N</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>PULSE_PWLK_P</td>
<td>0</td>
<td>43</td>
</tr>
<tr>
<td>R2R_P</td>
<td>4</td>
<td>85</td>
</tr>
</tbody>
</table>

Table A.2: Bias parameter values used for the inhibitory neuron LN2 in the implementation of the auditory feature detection network, see Fig. 3.2, in the Dynap-SE neuromorphic processor.
### APPENDIX A. BIAS PARAMETER VALUES

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Coarse Value</th>
<th>Fine Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Neuron Parameters</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IF_AHTAU_N</td>
<td>7</td>
<td>35</td>
</tr>
<tr>
<td>IF_AHTHR_N</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>IF_AHW_P</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>IF_BUF_P</td>
<td>3</td>
<td>80</td>
</tr>
<tr>
<td>IF_CASC_N</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>IF_DC_P</td>
<td>0</td>
<td>30</td>
</tr>
<tr>
<td>IF_NMDA_N</td>
<td>1</td>
<td>213</td>
</tr>
<tr>
<td>IF_RFR_N</td>
<td>4</td>
<td>40</td>
</tr>
<tr>
<td>IF_TAU1_N</td>
<td>5</td>
<td>39</td>
</tr>
<tr>
<td>IF_TAU2_N</td>
<td>0</td>
<td>15</td>
</tr>
<tr>
<td>IF_THR_N</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td><strong>Synapse Parameters</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NPDPIE_TAU_F_P</td>
<td>5</td>
<td>125</td>
</tr>
<tr>
<td>NPDPIE_TAU_S_P</td>
<td>6</td>
<td>40</td>
</tr>
<tr>
<td>NPDPIE_THR_F_P</td>
<td>2</td>
<td>30</td>
</tr>
<tr>
<td>NPDPIE_THR_S_P</td>
<td>2</td>
<td>200</td>
</tr>
<tr>
<td>NPDPII_TAU_F_P</td>
<td>5</td>
<td>100</td>
</tr>
<tr>
<td>NPDPII_TAU_S_P</td>
<td>7</td>
<td>40</td>
</tr>
<tr>
<td>NPDPII_THR_F_P</td>
<td>3</td>
<td>60</td>
</tr>
<tr>
<td>NPDPII_THR_S_P</td>
<td>7</td>
<td>40</td>
</tr>
<tr>
<td>PS_WEIGHT_EXC_F_N</td>
<td>0</td>
<td>120</td>
</tr>
<tr>
<td>PS_WEIGHT_EXC_S_N</td>
<td>1</td>
<td>40</td>
</tr>
<tr>
<td>PS_WEIGHT_INH_F_N</td>
<td>1</td>
<td>30</td>
</tr>
<tr>
<td>PS_WEIGHT_INH_S_N</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>PULSE_PWLK_P</td>
<td>5</td>
<td>40</td>
</tr>
<tr>
<td>R2R_P</td>
<td>4</td>
<td>85</td>
</tr>
</tbody>
</table>

Table A.3: Bias parameter values used for the coincidence detecting neuron LN3 in the implementation of the auditory feature detection network, see Fig. 3.2, in the Dynap-SE neuromorphic processor.
Table A.4: Bias parameter values used for the feature detecting neuron LN4 in the implementation of the auditory feature detection network, see Fig. 3.2, in the Dynap-SE neuromorphic processor.
APPENDIX A. BIAS PARAMETER VALUES
Bibliography


